**EMSD EG Project - Report**

**Sin Bo-chi (INFEG2002)**

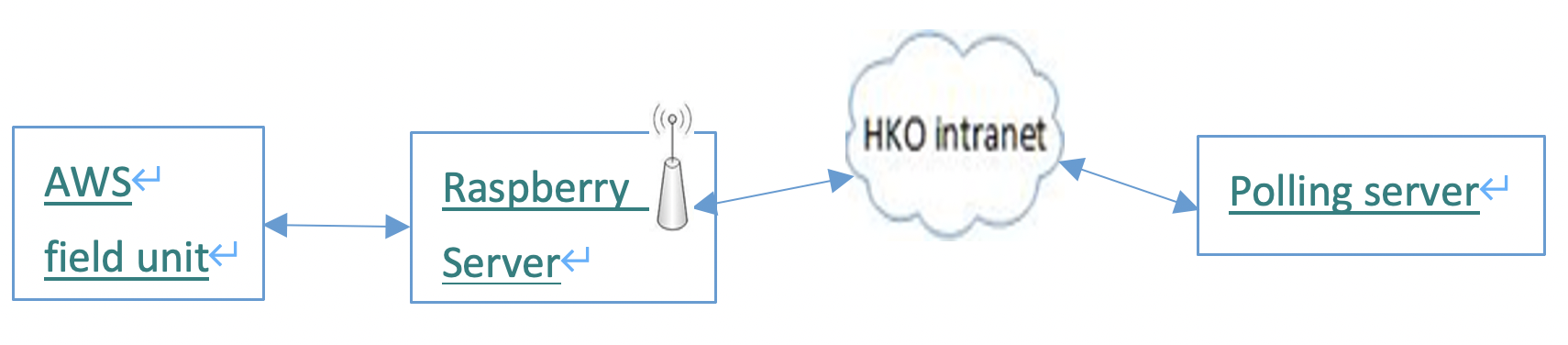
Dates: 19 Oct 2020 (Mon) - 13 Nov 2020 (Fri) [4 weeks]

Mentors: 1) Mr. YEUNG Chi Yu 2926 8043 [cyyeung@hko.gov.hk](mailto:cyyeung@hko.gov.hk)

2) Mr. LAI Wai Lun 2926 8044 [wllai@hko.gov.hk](mailto: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) :

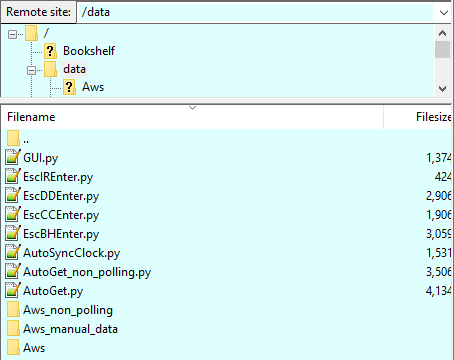
|  |  |  |
| --- | --- | --- |
| 1 | Implement a program to Sync Clock through the USB to Com port device on every 3 hour / manual input | ✔️ |
| 2 | Implement a program to receive AWS field unit from Special Com port device with no polling command | ✔️ |
| 3 | Implement a program to reset AWS field 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**

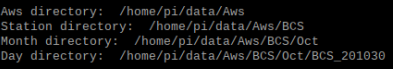
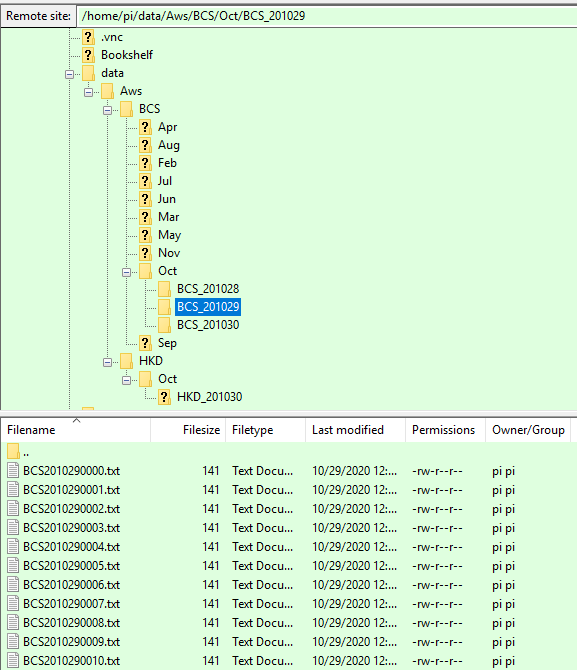
The whole system can be described in three aspects: (1) File structure, (2) Programs.

2.1 File structure

1. 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):   1. Housekeeping function |
| **AutoSyncClock.py** -   1. **ESC CC ↵ > Sync Clock**     1. From Raspberry Pi Server Clock to AWS Field Unit Clock    2. Immediate sync clock **by once**    3. 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**     1. From Manual Input to AWS Field Unit Clock    2. 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    1. **EscDDEnter.py**    2. **EscBHEnter.py**    3. **EscCCEnter.py**    4. **EscIREnter.py** |

**(3) Operation Procedures and Configurations**

3.1 Operation Procedures

|  |  |
| --- | --- |
| SFTP access through PC | 1. Enable VNC interface on Raspberry Pi    1. Menu -> Preferences -> Raspberry Pi Configuration      1. admin (User: Pi, Password: RaspberryPi4)    1. Have all access right in Raspberry Pi      1. user (User: ftpuser, Password: raspberry)    1. Have only read right in Raspberry Pi      1. LAN: 10.53.14.99 2. 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   1. Run python scripts by the terminal (same as automation)   2. Polling Program (GUI.py)      1. open a new terminal to run python script (python3 GUI.py)      2. Click on.the buttons and run python scripts |
| Enable Remote Desktop (VNC) | 1. Enable VNC interface on Raspberry Pi    1. Menu -> Preferences -> Raspberry Pi Configuration      * 1. Check IP address through terminal command - ifconfig      * 1. Check existing IP address through VNC server      1. Download and install VNC Viewer on a PC computer    1. Download link: <https://www.realvnc.com/en/connect/download/viewer/>    2. Make sure the PC computer and Raspberry Pi server are in the same network    3. Enter Raspberry Pi’s IP address, admin account and password    4. Successfully for using remote desktop |

3.2 Configuration

|  |  |
| --- | --- |
| Network Configuration | **SFTP**  Server Side:   1. Open SSH inside Raspberry Pi    1. Go to terminal -> type ‘sudo raspi-config’    2. Choose 5. Interfacing Options’    3. Choose P2 SSH      * 1. Choose YES to enable SSH server      1. vsftpd (management of SFTP server)    1. Go to terminal -> type ‘sudo apt-get install vsftpd’    2. Update configuration file       1. Modify vsftpd configuration file -> type ‘sudo nano /etc/vsftpd.conf’       2. 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/       3. Save and exit using CTRL-X, Y and ENTER    3. Create FTP directory and set permission right for user       1. mkdir /home/pi/DATA => chmod a-w /home/pi/DATA       2. (optional) if new user need to get data from Aws directory          1. chmod 755 /home/pi/DATA       3. Final permission should be like this:      * 1. Add new user (can only download files, cannot upload and modify files)      1. sudo useradd newuser      2. sudo passwd newuser      3. Enter new UNIX password: raspberry      4. Retype new UNIX password: raspberry      5. passwd: password updated successfully   2. Modify new user’s default directory      1. SFTP         1. sudo usermod -d /home/pi/DATA newuser         2. Everytime login to new user account by Putty / Filezilla by SFTP            1. => default directory is /home/pi/DATA      2. FTP         1. sudo nano /etc/vsftpd.conf         2. Add the following lines            1. allow\_writeable\_chroot=YES            2. local\_root=/home/pi   3. Restart FTP server      1. sudo service vsftpd restart   4. (optional) Improving SSH security (username/password security)      1. 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](https://filezilla-project.org/download.php?type=client)) 2. Go to site manager addr)      * 1. Set two connection presets with SFTP (host = Raspberry Pi’s IP address)      1. Admin user - Have all of the access rights      2. 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    1. sudo ufw default allow outgoing    2. 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:      1. Start firewall and enable to launch during the system boot time    1. sudo systemctl start ufw    2. 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    1. Type Raspberry Pi 4’s static IP address (10.53.14.99) to connect to the device    2. Enter username and password (pi, RaspberryPi4) |
| **Assignment of static IP address for Raspberry Pi**   1. check whether DHCPCD is already activated    1. sudo service dhcpcd status    2. if not, sudo service dhcpcd start => sudo systemctl enable dhcpcd 2. Configure configuration file    1. sudo nano /etc/dhcpcd.conf    2. Modify the file like below:      1. Reboot Raspberry Pi and check the IP address    1. sudo reboot    2. 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    1. #For Aws use - comment    2. [aws] - tag    3. name = XXX - content |
| Time synchronization | **NTP configuration**  NTP installation - sudo apt install ntp   1. Usage of NTP    1. service ntp status | start | stop | restart    2. Force time synchronization       1. Install additional command set - sudo apt install ntpdate       2. Check current delay compared to server - sudo ntpdate -q 0.us.pool.ntp.org       3. 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    1. Count timer - time (default)    2. Fetch current time - datetime (default) 5. Directory management    1. Move file to other directories - shutil (default)    2. **Object-oriented filesystem paths - pathlib (pip3 install pathlib)**    3. 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    1. @lxterminal -e python3 /home/pi/DATA/AutoGet.py    2. @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) |

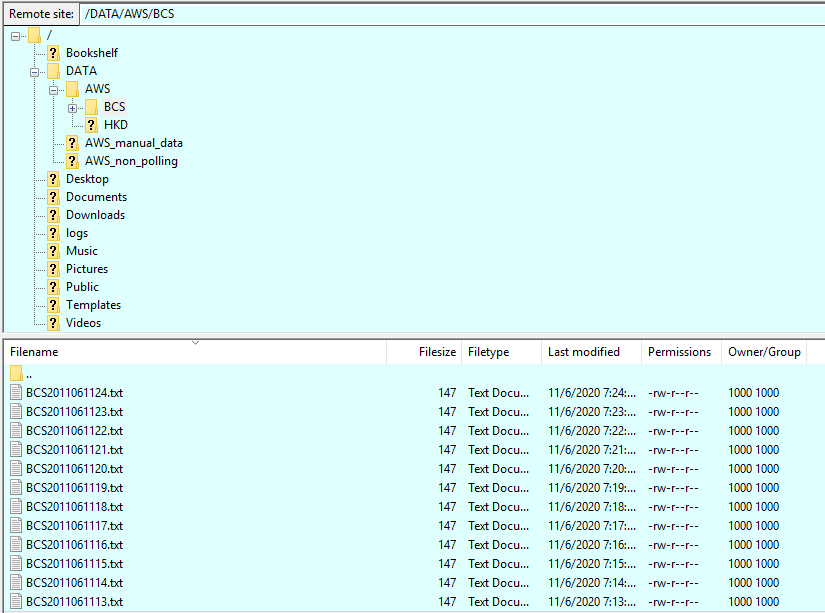
4.8 [Auto] AutoGet\_non\_polling.py

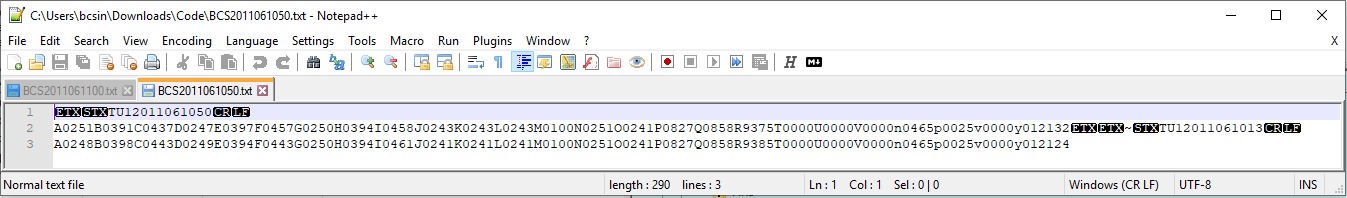
|  |  |
| --- | --- |
| Read Data | def ReadData(start):  while start == True:  #Get existing data  read\_port = port.read(300).decode('utf-8')  #Create File  if read\_port != '':  CreateFile(read\_port)  #Housekeeping when 0830  Housekeeping()  port.close() |
| Other function | Similar to AutoGet.py |

**(5) Result Capture Screen and Capability for Future Expansion**

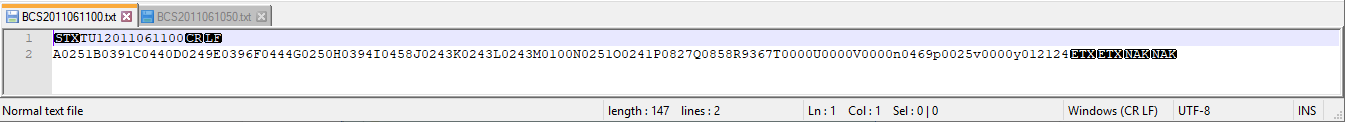
5.1 Result Capture Screen

5.1.1 Filezilla Screen (through SFTP and FTP)

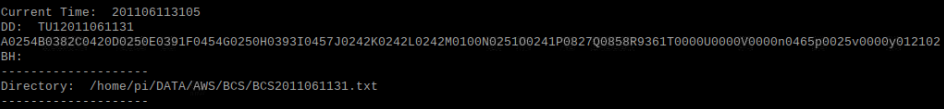


5.1.2 .txt file by AutoGet.py (with Present data + Backlog data)

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



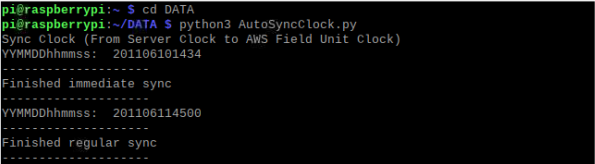
5.1.4 Information shown when python script is running



5.1.5 Log shown on polling screen



5.1.6 Time synchronization (Immediate + Regular sync)



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>