EMSD EG Project - Report

Sin Bo-chi (INFEG2002)

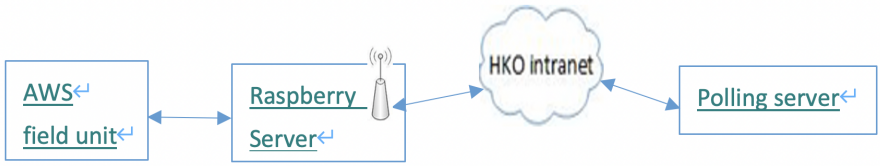
Dates: 19 Oct 2020 (Mon) - 13 Nov 2020 (Fri) [4 weeks]

Mentors: 1) Mr. YEUNG Chi Yu 2926 8043 cyyeung@hko.gov.hk
2) Mr. LAI Wai Lun 2926 8044 wllai@hko.gov.hk

(1) Project Description and Requirements

Project Description

- Project Title: Data Logging and Data Dissemination for Automatic Weather Station (AWS)
- Objectives:
  - Use the Raspberry server to communicate with AWS field unit which is used to sample various meteorological sensors on sites
  - Compile AWS report and wait for polling from HKO polling server. A diagram is shown as below:



- Hardware Provided: 1. Raspberry Pi 4 model B 2. USB to Com port device

Project Requirements (Compulsory) :

1	Setup a Raspberry server including installation of OS, network, clock synchronization ...etc.	✓
2	Implement a program to poll the AWS field unit through the USB to Com port device on every minute	✓
3	Store the polling report with a specified format on the server	✓
4	Setup the Raspberry server with ftp and sftp features.	✓
5	Carry out a communication test between the polling server and Raspberry server via mobile networks	✓
6	Implement a housekeep program to archive the polling report every day	✓

Project Requirements (Additional) :

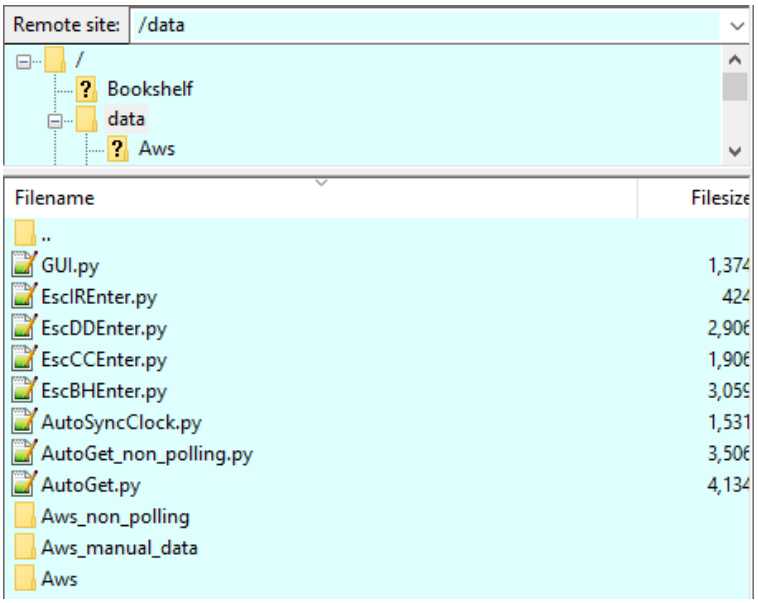
1	Implement a program to Sync Clock through the USB to Com port device on every 3 hour / manual input	✓
2	Implement a program to receive AWS field unit from Special Com port device with no polling command	✓
3	Implement a program to reset AWS field unit through the USB to Com port device	✓
4	Implement a User Interface to perform manual function towards AWS field unit	✓
5	Create different directories for different sources of .txt files	✓

(2) System Schematic Diagram

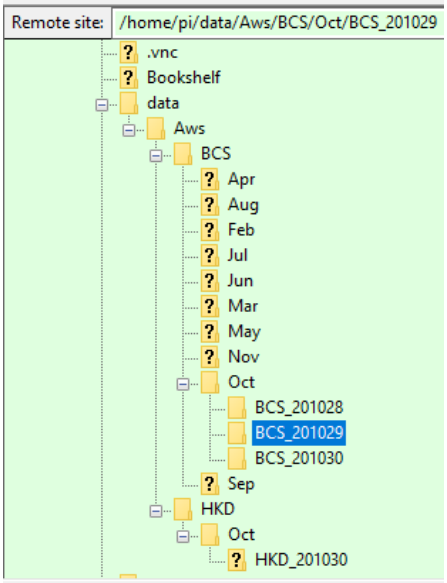
The whole system can be described in three aspects: (1) File structure, (2) Programs.

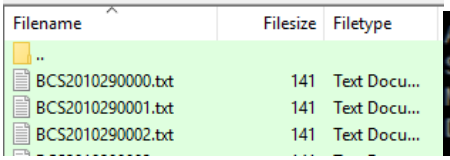
2.1 File structure

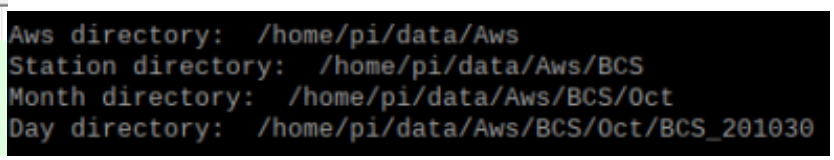
A. Overall structure



Refer to the above figure, the whole system is stored in **/home/pi/DATA**, which the python scripts (programs) and Aws directory are listed inside.







Inside Aws directory, there are different directories containing the data from different station/device (e.g. BCS, HKD in above figure). Inside **/home/pi/DATA/AWS**, the other directories are **all created through AutoGet.py**. When a record is fetched from a polling server, the related directories are created if they are not existing. All of the records are put inside **/home/pi/DATA/AWS/{station_name}** (Station directory) at once, after that be housekeeper into **/home/pi/DATA/AWS/{station_name}/{station_name}_YYMMDD** (Day directory) after Raspberry Pi server's time goes to 0830 on the next day.

2.2 Programs

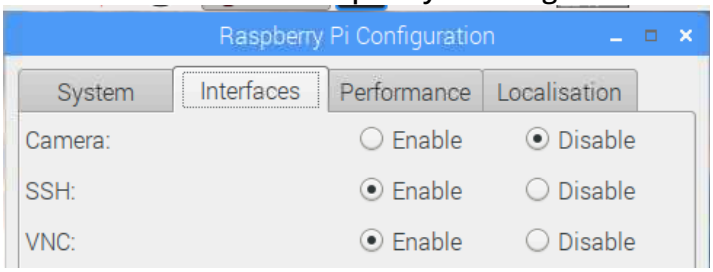
Automation	**AutoGet.py** - Periodically (1 minute): 1. ESC DD ↲ > Get Data 2. ESC BH ↲> Request Backlog Data 3. Store fetched data in .txt format with designated file name and directory Periodically (1 day): 1. Housekeeping function
	AutoGet_non_polling.py - Periodically (1 minute): 1. **Get Data without polling command** 2. Store fetched data in .txt format with designated file name and directory Periodically (1 day): 2. Housekeeping function
	AutoSyncClock.py - 1. ESC CC ↲ > Sync Clock a. From Raspberry Pi Server Clock to AWS Field Unit Clock b. Immediate sync clock **by once** c. Regulatory sync clock **every 3 hours from 2345**
Manual entry	**EscDDEnter.py** - Once: 1. ESC DD ↲ > Get Data 2. Store fetched data in .txt format with designated file name and directory
	EscBHEnter.py - Once: 1. ESC BH ↲> Request Backlog Data 2. Store fetched data in .txt format with designated file name and directory
	EscCCEnter.py - Once: 1. ESC CC ↲ > Sync Clock a. From Manual Input to AWS Field Unit Clock b. From Raspberry Pi Server Clock to AWS Field Unit Clock
	EscIREnter.py - Once: 1. ESC IR ↲ > Reset Field Unit
	GUI.py 1. A graphical user interface for the user to perform **a. EscDDEnter.py** **b. EscBHEnter.py** **c. EscCCEnter.py** **d. EscIREnter.py**

(3) Operation Procedures and Configurations

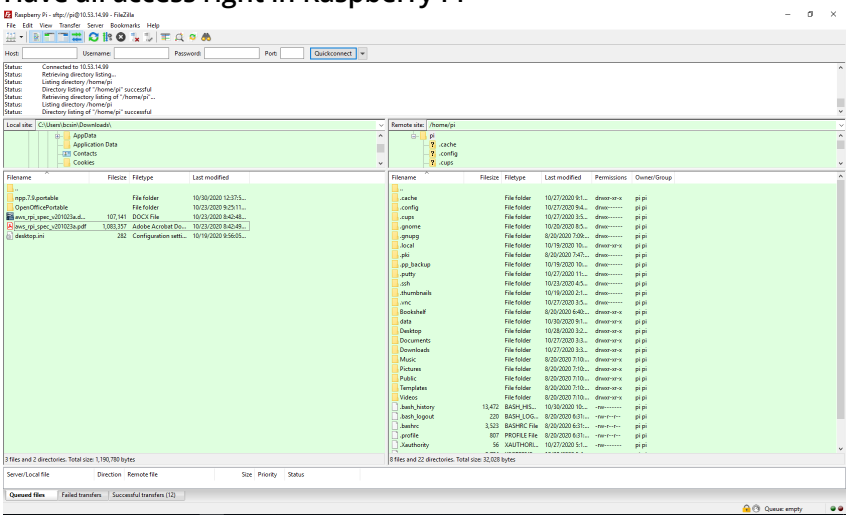
3.1 Operation Procedures

SFTP access through PC

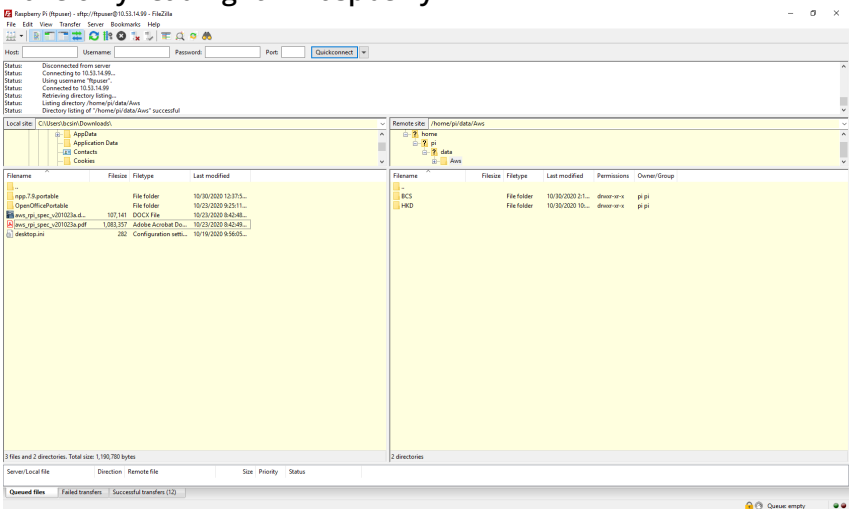
1. Enable VNC interface on Raspberry Pi
   a. Menu -> Preferences -> Raspberry Pi Configuration



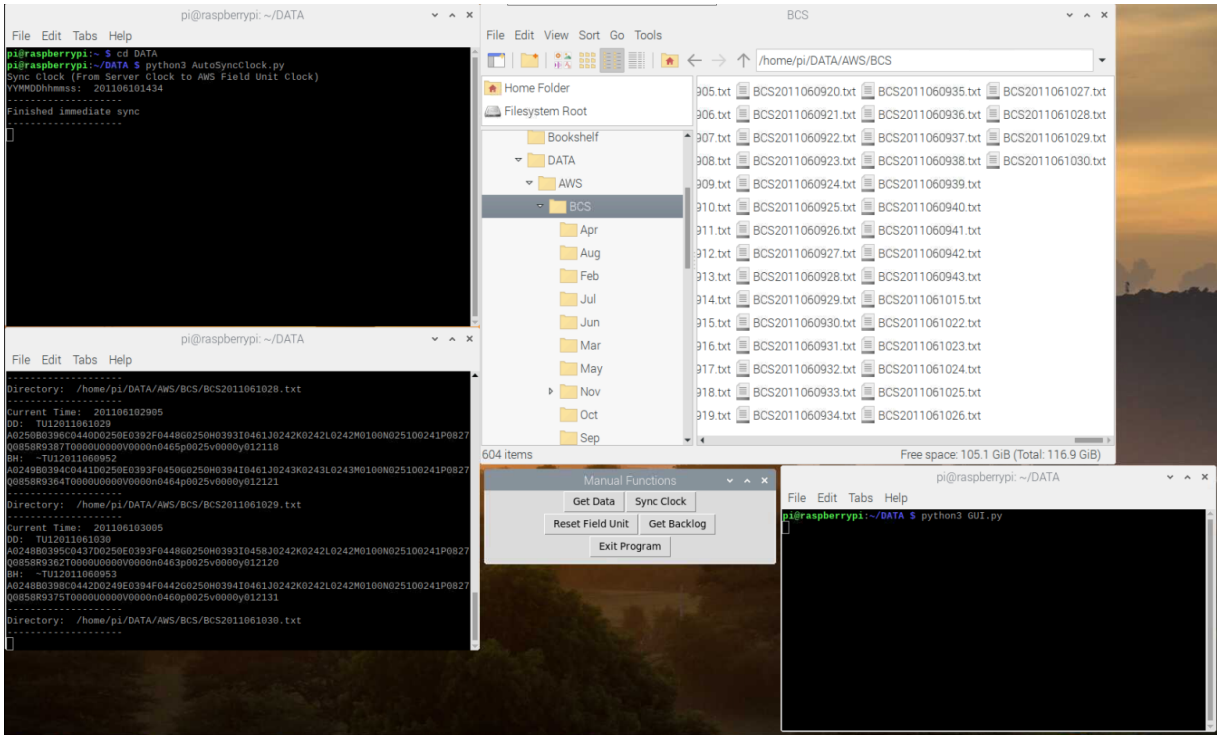
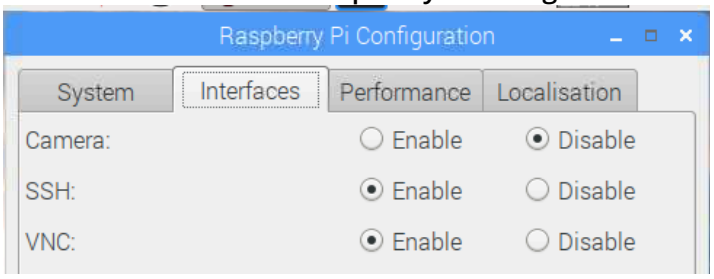
2. admin (User: Pi, Password: RaspberryPi4)
   a. Have all access right in Raspberry Pi



3. user (User: ftpuser, Password: raspberry)
   a. Have only read right in Raspberry Pi

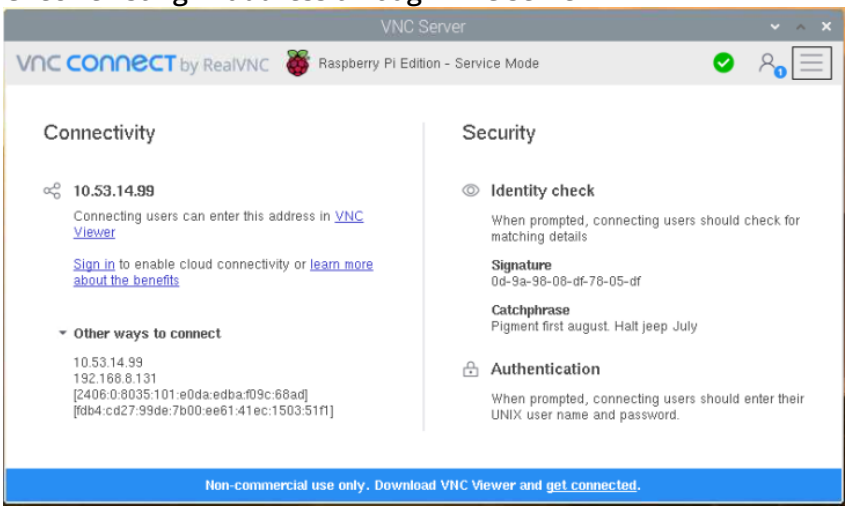


4. LAN: 10.53.14.99
5. WAN: 14.0.133.3

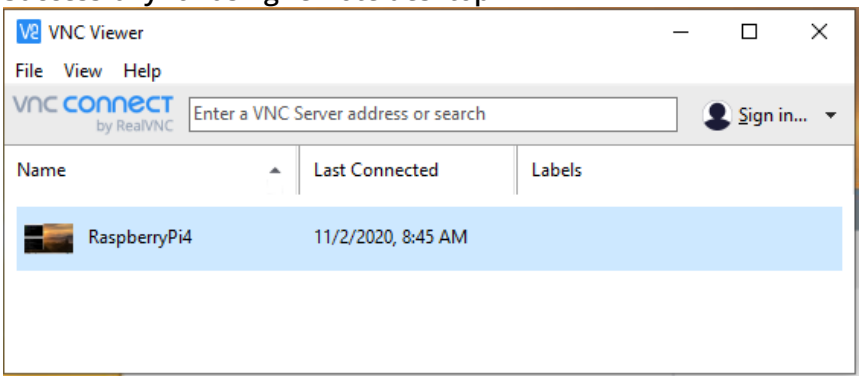
Run Python Script	
	Automation - AutoGet.py, AutoSyncClock.py 1. Open 1 terminal to run 1 python script 2. cd DATA/ 3. python3 AutoGet.py / python3 AutoSyncClock.py 4. Keep terminal open to run the programs continuously
	Manual entry - EscDDEnter.py, EscBHEnter.py, EscCCEnter.py, EscIREnter.py • Two methods a. Run python scripts by the terminal (same as automation) b. Polling Program (GUI.py) i. open a new terminal to run python script (python3 GUI.py) ii. Click on.the buttons and run python scripts
Enable Remote Desktop (VNC)	1. Enable VNC interface on Raspberry Pi a. Menu -> Preferences -> Raspberry Pi Configuration  b. Check IP address through terminal command - ifconfig

```
pi@raspberrypi:~ $ ifconfig
lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 17  bytes 1004 (1004.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 17  bytes 1004 (1004.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.7  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::efe2:42d7:49cf:cb0a  prefixlen 64  scopeid 0x20<link>
        ether b8:27:eb:50:59:dc  txqueuelen 1000  (Ethernet)
        RX packets 3578  bytes 239014 (233.4 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 3233  bytes 1550734 (1.4 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

pi@raspberrypi:~ $
```

c. Check existing IP address through VNC server



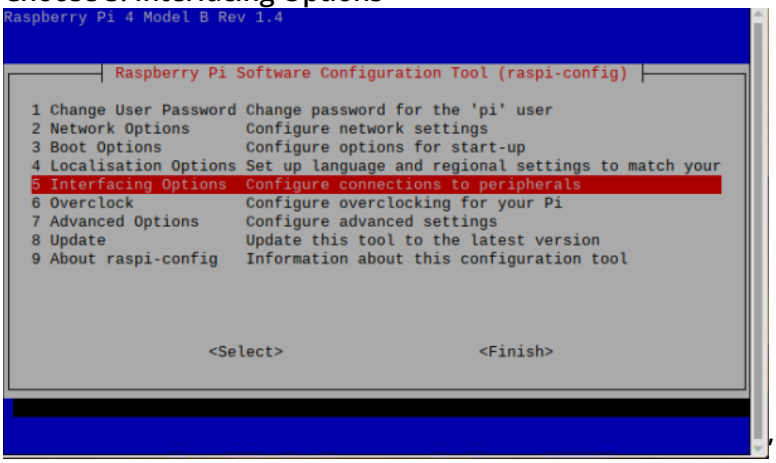
2. Download and install VNC Viewer on a PC computer
   a. Download link: https://www.realvnc.com/en/connect/download/viewer/
   b. Make sure the PC computer and Raspberry Pi server are in the same network
   c. Enter Raspberry Pi's IP address, admin account and password
   d. Successfully for using remote desktop
   e.



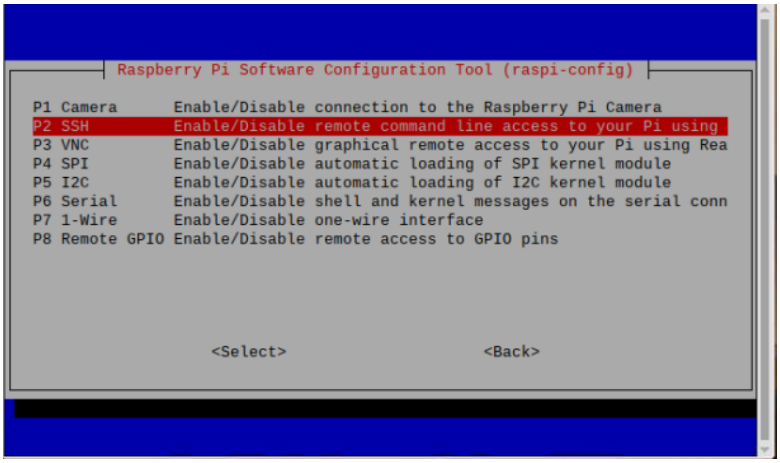
3.2 Configuration

Network Configuration	**SFTP** Server Side: 1. Open SSH inside Raspberry Pi a. Go to terminal -> type 'sudo raspi-config'

b. Choose 5. Interfacing Options



c. Choose P2 SSH



d. Choose YES to enable SSH server

2. vsftpd (management of SFTP server)
   a. Go to terminal -> type ‘sudo apt-get install vsftpd’
   b. Update configuration file
      i. Modify vsftpd configuration file -> type ‘sudo nano /etc/vsftpd.conf’
      ii. Find the following lines and un-comment them by deleting the # character
         1. anonymous_enable=NO
         2. local_enable=YES
         3. write_enable=YES
         4. chroot_local_user=YES

5. utf8_filesystem=YES
6. allow_writeable_chroot=YES
7. local_root=/home/pi/

iii. Save and exit using CTRL-X, Y and ENTER

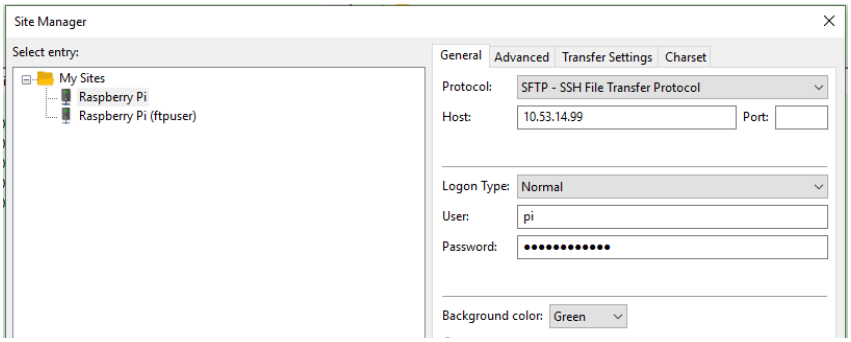
c. Create FTP directory and set permission right for user
   i. mkdir /home/pi/DATA => chmod a-w /home/pi/DATA
   ii. (optional) if new user need to get data from Aws directory
      1. chmod 755 /home/pi/DATA
   iii. Final permission should be like this:

```
pi@raspberrypi:~ $ ls -l data
total 36
-rw-r--r-- 1 pi pi 4934 Oct 29 16:56 AutoGet.py
-rw-r--r-- 1 pi pi 1835 Oct 30 09:53 AutoSyncClock.py
drwxr-xr-x 4 pi pi 4096 Oct 30 10:14 Aws
-rw-r--r-- 1 pi pi 1828 Oct 30 10:09 ESCCCEnter.py
-rw-r--r-- 1 pi pi 4347 Oct 30 10:13 ESCDDEnter.py
-rw-r--r-- 1 pi pi  424 Oct 29 15:35 ESCIREnter.py
-rw-r--r-- 1 pi pi 1381 Oct 29 15:31 GUI.py
```

d. Add new user (can only download files, cannot upload and modify files)
   i. sudo useradd newuser
   ii. sudo passwd newuser
   iii. Enter new UNIX password: raspberry
   iv. Retype new UNIX password: raspberry
   v. passwd: password updated successfully
e. Modify new user's default directory
   i. SFTP
      1. sudo usermod -d /home/pi/DATA newuser
      2. Everytime login to new user account by Putty / Filezilla by SFTP
         a. => default directory is /home/pi/DATA
   ii. FTP
      1. sudo nano /etc/vsftpd.conf
      2. Add the following lines
         a. allow_writeable_chroot=YES
         b. local_root=/home/pi
f. Restart FTP server
   i. sudo service vsftpd restart
g. (optional) Improving SSH security (username/password security)
   i. Only allow specific user to user SSH
      1. sudo nano /etc/ssh/sshd_config
      2. append to the end of the file => AllowUsers newuser pi
      3. sudo systemctl restart ssh

Client-Side - Filezilla

1. Download Filezilla (https://filezilla-project.org/download.php?type=client)
2. Go to site manager addr)



a. Set two connection presets with SFTP (host = Raspberry Pi's IP address)
   i. Admin user - Have all of the access rights
   ii. New user - Only have right to read and execute files

Firewall (ufw)

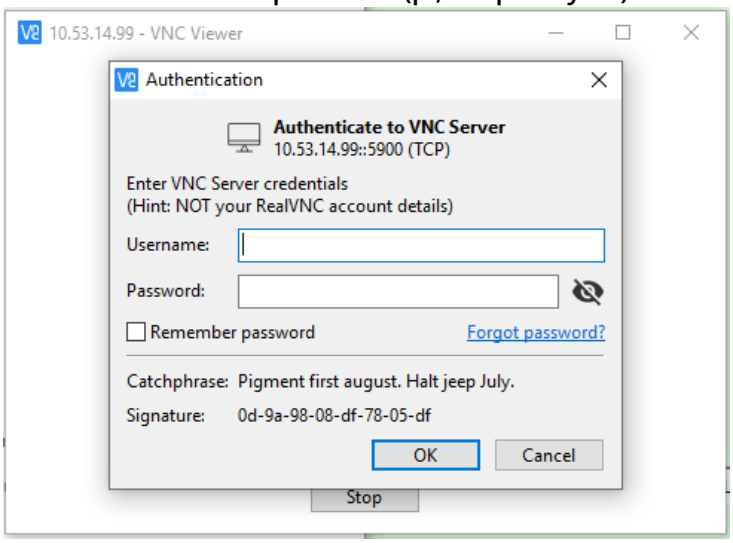
1. Check firewall status - sudo ufw status (Inactive as default)
2. Enable firewall - sudo ufw enable
3. Default setting of all incoming and outgoing connections
   a. sudo ufw default allow outgoing
   b. sudo ufw default allow incoming
4. Allow ssh/SFTP service access - sudo ufw allow ssh
5. Allow FTP service access - sudo ufw allow 20, sudo ufw allow 21, sudo ufw allow 990
6. Allow VNC data for incoming and outgoing - sudo ufw allow 5500, sudo ufw allow 5900
7. Final result should be like below:

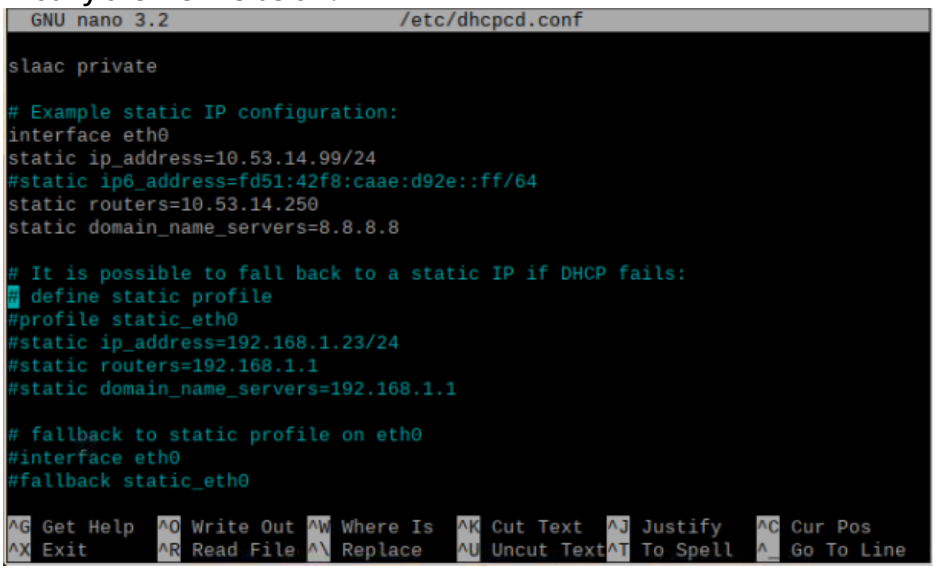
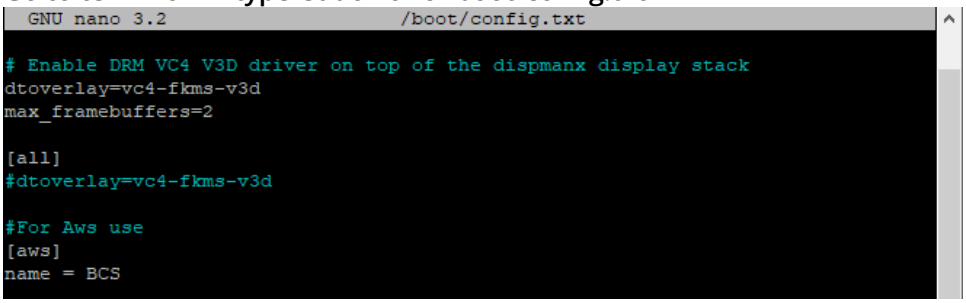
```
pi@raspberrypi:~ $ sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
22/tcp                     ALLOW       Anywhere
21/tcp                     ALLOW       Anywhere
990/tcp                    ALLOW       Anywhere
5500                       ALLOW       Anywhere
5900                       ALLOW       Anywhere
22/tcp (v6)                ALLOW       Anywhere (v6)
21/tcp (v6)                ALLOW       Anywhere (v6)
990/tcp (v6)               ALLOW       Anywhere (v6)
5500 (v6)                  ALLOW       Anywhere (v6)
5900 (v6)                  ALLOW       Anywhere (v6)
```

8. Start firewall and enable to launch during the system boot time
   a. sudo systemctl start ufw
   b. sudo systemctl enable ufw

Remote Desktop (VNC)

1. VNC Server on Raspberry Pi 4 - Default in Raspberry Pi 4
2. VNC Viewer on the computer
   a. Type Raspberry Pi 4's static IP address (10.53.14.99) to connect to the device
   b. Enter username and password (pi, RaspberryPi4)



	Assignment of static IP address for Raspberry Pi 1. check whether DHCPCD is already activated a. sudo service dhcpcd status b. if not, sudo service dhcpcd start => sudo systemctl enable dhcpcd 2. Configure configuration file a. sudo nano /etc/dhcpcd.conf b. Modify the file like below:  3. Reboot Raspberry Pi and check the IP address a. sudo reboot b. ifconfig
Configuration file	**Config.txt** 1. Go to terminal => type 'sudo nano /boot/config.txt'  2. Go to the end of the file, add the last three rows as the figure above a. #For Aws use - comment b. [aws] - tag c. name = XXX - content
Time synchronization	**NTP configuration** NTP installation - sudo apt install ntp 1. Usage of NTP a. service ntp status | start | stop | restart b. Force time synchronization i. Install additional command set - sudo apt install ntpdate ii. Check current delay compared to server - sudo ntpdate -q 0.us.pool.ntp.org iii. Fix delay -

	1. sudo service ntp stop 2. sudo ntpdate 0.us.pool.ntp.org 3. sudo service ntp start
Install required Python libraries	**Used Python Libraries:** 1. GUI - tkinter (default) 2. Serial port in/out - serial (default) **3. Read config file - configparser (pip3 install configparser)** 4. Time management a. Count timer - time (default) b. Fetch current time - datetime (default) 5. Directory management a. Move file to other directories - shutil (default) **b. Object-oriented filesystem paths - pathlib (pip3 install pathlib)** c. Operation of file - os (default) 6. Search substring in long string - re (default)
Autostart python script	**Procedure:** 1. sudo nano /home/pi/.config/lxsession/LXDE-pi/autostart 2. Input commands a. @lxterminal -e python3 /home/pi/DATA/AutoGet.py b. @lxterminal -e python3 /home/pi/DATA/AutoSyncClock.py

(4) Details of Programs

4.1 GUI.py

Libraries used

```
import tkinter as tk
import os
```

Create window and frame

```
# Create window and frame
window = tk.Tk()
window.title('Manual Functions')
window.geometry("400x100")
top_frame = tk.Frame(window)
top_frame.pack()
middle_frame = tk.Frame(window)
middle_frame.pack()
bottom_frame = tk.Frame(window)
bottom_frame.pack()
```

Event Handler for scripts

```
# event handler for 4 python scripts
def get_data():
#     print('get_data')
   os.system('python3 EscDDEnter.py')

def sync_clock():
#     print('sync_clock')
   os.system('python3 EscCCEnter.py')
```

	def reset_field_unit(): # print('reset_field_unit') os.system('python3 EscIREnter.py') def get_backlog(): # print('get_backlog') os.system('python3 EscBHEnter.py') def exit_program(): exit()
Create Buttons	# 4 function keys topleft_button = tk.Button(top_frame, text='Get Data', command=get_data) topleft_button.pack(side=tk.LEFT) topright_button = tk.Button(top_frame, text='Sync Clock', command=sync_clock) topright_button.pack(side=tk.LEFT) middleleft_button = tk.Button(middle_frame, text='Reset Field Unit', command=reset_field_unit) middleleft_button.pack(side=tk.LEFT) middleright_button = tk.Button(middle_frame, text='Get Backlog', command=get_backlog) middleright_button.pack(side=tk.LEFT) # exit button bottom_button = tk.Button(bottom_frame, text='Exit Program', command=exit_program) bottom_button.pack(side=tk.LEFT)
Main Program	# Main Program window.mainloop()

4.2 [Manual] EscIREnter.py

Libraries used	import serial
Defining variables	#Defining the port parameters port = serial.Serial("/dev/ttyUSB0", baudrate=2400, timeout=1) #Variables #String Command get_string = [27, 73, 82, 13]

Reset Field Unit	`def Reset_Field_Unit():` `  #Write Command` `  port.write(bytearray(get_string))` `  print(bytearray(get_string))` `  print('--------------------')` `  port.close()` `  print('Finished')`
Main Function	`#Main Function` `print('Reset Field Unit')` `Reset_Field_Unit()`

4.3 [Manual] EscCCEnter.py

Libraries used	`import serial` `from datetime import date, datetime` `import re`
Defining variables	`#Defining the port parameters` `port = serial.Serial("/dev/ttyUSB0", baudrate=2400, timeout=1)` `#Variables` `#String Command` `get_string = [27, 67, 67]` `final_enter = [13, 10]` `#null variable` `selection = ''`
Sync Clock	`def SyncClock(time):` `  if time != '':` `  #Write Command` `  port.write(bytearray(get_string))` `  port.write(bytearray(time, 'utf-8'))` `  port.write(bytearray(final_enter))` `  print('--------------------')` `  port.close()` `  print('Finished')`
Fetch Time	`def FetchTime(data):` `  time = ''` `  #Get Time - YYMMDDhhmmss` `  match = re.search(r'\d\d\d\d\d\d\d\d\d\d\d\d', data)` `  if match:` `  time = match.group()` `  else:` `  print('Cannot find time')`

	`#Start sync clock` `SyncClock(time)`
Fetch Choice	`def FetchChoice(number):` `choice = ''` `match2 = re.search(r'\d', number)` `if match2:` `choice = match2.group()` `else:` `print('Cannot find choice')` `if choice == '1':` `#Manual input` `print('Sync Clock (Manual input to AWS Field Unit Clock)')` `#input` `print('Enter current time (YYMMDDhhmmss): ')` `time_string = input()` `print('YYMMDDhhmmss: ', time_string)` `FetchTime(time_string)` `elif choice == '2':` `#Server Clock input` `print('Sync Clock (From Server Clock to AWS Field Unit Clock)')` `now = datetime.now()` `time_string = now.strftime("%y%m%d%H%M%S")` `print('YYMMDDhhmmss: ', time_string)` `FetchTime(time_string)` `else:` `print('Wrong input!')` `exit()`
Main function	`#Main Function` `#Choose between using (1) Manual Input / (2) Server Clock` `print('Choose between using (1) Manual Input and (2) Server Clock (Type 1 or 2)')` `number = input()` `FetchChoice(number)`

4.4 [Manual] EscDDEnter.py

Libraries used	`import configparser` `import serial` `import re` `import time` `from pathlib import Path` `from datetime import datetime`

Defining variables

```
#Defining the port parameters
port = serial.Serial("/dev/ttyUSB0", baudrate=2400, timeout=1)

#Variables
#Fetch device name from config file
config = configparser.ConfigParser()
config.read('/boot/config.txt')
station_name = config.get('aws', 'name')

#String Command
get_DD = [27, 68, 68, 13]
#Timestamp variable
now = datetime.now()
month = now.strftime("%b")

#Pathways
current_path = Path('/home/pi/DATA/')
#Change Aws_choice to go to different directories
Aws_choice = 'AWS_manual_data'
Aws_path = ''
station_path = ''
day_path = ''
```

Create Directories

```
def CreateAwsDirectory():
    #Aws
    if not (current_path / Aws_choice).exists():
        Aws_path = current_path / Aws_choice
        Aws_path.mkdir()
    else:
        Aws_path = current_path / Aws_choice

def CreateStationDirectory():
    #Station's short name - Assume BCS is used
    if not (current_path / Aws_choice / station_name).exists():
        station_path = current_path / Aws_choice / station_name
        station_path.mkdir()
    else:
        station_path = current_path / Aws_choice / station_name

def CreateMonthDirectory():
    #Month
    if not (current_path / Aws_choice / station_name / month).exists():
        month_path = current_path / Aws_choice / station_name / month
        month_path.mkdir()
    else:
```

	month_path = current_path / Aws_choice / station_name / month def CreateDayDirectory(): #Month - Jan to Dec now = datetime.now() if not (current_path / Aws_choice / station_name / month / f'{station_name}_{now.strftime("%y%m%d")}').exists(): day_path = current_path / Aws_choice / station_name / month / f'{station_name}_{now.strftime("%y%m%d")}' day_path.mkdir() else: day_path = current_path / Aws_choice / station_name / month / f'{station_name}_{now.strftime("%y%m%d")}'
Read existing data	def ReadData(start): count = 0 while count < 1: #Get existing data port.write(get_DD) read_port_DD = port.read(300).decode('utf-8') #Print read code print('DD: ', read_port_DD) count += 1 CreateFile(read_port_DD) port.close() print('Finished')
Create txt file	def CreateFile(data1): #Filename - Station + YYMMDDHHMM CreateAwsDirectory() CreateStationDirectory() CreateMonthDirectory() CreateDayDirectory() now = datetime.now() current_path = Path('/home/pi/DATA/') / Aws_choice / station_name file_path = current_path / f'{station_name}{now.strftime("%y%m%d%H%M")}.txt' print(f"Directory: ", file_path) with file_path.open('a+') as f: f.write(f'{data1}')
Start of function	#Main Function #Current directory: /home/pi/DATA print('Station Name: ', station_name) print('------------------------------')

```
#Start reading data
ReadData(True)
```

4.5 [Manual] EscBHEnter.py

Libraries used

```
import configparser
import serial
import re
import time
from pathlib import Path
from datetime import datetime
```

Defining variables

```
#Defining the port parameters
port = serial.Serial("/dev/ttyUSB0", baudrate=2400, timeout=1)

#Variables
#Fetch device name from config file
config = configparser.ConfigParser()
config.read('/boot/config.txt')
station_name = config.get('aws', 'name')

#String Command
get_DD = [27, 66, 72, 13]
#Timestamp variable
now = datetime.now()
month = now.strftime("%b")

#Pathways
current_path = Path('/home/pi/DATA/')
#Change Aws_choice to go to different directories
Aws_choice = 'AWS_manual_data'
Aws_path = ''
station_path = ''
day_path = ''
```

Create Directories

```
def CreateAwsDirectory():
   #Aws_anual_data
   if not (current_path / Aws_choice).exists():
       Aws_path = current_path / Aws_choice
       Aws_path.mkdir()
   else:
       Aws_path = current_path / Aws_choice

def CreateStationDirectory():
   #Station's short name - Assume BCS is used
```

```
    if not (current_path / Aws_choice / station_name).exists():
        station_path = current_path / Aws_choice / station_name
        station_path.mkdir()
    else:
        station_path = current_path / Aws_choice / station_name

def CreateMonthDirectory():
    #Month
    if not (current_path / Aws_choice / station_name / month).exists():
        month_path = current_path / Aws_choice / station_name / month
        month_path.mkdir()
    else:
        month_path = current_path / Aws_choice / station_name / month

def CreateDayDirectory():
    #Month - Jan to Dec
    now = datetime.now()
    if not (current_path / Aws_choice / station_name / month /
f'{station_name}_{now.strftime("%y%m%d")}').exists():
        day_path = current_path / Aws_choice / station_name / month /
f'{station_name}_{now.strftime("%y%m%d")}'
        day_path.mkdir()
    else:
        day_path = current_path / Aws_choice / station_name / month /
f'{station_name}_{now.strftime("%y%m%d")}'
```

Read existing data

```
def ReadData(start):
    count = 0
    while count < 1:
        #Get BackLog data
        port.write(get_BH)
        read_port_BH = port.read(300).decode('utf-8')
        #Print read code
        print('BH: ', read_port_BH)
        print('--------------------')
        count += 1

        CreateFile(read_port_BH)

    port.close()
    print('Finished')
```

Create File

```
def CreateFile(data2):
    #Filename - Station + YYMMDDHHMM
    CreateAwsDirectory()
```

	CreateStationDirectory() CreateMonthDirectory() CreateDayDirectory() now = datetime.now() current_path = Path('/home/pi/DATA/') / Aws_choice / station_name file_path = current_path / f'{station_name}{now.strftime("%y%m%d%H%M")}.txt' print(f"Directory: ", file_path) with file_path.open('a+') as f: f.write(f'{data2}')
Main Function	#Main Function #Current directory: /home/pi/DATA print('Station Name: ', station_name) print('------------------------------') #Start reading data ReadData(True)

4.6 [Auto] AutoSyncClock.py

Regular Sync per 3 hour	def RegularSyncClock(time_string): while start == True: now = datetime.now() matches = ['2345', '0245', '0545', '0845', '1145', '1445', '1745'] time_string = now.strftime("%y%m%d%H%M%S") if now.strftime("%H%M") in matches: print('YYMMDDhhmmss: ', time_string) #Write Command port.write(bytearray(get_string)) port.write(bytearray(time_string, 'utf-8')) port.write(bytearray(final_enter)) print('--------------------') print('Finished regular sync') print('--------------------') time.sleep(60) port.close()
Other function	Similar to EscCCEnter.py

4.7 [Auto] AutoGet.py

Libraries used	import configparser import serial import time import re

	`` import shutil import os from pathlib import Path from datetime import date, datetime, timedelta ```
Defining Variables	``` #String Command get_DD = [27, 68, 68, 13] get_BH = [27, 66, 72, 13] ``` **Other variables are similar to EscDDEnter.py and EscBHEnter.py**
Read Data	``` def ReadData(start): while start == True: now = datetime.now() if now.strftime("%S") == '05': print('Current Time: ', now.strftime("%y%m%d%H%M%S")) #Get existing data port.write(get_DD) read_port_DD = port.read(300).decode('utf-8') #Print read code print('DD: ', read_port_DD) #Get BackLog data port.write(get_BH) read_port_BH = port.read(300).decode('utf-8') #Print read code print('BH: ', read_port_BH) print('--------------------') #Create File CreateFile(read_port_DD, read_port_BH) #Housekeeping when 0830 Housekeeping() print('--------------------') port.close() ```
Create File	``` def CreateFile(data1, data2): #Filename - Station + YYMMDDHHMM CreateAwsDirectory() CreateStationDirectory() CreateMonthDirectory() CreateDayDirectory() now = datetime.now() current_path = Path.cwd() / Aws_choice / station_name ```

	file_path = current_path / f'{station_name}{now.strftime("%y%m%d%H%M")}.txt' print(f"Directory: ", file_path) #Append two strings into one .txt file with file_path.open('a+') as f: f.write(f'{data1}{data2}')
Housekeeping	def Housekeeping(): now = datetime.now() yesterday = now - timedelta(days=1) source_path = Path('/home/pi/DATA/') / Aws_choice / station_name destination_path = source_path / month / f'{station_name}_{yesterday.strftime("%y%m%d")}' files = os.listdir(source_path) #Do housekeeping when time goes to 0830 if (now.strftime("%H%M") == '0830'): for filename in files: if yesterday.strftime("%y%m%d") in filename: if filename.endswith('.txt'): shutil.move(f"{source_path}/{filename}", destination_path) print('*****') print('Housekeeping successful!') print('*****')
Main Function	#Main Function #Current directory: /home/pi/DATA print('Station Name: ', station_name) print('------------------------------') #Start reading data ReadData(True)

4.8 [Auto] AutoGet_non_polling.py

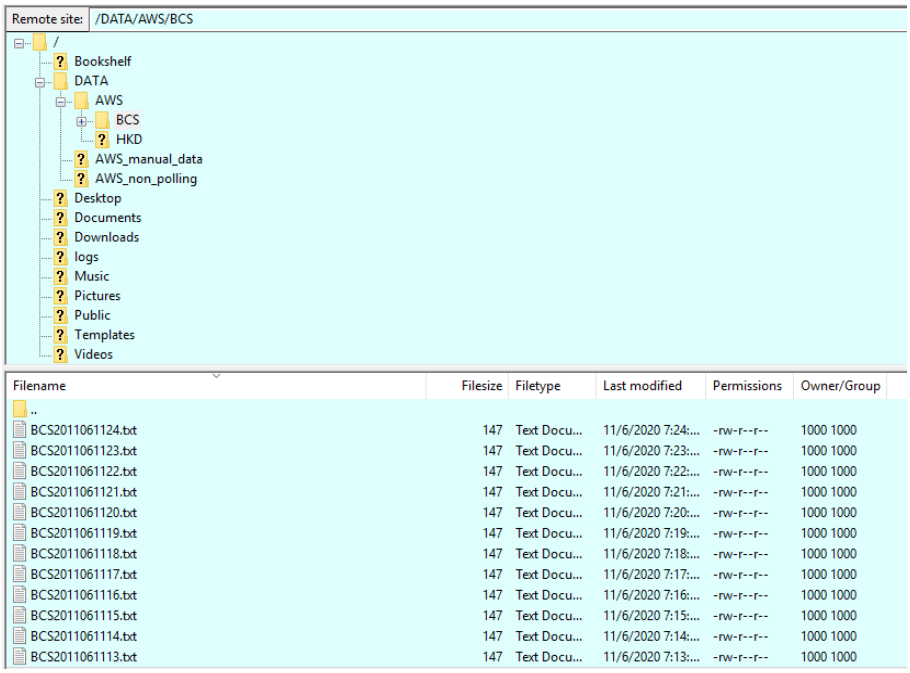
Read Data	def ReadData(start): while start == True: #Get existing data read_port = port.read(300).decode('utf-8') #Create File if read_port != '': CreateFile(read_port) #Housekeeping when 0830 Housekeeping() port.close()

Other function	Similar to AutoGet.py

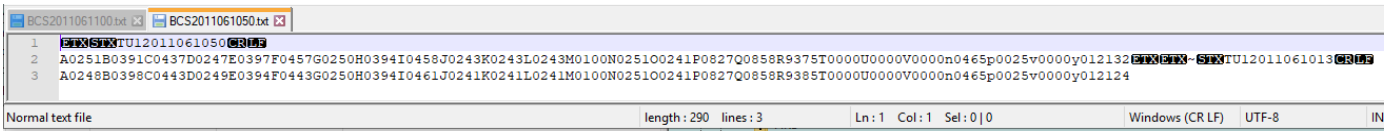
(5) Result Capture Screen and Capability for Future Expansion

5.1 Result Capture Screen

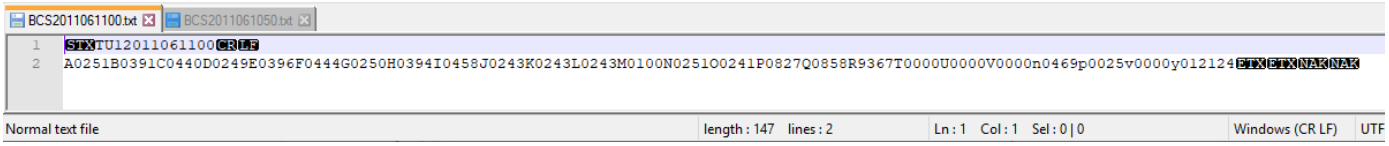
5.1.1 Filezilla Screen (through SFTP and FTP)



5.1.2 .txt file by AutoGet.py (with Present data + Backlog data)



5.1.3 .txt file by AutoGet.py (with Present data only)



5.1.4 Information shown when python script is running

5.1.5 Log shown on polling screen

5.1.6 Time synchronization (Immediate + Regular sync)

```
pi@raspberrypi:~ $ cd DATA
pi@raspberrypi:~/DATA $ python3 AutoSyncClock.py
Sync Clock (From Server Clock to AWS Field Unit Clock)
YYMMDDhhmmss:  201106101434
--------------------
Finished immediate sync
--------------------
YYMMDDhhmmss:  201106114500
--------------------
Finished regular sync
--------------------
```

5.2 Capability for Future Expansion

5.2.1 Stability of GUI.py

At this moment, GUI.py is based on forming the graphic interface by tkinter library and opening other python scripts by os library. However, there is an error when EscCCEnter.py's "manual input" function is used, then time of polling server is changed but GUI.py is crashed. The reason behind is that the compatibility of tkinter library cannot match with Python's default input function. One practical solution

is that the input field can be built-in inside a graphical interface, and users do not need to type time by command-line interface. In this case, error may be prevented and stability of GUI.py can be ensured.

5.2.2 4G modem

Throughout this project, we have tried two ways of network connection - LAN and WAN. For LAN, we used an Ethernet Switch to connect between HKO computer and Raspberry Pi 4, therefore two static IP addresses are used - HKO computer is 10.53.14.44 and Raspberry Pi 4 is 10.53.14.99. For WAN, Raspberry Pi 4 was connected to a router set LAN IP address as 192.168.8.1 (1010 4G network), then a virtual IP address 192.168.8.131 was assigned to Raspberry Pi 4. After that, any device connecting to the 4G network can access Raspberry Pi 4 by WAN IP address 14.0.133.3 through FTP and SFTP protocols. However, the 2 testing scenarios were just located in an indoor environment, therefore on-site testing should be tried, since real-time environment is an important factor of receiving 4G signals. Also, the 4G modem used on site is not the same as HuaWei 4G modem inside HKO, also Raspberry Pi's performance may be suffered from high temperature, therefore the actual performance may be totally different.

(6) Reference

6.1 pathlib Library
- https://medium.com/@ageitgey/python-3-quick-tip-the-easy-way-to-deal-with-file-paths-on-windows-mac-and-linux-11a072b58d5f
- http://zetcode.com/python/pathlib/
- http://www.ityouknow.com/python/2019/10/19/python-pathlib-035.html

6.2 Regular Expressions
- https://developers.google.com/edu/python/regular-expressions

6.3 FTP and SFTP Server
- http://yhhuang1966.blogspot.com/2017/02/ftp.html
- https://www.raspberrypi-spy.co.uk/2018/05/creating-ftp-server-with-raspberry-pi/
- https://linuxize.com/post/how-to-setup-ftp-server-with-vsftpd-on-raspberry-pi/

6.4 Firewall Settings
- https://www.layerstack.com/resources/tutorials/How-to-set-up-configure-secure-vsFTPd-on-Linux-Cloud-Servers
- https://www.raspberrypi.org/documentation/configuration/security.md
- https://askubuntu.com/questions/161346/how-to-configure-ufw-to-allow-ip-forwarding
- https://www.peterdavehello.org/2016/01/ubuntu-based-gnulinux-firewall-ufw-essential-config/

6.5 Configuration File Settings
- https://medium.com/better-programming/tips-and-tricks-for-handling-configuration-files-in-python-a9d7429aa50b
- https://www.raspberrypi.org/documentation/configuration/config-txt/

6.6 Static IP address
- https://www.ionos.com/digitalguide/server/configuration/provide-raspberry-pi-with-a-static-ip-address/

6.7 Time synchronization
- https://raspberrytips.com/time-sync-raspberry-pi/

6.8 Move file (shutil Library)
- https://stackoverflow.com/questions/61620485/python-if-filename-in-specified-path-contains-string-then-move-to-folder

6.9 TCP and UDP Port Number
- https://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers

6.10 Autostart running script when reboot
- https://stackoverflow.com/questions/41913956/how-to-automatically-launch-python-file-once-gui-has-loaded-on-raspbian-pixel