IoT Lab Group 18

XU Huafeng 19092472g
HUANG Weihai 19090418g
Ding Yuepu 19087453g
NG SAI KIT 15008568g

- ## LABORATORY 1
  - ### Exercise 1 :

1. Run the phase one IoT demo with a method tp vary the temperature data and humidity data.

2. Export the data collected in Exercise 1.

3. Demonstrate your results to the tutor and submit your findings in the lab report.

1.1.Temperature:

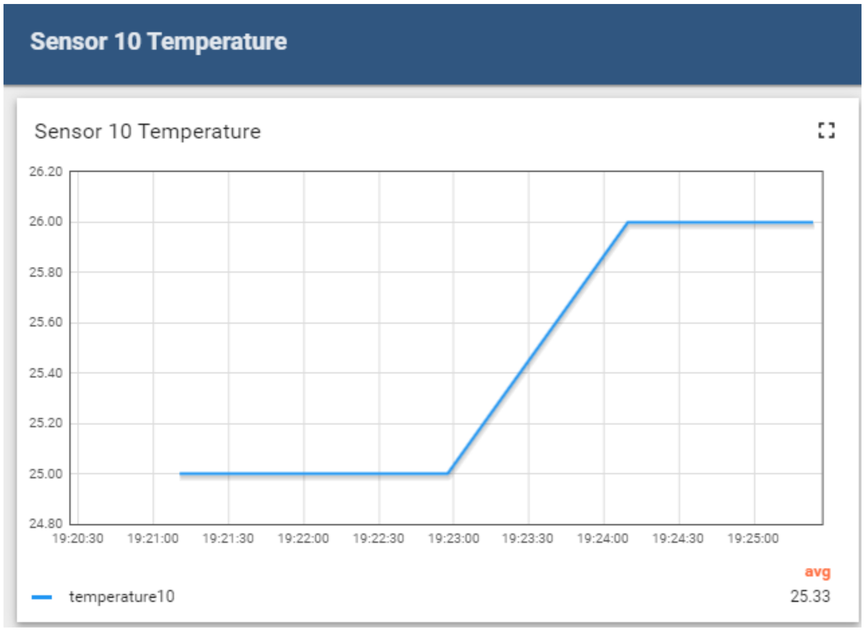


Fig1_1.Temperature

26	10/25/2019	7:25:23 PM
26	10/25/2019	7:24:09 PM
25	10/25/2019	7:22:57 PM
25	10/25/2019	7:22:18 PM
25	10/25/2019	7:21:42 PM
25	10/25/2019	7:21:10 PM

Table1_1. Temperature

1.2.Humidity

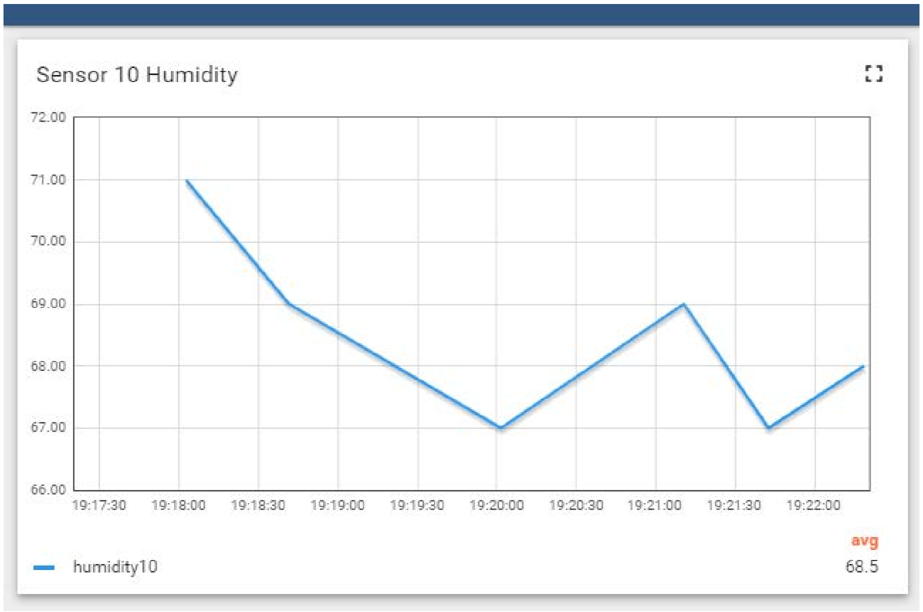


Fig1_2.Humidity

68	10/25/2019	7:22:18 PM
67	10/25/2019	7:21:42 PM
69	10/25/2019	7:21:10 PM
67	10/25/2019	7:20:01 PM
69	10/25/2019	7:18:41 PM
71	10/25/2019	7:18:02 PM

Table1_2. Humidity

- **Exercise 2:**

1. Run the phase two IoT demo by changing the button status and changing the voltage.
2. Change the voltage and export the voltage data collected from the server.
3. Observe the LED brightness and pattern.
4. Change the button status and export the sensor data collected from the server.
5. Observe the LED lighting pattern and summarize the change in the transmission rate via pressing the button.
6. Demonstrate your results to the tutors and submit your findings in the lab report.

2.1&2.Sensor Voltage

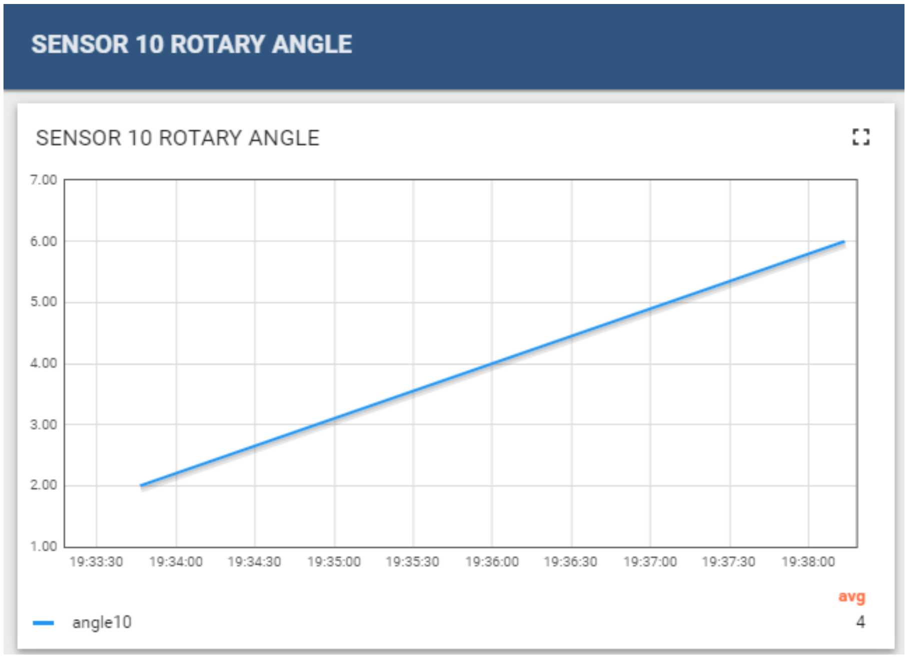


Fig2_1.Sensor Voltage

6	10/25/2019	7:38:13 PM
2	10/25/2019	7:33:46 PM

Table 2_1

2.3.The brightness of the LED

- Weak brightness for counterclockwise voltage control
- Strong brightness for clockwise voltage control

2.4.Change the button status and export the sensor data collected from the server

25	10/25/2019	7:50:30 PM
25	10/25/2019	7:47:52 PM
26	10/25/2019	7:42:32 PM
26	10/25/2019	7:40:37 PM
26	10/25/2019	7:40:03 PM
26	10/25/2019	7:39:24 PM
26	10/25/2019	7:38:45 PM
26	10/25/2019	7:38:13 PM
26	10/25/2019	7:33:46 PM
26	10/25/2019	7:31:17 PM
26	10/25/2019	7:29:37 PM

Table2_1. sensor data

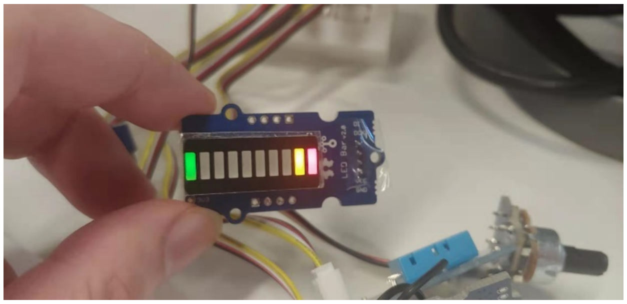


Fig2_2

Then, original setting => 30 seconds,

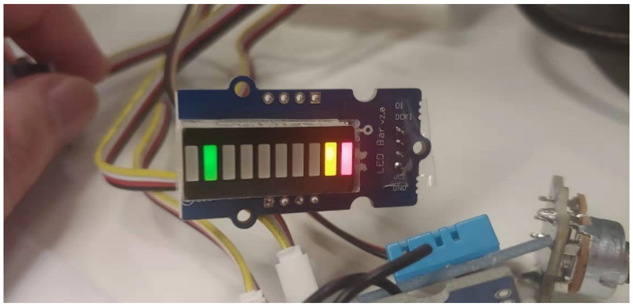


Fig2_3

Then, turn to 2nd led by button, change to over 2minutes… (sometimes missed transmission => longer time)

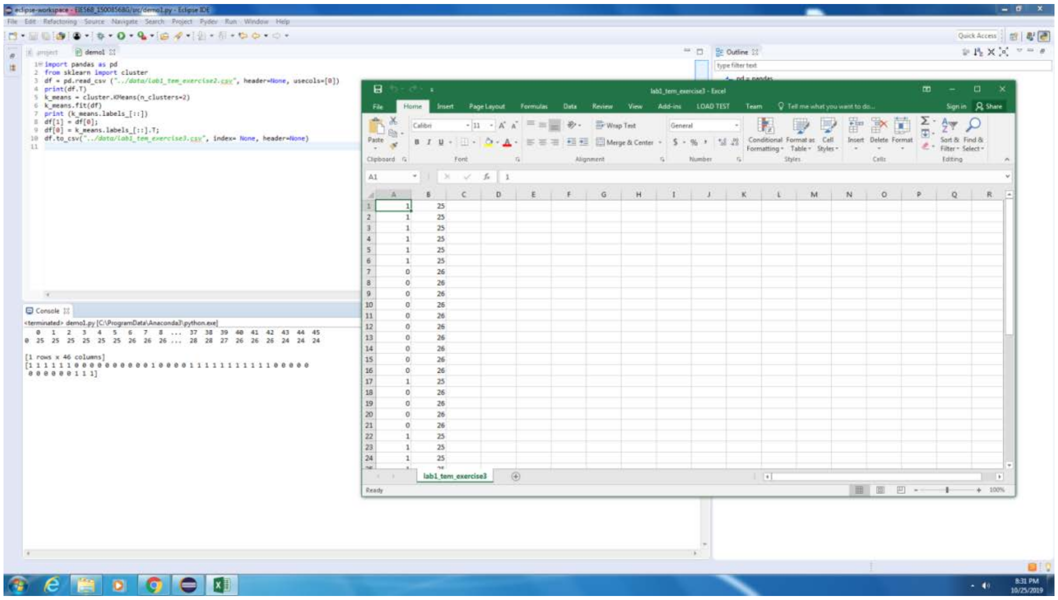
- # LABORATORY 2a
  - ## EXERCISE :

1. Change the input with data from other sensors and run the program to identify the unsupervised learning results.

2. Generate the result files.

3. Demonstrate your results to the tutor and submit your findings in the lab report.

Firstly,when we make CLUSTER = 2

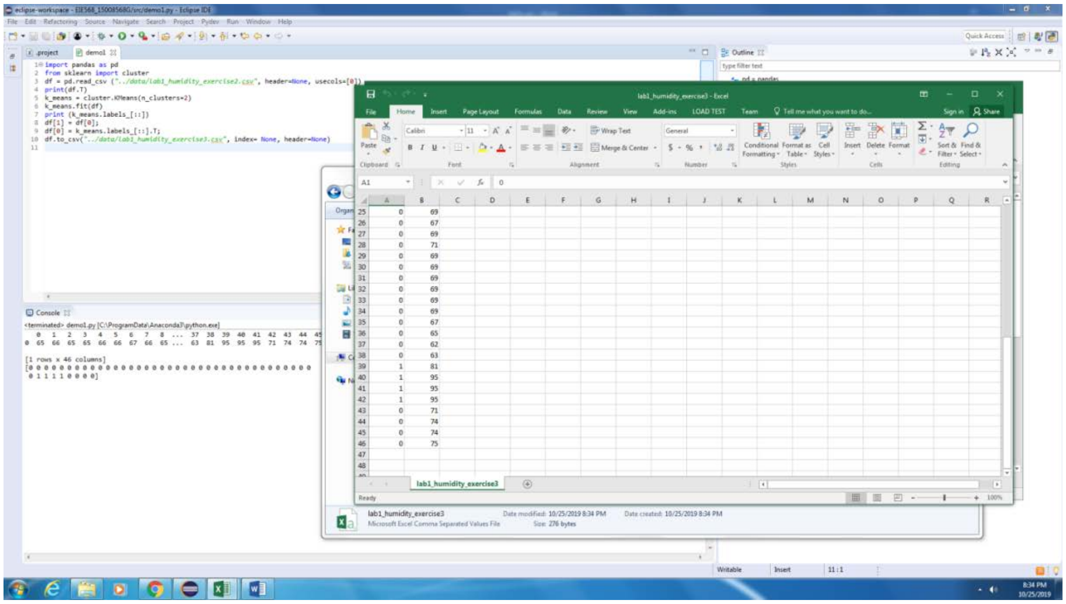


```
import pandas as pd
from sklearn import cluster
df = pd.read_csv ("../data/lab1_tem_exercise2.csv", header=None, usecols=[0])
print(df.T)
k_means = cluster.KMeans(n_clusters=2)
k_means.fit(df)
print (k_means.labels_[::])
df[1] = df[0];
df[0] = k_means.labels_[::].T;
df.to_csv("../data/lab1_tem_exercise3.csv", index= None, header=None)
```

```
<terminated> demo1.py [C:\ProgramData\Anaconda3\python.exe]
   0   1   2   3   4   5   6   7   8  ...  37  38  39  40  41  42  43  44  45
0  25  25  25  25  25  25  26  26  26 ...  28  28  27  26  26  26  24  24  24

[1 rows x 46 columns]
[1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0
 0 0 0 0 0 0 1 1 1]
```

lab1_tem_exercise3

	A	B
1	1	25
2	1	25
3	1	25
4	1	25
5	1	25
6	1	25
7	0	26
8	0	26
9	0	26
10	0	26
11	0	26
12	0	26
13	0	26
14	0	26
15	0	26
16	0	26
17	1	25
18	0	26
19	0	26
20	0	26
21	0	26
22	1	25
23	1	25
24	1	25



```
import pandas as pd
from sklearn import cluster
df = pd.read_csv ("../data/lab1_humidity_exercise2.csv", header=None, usecols=[0])
print(df.T)
k_means = cluster.KMeans(n_clusters=2)
k_means.fit(df)
print (k_means.labels_[::])
df[1] = df[0];
df[0] = k_means.labels_[::].T;
df.to_csv("../data/lab1_humidity_exercise3.csv", index= None, header=None)
```

```
<terminated> demo1.py [C:\ProgramData\Anaconda3\python.exe]
   0   1   2   3   4   5   6   7   8  ...  37  38  39  40  41  42  43  44  45
0  65  66  65  65  66  66  67  66  65 ...  63  81  95  95  95  71  74  74  75

[1 rows x 46 columns]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 1 1 1 1 0 0 0 0]
```

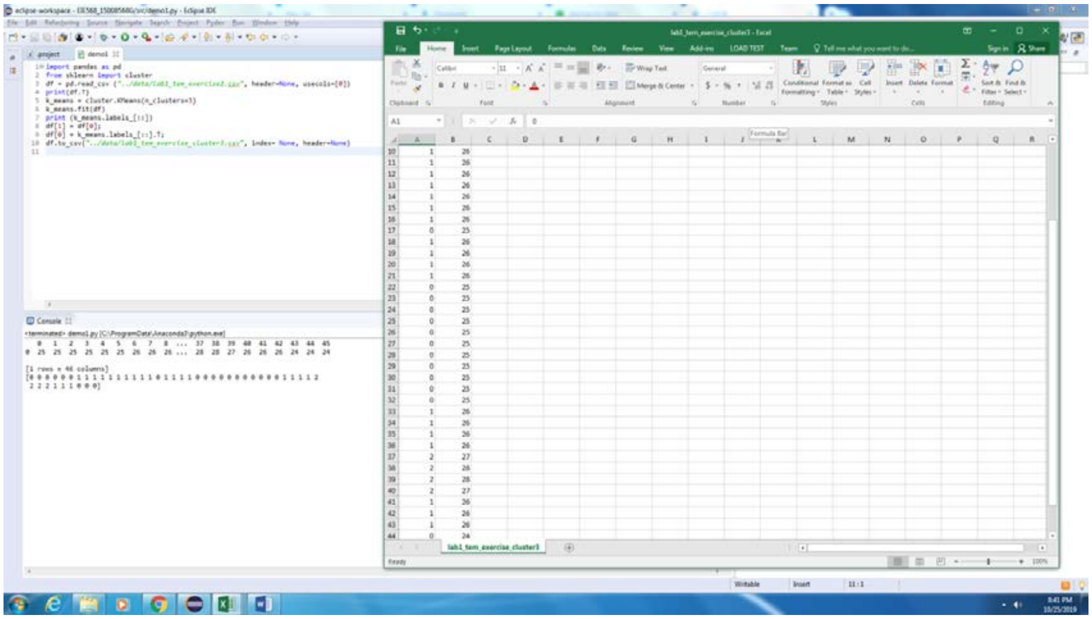
lab1_humidity_exercise3

	A	B
25	0	69
26	0	67
27	0	69
28	0	71
29	0	69
30	0	69
31	0	69
32	0	69
33	0	69
34	0	69
35	0	67
36	0	65
37	0	62
38	0	63
39	1	81
40	1	95
41	1	95
42	1	95
43	0	71
44	0	74
45	0	74
46	0	75

lab1_humidity_exercise3 Date modified: 10/25/2019 8:34 PM Date created: 10/25/2019 8:34 PM
Microsoft Excel Comma Separated Values File Size: 276 bytes

Fig3_1. cluster=2

Then, we make CLUSTER = 3



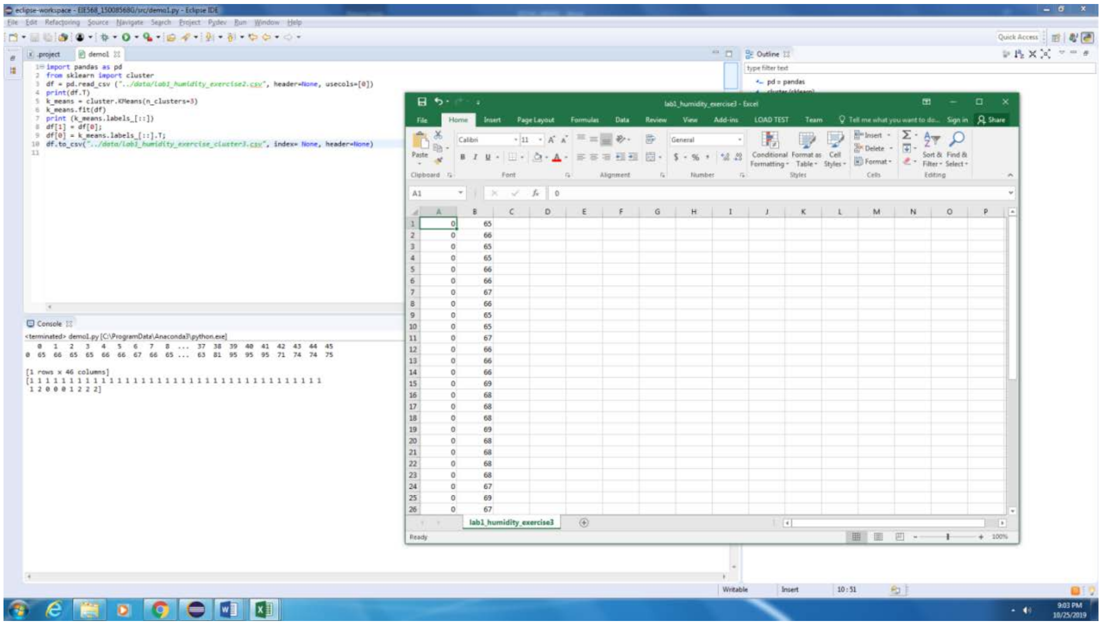
eclipse-workspace - EIE568_15008568G/src/demo1.py - Eclipse IDE

```python
import pandas as pd
from sklearn import cluster
df = pd.read_csv ("../data/lab1_tem_exercise2.csv", header=None, usecols=[0])
print(df.T)
k_means = cluster.KMeans(n_clusters=3)
k_means.fit(df)
print (k_means.labels_[::])
df[1] = df[0];
df[0] = k_means.labels_[::].T;
df.to_csv("../data/lab1_tem_exercise_cluster3.csv", index= None, header=None)
```

```
<terminated> demo1.py [C:\ProgramData\Anaconda3\python.exe]
    0   1   2   3   4   5   6   7   8  ...  37  38  39  40  41  42  43  44  45
0  25  25  25  25  25  25  26  26  26  ...  28  28  27  26  26  26  24  24  24

[1 rows x 46 columns]
[0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 2
 2 2 2 1 1 1 0 0 0]
```

lab1_tem_exercise_cluster3 - Excel

	A	B
10	1	26
11	1	26
12	1	26
13	1	26
14	1	26
15	1	26
16	1	26
17	0	25
18	1	26
19	1	26
20	1	26
21	1	26
22	0	25
23	0	25
24	0	25
25	0	25
26	0	25
27	0	25
28	0	25
29	0	25
30	0	25
31	0	25
32	0	25
33	1	26
34	1	26
35	1	26
36	1	26
37	2	27
38	2	28
39	2	28
40	2	27
41	1	26
42	1	26
43	1	26
44	0	24



eclipse-workspace - EIE568_15008568G/src/demo1.py - Eclipse IDE

```python
import pandas as pd
from sklearn import cluster
df = pd.read_csv ("../data/lab1_humidity_exercise2.csv", header=None, usecols=[0])
print(df.T)
k_means = cluster.KMeans(n_clusters=3)
k_means.fit(df)
print (k_means.labels_[::])
df[1] = df[0];
df[0] = k_means.labels_[::].T;
df.to_csv("../data/lab1_humidity_exercise_cluster3.csv", index= None, header=None)
```

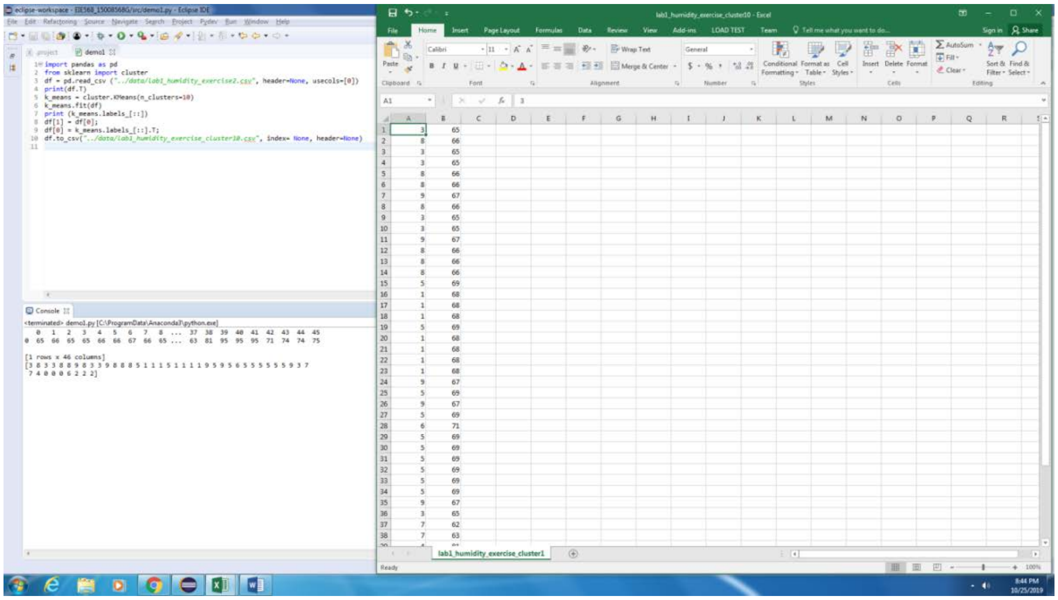
```
<terminated> demo1.py [C:\ProgramData\Anaconda3\python.exe]
    0   1   2   3   4   5   6   7   8  ...  37  38  39  40  41  42  43  44  45
0  65  66  65  65  66  66  67  66  65  ...  63  81  95  95  95  71  74  74  75

[1 rows x 46 columns]
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 2 0 0 0 1 2 2 2]
```

lab1_humidity_exercise3 - Excel

	A	B
1	0	65
2	0	66
3	0	65
4	0	65
5	0	66
6	0	66
7	0	67
8	0	66
9	0	65
10	0	65
11	0	67
12	0	66
13	0	66
14	0	66
15	0	69
16	0	68
17	0	68
18	0	68
19	0	69
20	0	68
21	0	68
22	0	68
23	0	68
24	0	67
25	0	69
26	0	67

Fig3_2. cluster=3

If we make CLUSTER = 10


eclipse-workspace - EIE568_15008568G/src/demo1.py - Eclipse IDE
demo1
lab1_humidity_exercise_cluster10 - Excel
lab1_humidity_exercise_cluster1
A1
Ready

```
import pandas as pd
from sklearn import cluster
df = pd.read_csv ("../data/lab1_humidity_exercise2.csv", header=None, usecols=[0])
print(df.T)
k_means = cluster.KMeans(n_clusters=10)
k_means.fit(df)
print (k_means.labels_[::])
df[1] = df[0];
df[0] = k_means.labels_[::].T;
df.to_csv("../data/lab1_humidity_exercise_cluster10.csv", index= None, header=None)
```

```
<terminated> demo1.py [C:\ProgramData\Anaconda3\python.exe]
   0   1   2   3   4   5   6   7   8 ...  37  38  39  40  41  42  43  44  45
0  65  66  65  65  66  66  67  66  65 ...  63  81  95  95  95  71  74  74  75

[1 rows x 46 columns]
[3 8 3 3 8 8 9 8 3 3 9 8 8 8 5 1 1 1 5 1 1 1 1 9 5 9 5 6 5 5 5 5 5 5 9 3 7
 7 4 0 0 0 6 2 2 2]
```

	A	B
1	3	65
2	8	66
3	3	65
4	3	65
5	8	66
6	8	66
7	9	67
8	8	66
9	3	65
10	3	65
11	9	67
12	8	66
13	8	66
14	8	66
15	5	69
16	1	68
17	1	68
18	1	68
19	5	69
20	1	68
21	1	68
22	1	68
23	1	68
24	9	67
25	5	69
26	9	67
27	5	69
28	6	71
29	5	69
30	5	69
31	5	69
32	5	69
33	5	69
34	5	69
35	9	67
36	3	65
37	7	62
38	7	63

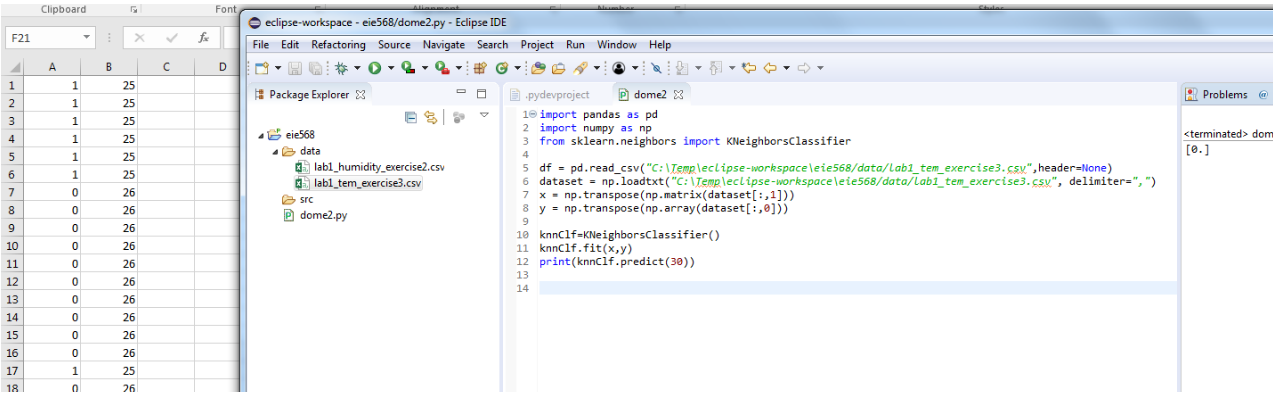
Fig3_3. cluster=10

From these exercises, we learn to use various components to build a simple IoT system to collecting temperature, humidity, and voltage data. Then we learn how to processing the data collected by simple programming language . Working with classmates together and with the help of teachers, we deeply know the IoT system and unsupervised learning.

- # Laboratory 2b
    - ## Exercise 1

1. **Change the input test data and run the program to obtain the supervised learning result.**

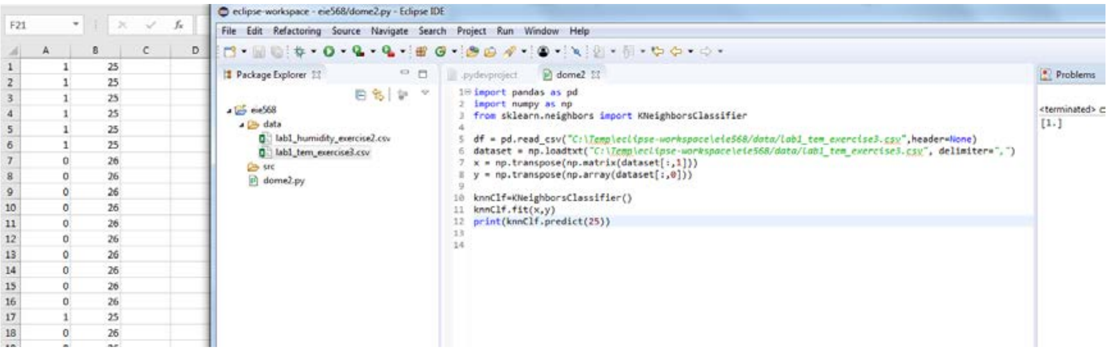
In the first try, we use the Temperature data set as input. When we set the predict value to **30**, the output is **0**. When we set the predict value to **25**, the output is **1**. In addition, when our predict value is 25.5, the output is also 1. Because in our dataset, the data below 26 is classified as 1 and the data above 26 is classified as 0. This is a bit different from our lab example. Because the KNeighborsClassifier's output is decided by its input dataset. The lab result is show below:



	A	B	C	D
1	1	25		
2	1	25		
3	1	25		
4	1	25		
5	1	25		
6	1	25		
7	0	26		
8	0	26		
9	0	26		
10	0	26		
11	0	26		
12	0	26		
13	0	26		
14	0	26		
15	0	26		
16	0	26		
17	1	25		
18	0	26		

```
import pandas as pd
import numpy as np
from sklearn.neighbors import KNeighborsClassifier

df = pd.read_csv("C:\Temp\eclipse-workspace\eie568/data/lab1_tem_exercise3.csv",header=None)
dataset = np.loadtxt("C:\Temp\eclipse-workspace\eie568/data/lab1_tem_exercise3.csv", delimiter=",")
x = np.transpose(np.matrix(dataset[:,1]))
y = np.transpose(np.array(dataset[:,0]))

knnClf=KNeighborsClassifier()
knnClf.fit(x,y)
print(knnClf.predict(30))
```

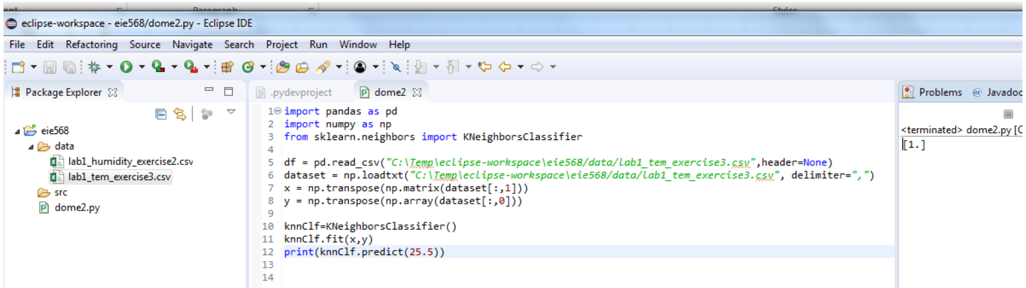
```
<terminated> dom
[0.]
```



	A	B	C	D
1	1	25		
2	1	25		
3	1	25		
4	1	25		
5	1	25		
6	1	25		
7	0	26		
8	0	26		
9	0	26		
10	0	26		
11	0	26		
12	0	26		
13	0	26		
14	0	26		
15	0	26		
16	0	26		
17	1	25		
18	0	26		

```
import pandas as pd
import numpy as np
from sklearn.neighbors import KNeighborsClassifier

df = pd.read_csv("C:\Temp\eclipse-workspace\eie568/data/lab1_tem_exercise3.csv",header=None)
dataset = np.loadtxt("C:\Temp\eclipse-workspace\eie568/data/lab1_tem_exercise3.csv", delimiter=",")
x = np.transpose(np.matrix(dataset[:,1]))
y = np.transpose(np.array(dataset[:,0]))

knnClf=KNeighborsClassifier()
knnClf.fit(x,y)
print(knnClf.predict(25))
```

```
<terminated> c
[1.]
```

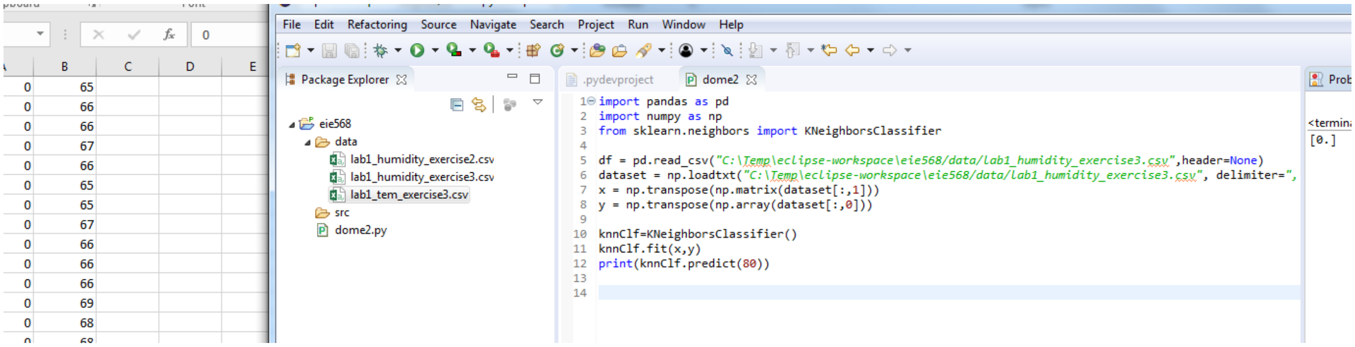


```
import pandas as pd
import numpy as np
from sklearn.neighbors import KNeighborsClassifier

df = pd.read_csv("C:\Temp\eclipse-workspace\eie568/data/lab1_tem_exercise3.csv",header=None)
dataset = np.loadtxt("C:\Temp\eclipse-workspace\eie568/data/lab1_tem_exercise3.csv", delimiter=",")
x = np.transpose(np.matrix(dataset[:,1]))
y = np.transpose(np.array(dataset[:,0]))

knnClf=KNeighborsClassifier()
knnClf.fit(x,y)
print(knnClf.predict(25.5))
```

```
<terminated> dome2.py [C
[1.]
```

Fig2b_1. Temperature in KNeighborsClassifier

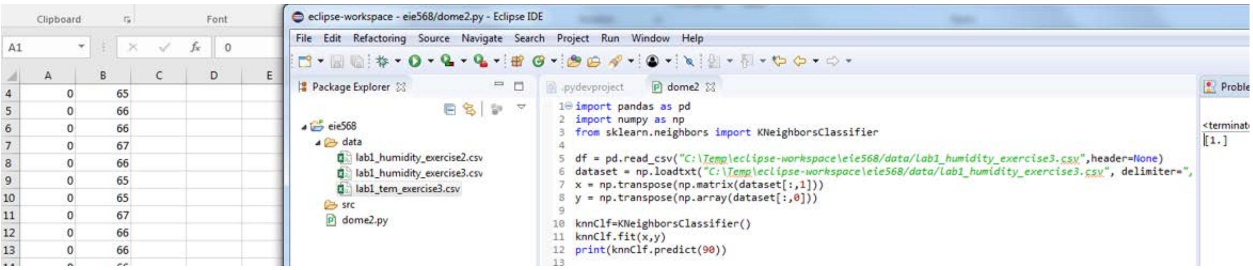
The we change the input data set from the Humidity data. When we input predict value as 80, the output is 0. When we input predict value as 90, the output is 1. the result is showed as follow:



	B	C	D	E
0	65			
0	66			
0	66			
0	67			
0	66			
0	65			
0	65			
0	67			
0	66			
0	66			
0	66			
0	69			
0	68			

```
import pandas as pd
import numpy as np
from sklearn.neighbors import KNeighborsClassifier

df = pd.read_csv("C:\Temp\eclipse-workspace\eie568/data/lab1_humidity_exercise3.csv",header=None)
dataset = np.loadtxt("C:\Temp\eclipse-workspace\eie568/data/lab1_humidity_exercise3.csv", delimiter=",
x = np.transpose(np.matrix(dataset[:,1]))
y = np.transpose(np.array(dataset[:,0]))

knnClf=KNeighborsClassifier()
knnClf.fit(x,y)
print(knnClf.predict(80))
```

```
<termin
[0.]
```



	A	B	C	D	E
4	0	65			
5	0	66			
6	0	66			
7	0	67			
8	0	66			
9	0	65			
10	0	65			
11	0	67			
12	0	66			
13	0	66			

```
import pandas as pd
import numpy as np
from sklearn.neighbors import KNeighborsClassifier

df = pd.read_csv("C:\Temp\eclipse-workspace\eie568/data/lab1_humidity_exercise3.csv",header=None)
dataset = np.loadtxt("C:\Temp\eclipse-workspace\eie568/data/lab1_humidity_exercise3.csv", delimiter=",
x = np.transpose(np.matrix(dataset[:,1]))
y = np.transpose(np.array(dataset[:,0]))

knnClf=KNeighborsClassifier()
knnClf.fit(x,y)
print(knnClf.predict(90))
```

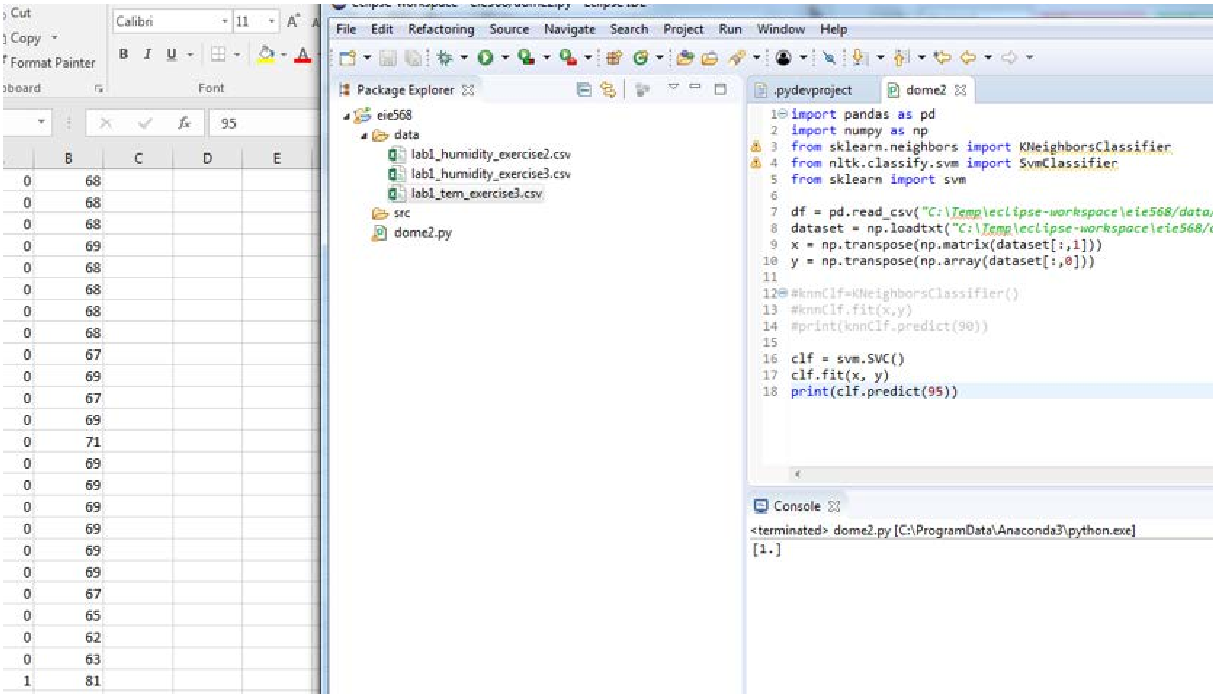
```
<terminat
[1.]
```

Fig2b_2. Humidity in KNeighborsClassifier

2. Change the supervised learning method (for example, SVM) and run the program to obtain the supervised learning result.

We change the classified method as **SVM**. In machine learning, support-vector machines (SVMs, also support-vector networks) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new example to one category or the other, making it a non-probabilistic binary linear classifier (although methods such as Platt scaling exist to use SVM in a probabilistic classification setting).

We also use Temperature data and Humidity data as our training dataset. The predict output data is just similar as the KNeighborsClassifier. The Output results are shown as followings:

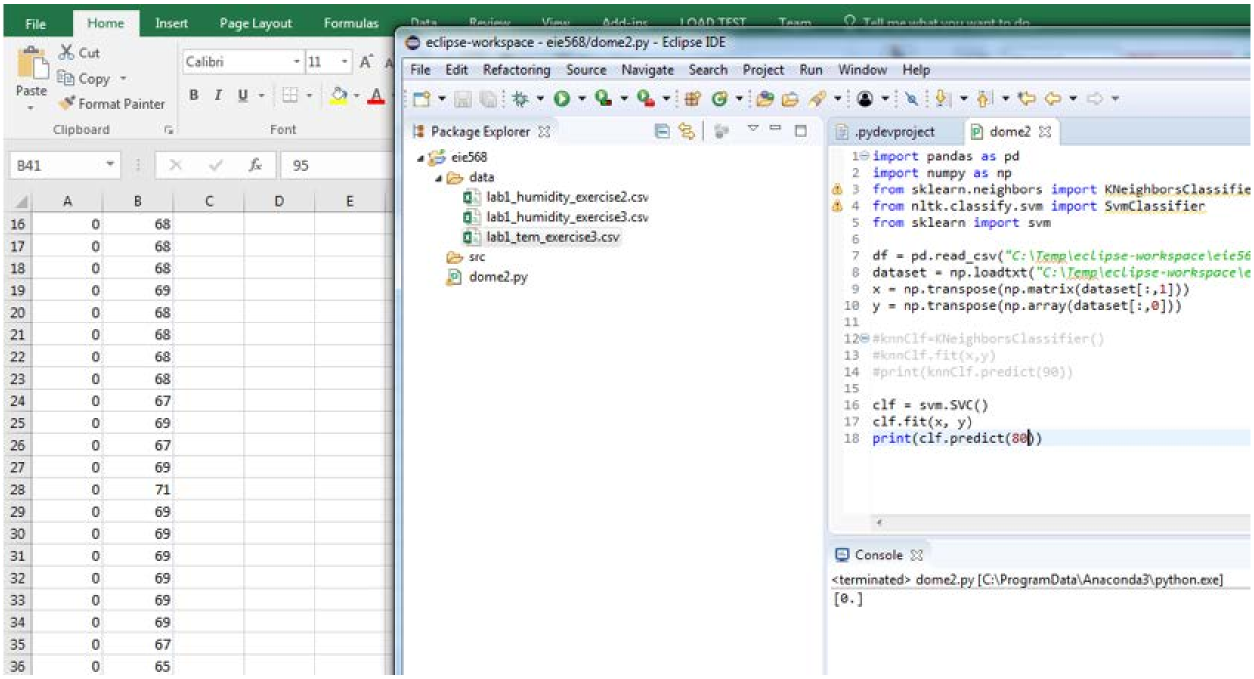


95

	B	C	D	E
0	68			
0	68			
0	68			
0	69			
0	68			
0	68			
0	68			
0	68			
0	67			
0	69			
0	67			
0	69			
0	71			
0	69			
0	69			
0	69			
0	69			
0	69			
0	69			
0	67			
0	65			
0	62			
0	63			
1	81			

```
import pandas as pd
import numpy as np
from sklearn.neighbors import KNeighborsClassifier
from nltk.classify.svm import SvmClassifier
from sklearn import svm

df = pd.read_csv("C:\Temp\eclipse-workspace\eie568/data/
dataset = np.loadtxt("C:\Temp\eclipse-workspace\eie568/(
x = np.transpose(np.matrix(dataset[:,1]))
y = np.transpose(np.array(dataset[:,0]))

#knnClf=KNeighborsClassifier()
#knnClf.fit(x,y)
#print(knnClf.predict(90))

clf = svm.SVC()
clf.fit(x, y)
print(clf.predict(95))
```

```
<terminated> dome2.py [C:\ProgramData\Anaconda3\python.exe]
[1.]
```



File
Home
Insert
Page Layout
Formulas
Paste
Cut
Copy
Format Painter
Clipboard
Calibri
11
Font
eclipse-workspace - eie568/dome2.py - Eclipse IDE
File Edit Refactoring Source Navigate Search Project Run Window Help
Package Explorer
eie568
data
lab1_humidity_exercise2.csv
lab1_humidity_exercise3.csv
lab1_tem_exercise3.csv
src
dome2.py
.pydevproject
dome2
Console

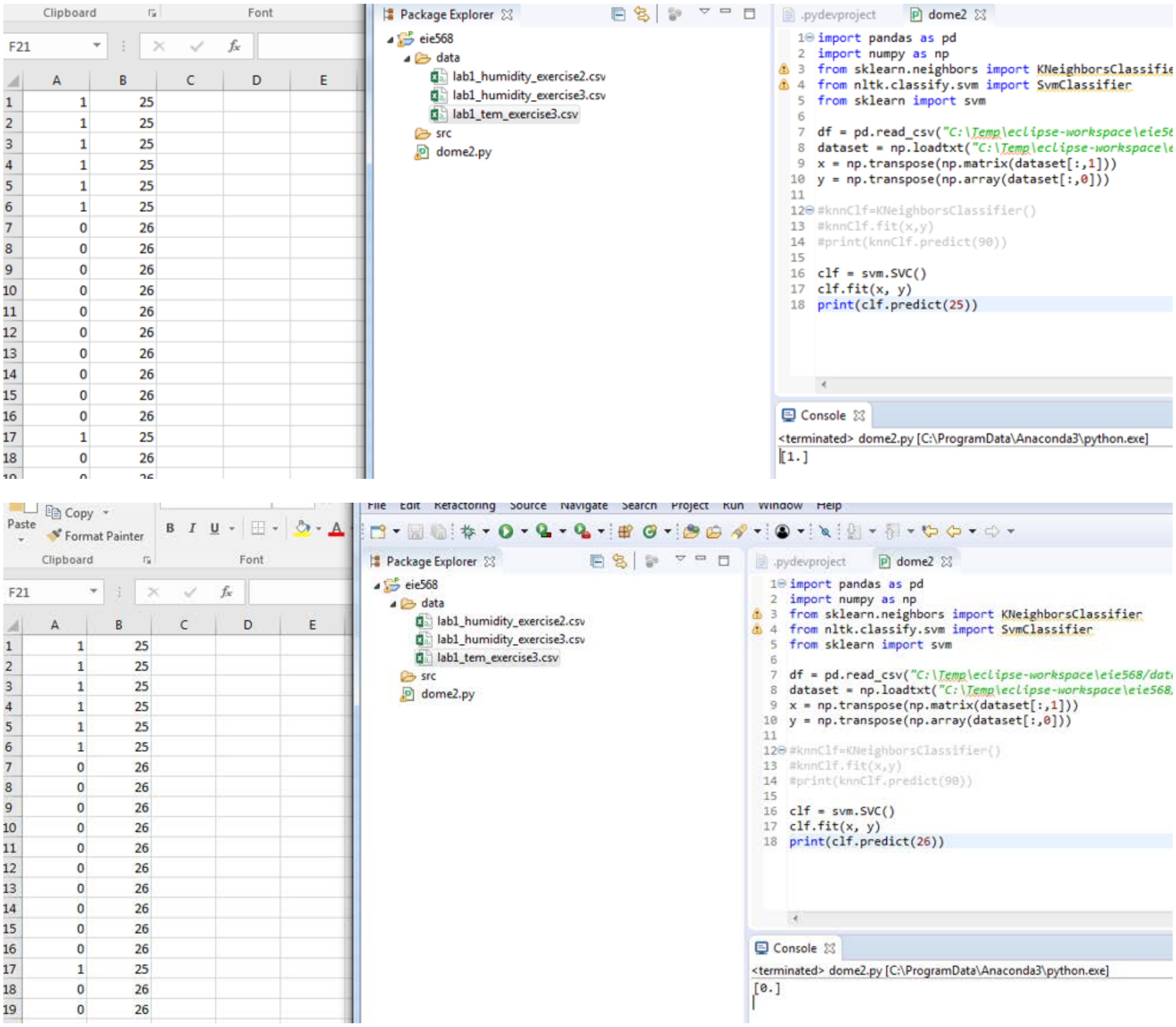
B41 95

	A	B	C	D	E
16	0	68			
17	0	68			
18	0	68			
19	0	69			
20	0	68			
21	0	68			
22	0	68			
23	0	68			
24	0	67			
25	0	69			
26	0	67			
27	0	69			
28	0	71			
29	0	69			
30	0	69			
31	0	69			
32	0	69			
33	0	69			
34	0	69			
35	0	67			
36	0	65			

```
import pandas as pd
import numpy as np
from sklearn.neighbors import KNeighborsClassifie
from nltk.classify.svm import SvmClassifier
from sklearn import svm

df = pd.read_csv("C:\Temp\eclipse-workspace\eie56
dataset = np.loadtxt("C:\Temp\eclipse-workspace\e
x = np.transpose(np.matrix(dataset[:,1]))
y = np.transpose(np.array(dataset[:,0]))

#knnClf=KNeighborsClassifier()
#knnClf.fit(x,y)
#print(knnClf.predict(90))

clf = svm.SVC()
clf.fit(x, y)
print(clf.predict(80))
```

```
<terminated> dome2.py [C:\ProgramData\Anaconda3\python.exe]
[0.]
```

Fig2b_3. Humidity in SVM



	A	B	C	D	E
1	1	25			
2	1	25			
3	1	25			
4	1	25			
5	1	25			
6	1	25			
7	0	26			
8	0	26			
9	0	26			
10	0	26			
11	0	26			
12	0	26			
13	0	26			
14	0	26			
15	0	26			
16	0	26			
17	1	25			
18	0	26			

```
import pandas as pd
import numpy as np
from sklearn.neighbors import KNeighborsClassifie
from nltk.classify.svm import SvmClassifier
from sklearn import svm

df = pd.read_csv("C:\Temp\eclipse-workspace\eie56
dataset = np.loadtxt("C:\Temp\eclipse-workspace\e
x = np.transpose(np.matrix(dataset[:,1]))
y = np.transpose(np.array(dataset[:,0]))

#knnClf=KNeighborsClassifier()
#knnClf.fit(x,y)
#print(knnClf.predict(90))

clf = svm.SVC()
clf.fit(x, y)
print(clf.predict(25))
```

```
<terminated> dome2.py [C:\ProgramData\Anaconda3\python.exe]
[1.]
```

	A	B	C	D	E
1	1	25			
2	1	25			
3	1	25			
4	1	25			
5	1	25			
6	1	25			
7	0	26			
8	0	26			
9	0	26			
10	0	26			
11	0	26			
12	0	26			
13	0	26			
14	0	26			
15	0	26			
16	0	26			
17	1	25			
18	0	26			
19	0	26			

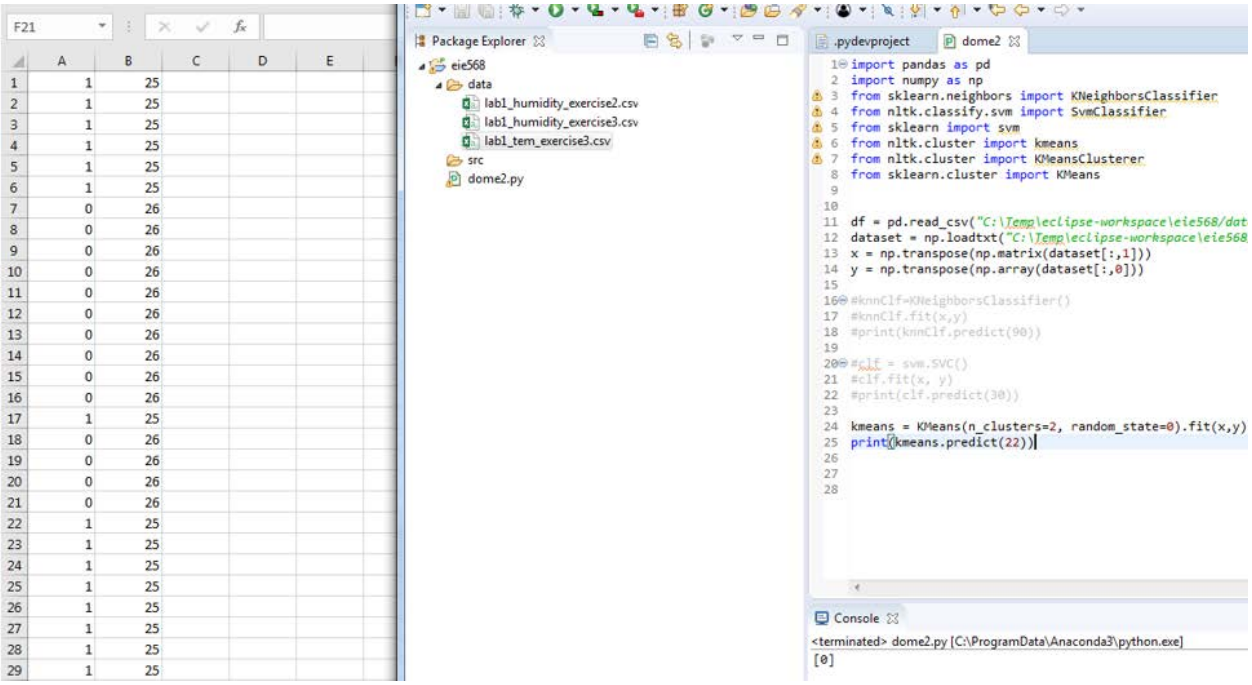
```
import pandas as pd
import numpy as np
from sklearn.neighbors import KNeighborsClassifier
from nltk.classify.svm import SvmClassifier
from sklearn import svm

df = pd.read_csv("C:\Temp\eclipse-workspace\eie568/dat
dataset = np.loadtxt("C:\Temp\eclipse-workspace\eie568,
x = np.transpose(np.matrix(dataset[:,1]))
y = np.transpose(np.array(dataset[:,0]))

#knnClf=KNeighborsClassifier()
#knnClf.fit(x,y)
#print(knnClf.predict(90))

clf = svm.SVC()
clf.fit(x, y)
print(clf.predict(26))
```

```
<terminated> dome2.py [C:\ProgramData\Anaconda3\python.exe]
[0.]
```

Fig2b_4. Temperature in SVM

In addition, we use K-means Classifier to classify our dataset. K-means clustering aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster. This results in a partitioning of the data space into Voronoi cells. As for our dataset, we choose 2 cluster for K-Means method. For Temperature Dataset, when the input data is 22, the output data is 0. When the input data is 30, the output data is 1. From the result, we can see that the output is different from SVM and K-Neighbors.

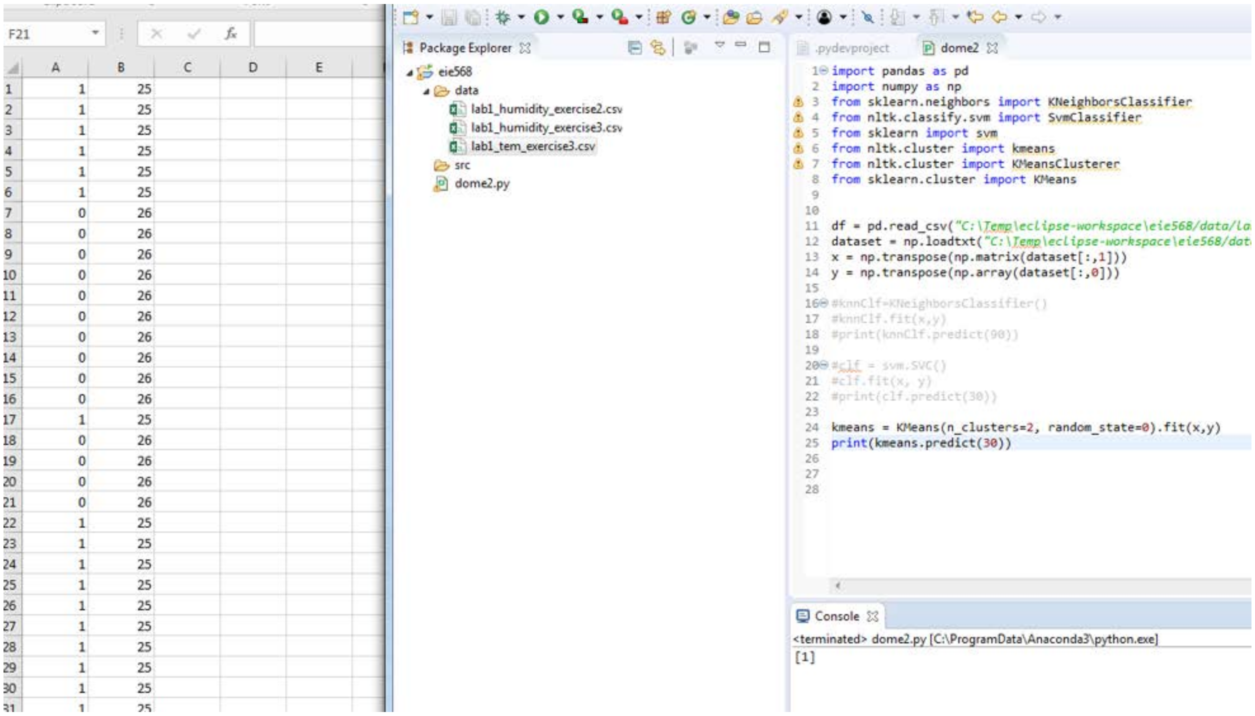
As conclusion, different learning method will have a bit different in the result, and we should choice the best suitable learning method for the application.



	A	B	C	D	E
1	1	25			
2	1	25			
3	1	25			
4	1	25			
5	1	25			
6	1	25			
7	0	26			
8	0	26			
9	0	26			
10	0	26			
11	0	26			
12	0	26			
13	0	26			
14	0	26			
15	0	26			
16	0	26			
17	1	25			
18	0	26			
19	0	26			
20	0	26			
21	0	26			
22	1	25			
23	1	25			
24	1	25			
25	1	25			
26	1	25			
27	1	25			
28	1	25			
29	1	25			

```python
import pandas as pd
import numpy as np
from sklearn.neighbors import KNeighborsClassifier
from nltk.classify.svm import SvmClassifier
from sklearn import svm
from nltk.cluster import kmeans
from nltk.cluster import KMeansClusterer
from sklearn.cluster import KMeans


df = pd.read_csv("C:\Temp\eclipse-workspace\eie568/dat
dataset = np.loadtxt("C:\Temp\eclipse-workspace\eie568
x = np.transpose(np.matrix(dataset[:,1]))
y = np.transpose(np.array(dataset[:,0]))

#knnClf=KNeighborsClassifier()
#knnClf.fit(x,y)
#print(knnClf.predict(90))

#clf = svm.SVC()
#clf.fit(x, y)
#print(clf.predict(30))

kmeans = KMeans(n_clusters=2, random_state=0).fit(x,y)
print(kmeans.predict(22))
```

```
<terminated> dome2.py [C:\ProgramData\Anaconda3\python.exe]
[0]
```



	A	B	C	D	E
1	1	25			
2	1	25			
3	1	25			
4	1	25			
5	1	25			
6	1	25			
7	0	26			
8	0	26			
9	0	26			
10	0	26			
11	0	26			
12	0	26			
13	0	26			
14	0	26			
15	0	26			
16	0	26			
17	1	25			
18	0	26			
19	0	26			
20	0	26			
21	0	26			
22	1	25			
23	1	25			
24	1	25			
25	1	25			
26	1	25			
27	1	25			
28	1	25			
29	1	25			
30	1	25			
31	1	25			

```python
import pandas as pd
import numpy as np
from sklearn.neighbors import KNeighborsClassifier
from nltk.classify.svm import SvmClassifier
from sklearn import svm
from nltk.cluster import kmeans
from nltk.cluster import KMeansClusterer
from sklearn.cluster import KMeans


df = pd.read_csv("C:\Temp\eclipse-workspace\eie568/data/la
dataset = np.loadtxt("C:\Temp\eclipse-workspace\eie568/dat
x = np.transpose(np.matrix(dataset[:,1]))
y = np.transpose(np.array(dataset[:,0]))

#knnClf=KNeighborsClassifier()
#knnClf.fit(x,y)
#print(knnClf.predict(90))

#clf = svm.SVC()
#clf.fit(x, y)
#print(clf.predict(30))

kmeans = KMeans(n_clusters=2, random_state=0).fit(x,y)
print(kmeans.predict(30))
```

```
<terminated> dome2.py [C:\ProgramData\Anaconda3\python.exe]
[1]
```

Fig2b_5. Temperature in K-means

- **Task 3. To design an IoT system application using the sensors in the demo system**

  We design a system to detect the laboratory's Voltage, Temperature and Humidity.

  The **voltage, temperature and humidity** management is very important for most laboratories. The stability of voltage, temperature and humidity is sufficient to ensure the stability and safety of the experiment. In the general chemical laboratory, the temperature is very important for the chemical reaction. Sometime it will affect the testing, and you cannot get the result you want, and even worse, the temperature change of some chemicals can cause explosion and deterioration. In other situation, the stability of voltage is very important. If the voltage change quickly, we must take immediately action to handle it. Otherwise, it will affect the testing data collection. The monitoring of temperature and humidity in biological laboratories is also very important. Cells, tissues and organisms all are sensitive to temperature and humidity are very susceptible to irreversible damage caused by temperature and humidity changes.

  It is very important to establish a complete temperature and humidity monitoring system and use a temperature and humidity monitoring system. Many laboratories do not pay enough attention to this, and management is relatively loose, which cause a lot of dangers.

  Actually, it is very easy to measure and analysis the lab's voltage, temperature and humidity with IOT technology nowadays. We are able to design a very simple and cheap **IOT system** with **Arduino** or others mini processor to detect and using machine learning to analysis those data. Using **LoRa or Sigfox** to transmit the collected data, and using machine learning to analysis it. When the temperature or humidity or voltage is beyond the normal level. The system will remind user to address it.

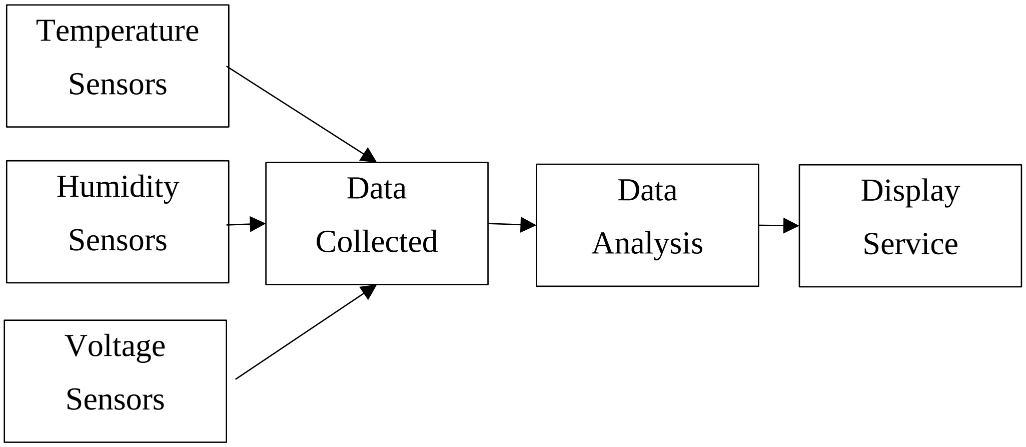


Fig2b_6. IOT system for Laboratory Voltage, Temperature and Humidity

- **Task 4. To analyze the designed IoT application data**
  1. **To illustrate the findings in the analysis based on unsupervised learning or supervised learning methods.**

The data used in our lab is under **supervised learning**. All the collected data has the label. Moreover, a function (model parameter) is learnt from a given training data set, and when new data arrives, the result can be predicted based on this function. The training set used for supervised learning include input and output, and which can assign specific usage for the output data. The meaning of the training set is assigned by people. In our lab, we label the temperature for 0 or 1 and the humidity data just as the similar.

Supervised learning is the most common problem of classification (attention and clustering). An existing model is trained to obtain an optimal model (this model belongs to a set of functions, Optimal representation of the best under a certain evaluation criterion), and then use this model to map all the inputs to the corresponding output, and make a simple judgment on the output to achieve the purpose of classification. It also has the ability to classify unknown data. The goal of supervised learning is often to let computers learn the classification systems (models) we have created. Supervised learning is a common technique for training neural networks and decision trees. These two techniques are highly dependent on information given by a predetermined classification system. For neural networks, the classification system uses information to determine network errors and then constantly adjusts network parameters. For decision trees, the classification system uses it to determine which attributes provide the most information.

Common supervised learning algorithms: regression analysis and statistical classification. The most typical algorithms are KNN and SVM. It is also the method we use in our lab.

- **Laboratory 3a**

  Step 1: Install Telegram on our phone

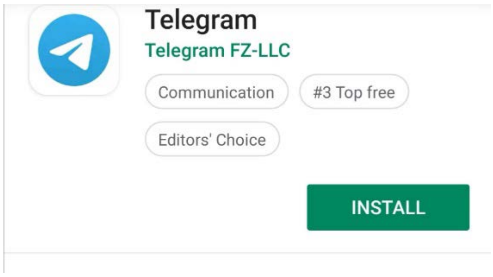
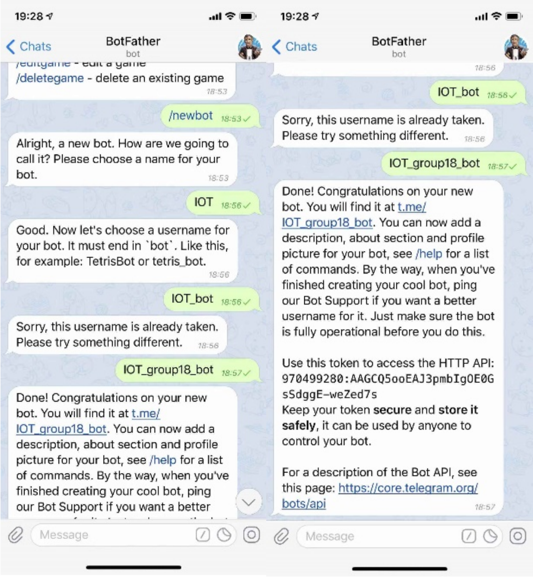


Fig3a_1. Install Telegram

Step 2: Creating a Telegram BOT and Getting our token for access. Our bot in Telegram is **IOT_group18_bot**.



/deletegame - delete an existing game 18:53

/newbot 18:53

Alright, a new bot. How are we going to call it? Please choose a name for your bot. 18:53

IOT 18:56

Good. Now let's choose a username for your bot. It must end in `bot`. Like this, for example: TetrisBot or tetris_bot. 18:56

IOT_bot 18:56

Sorry, this username is already taken. Please try something different. 18:56

IOT_group18_bot 18:57

Done! Congratulations on your new bot. You will find it at t.me/IOT_group18_bot. You can now add a description, about section and profile picture for your bot, see /help for a list of commands. By the way, when you've finished creating your cool bot, ping our Bot Support if you want a better

IOT_bot 18:56

Sorry, this username is already taken. Please try something different. 18:56

IOT_group18_bot 18:57

Done! Congratulations on your new bot. You will find it at t.me/IOT_group18_bot. You can now add a description, about section and profile picture for your bot, see /help for a list of commands. By the way, when you've finished creating your cool bot, ping our Bot Support if you want a better username for it. Just make sure the bot is fully operational before you do this.

Use this token to access the HTTP API:
970499280:AAGCQ5ooEAJ3pmbIgOE0GsSdggE-weZed7s
Keep your token **secure** and **store it safely**, it can be used by anyone to control your bot.

For a description of the Bot API, see this page: https://core.telegram.org/bots/api 18:57

Fig3a_2. Telegram on our phone

Step 3: Telepot for installing Telegram on Raspberry Pi:

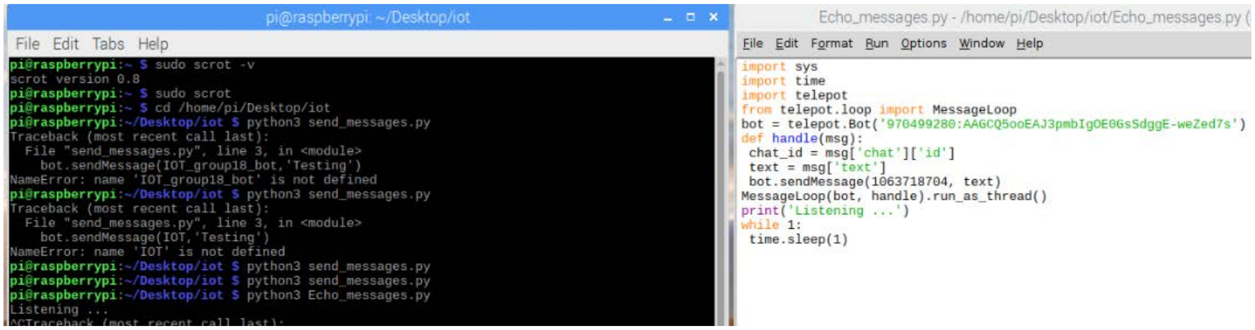
```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install python-pip
sudo pip3 install teleport
```

Step 4: Programming for Raspberry Pi

Task1: Messages receiver.

The code as follow:

```
import time
import telepot
from telepot.loop import MessageLoop
from pprint import pprint
# bot = telepot.Bot('TOKEN ')

bot = telepot.Bot('970499280:AAGCQ5ooEAJ3pmbIgOE0GsSdggE-weZed7s')

def handle(msg):
    pprint(msg)

MessageLoop(bot, handle).run_as_thread()
print('Listening ...')

while 1:
    time.sleep(1)
```


pi@raspberrypi: ~/Desktop/iot
File Edit Tabs Help
Echo_messages.py - /home/pi/Desktop/iot/Echo_messages.py (
File Edit Format Run Options Window Help

```
pi@raspberrypi:~ $ sudo scrot -v
scrot version 0.8
pi@raspberrypi:~ $ sudo scrot
pi@raspberrypi:~ $ cd /home/pi/Desktop/iot
pi@raspberrypi:~/Desktop/iot $ python3 send_messages.py
Traceback (most recent call last):
  File "send_messages.py", line 3, in <module>
    bot.sendMessage(IOT_group18_bot,'Testing')
NameError: name 'IOT_group18_bot' is not defined
pi@raspberrypi:~/Desktop/iot $ python3 send_messages.py
Traceback (most recent call last):
  File "send_messages.py", line 3, in <module>
    bot.sendMessage(IOT,'Testing')
NameError: name 'IOT' is not defined
pi@raspberrypi:~/Desktop/iot $ python3 send_messages.py
pi@raspberrypi:~/Desktop/iot $ python3 send_messages.py
pi@raspberrypi:~/Desktop/iot $ python3 Echo_messages.py
Listening ...
^CTraceback (most recent call last):
```

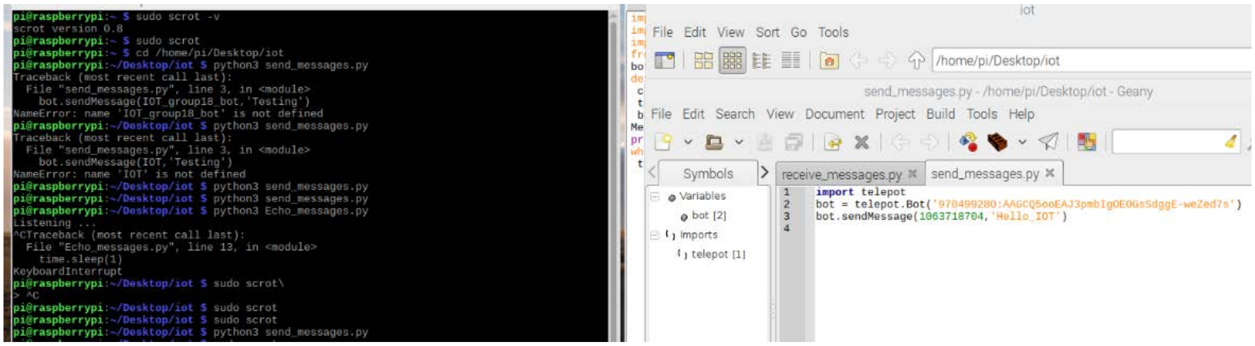
```
import sys
import time
import telepot
from telepot.loop import MessageLoop
bot = telepot.Bot('970499280:AAGCQ5ooEAJ3pmbIgOE0GsSdggE-weZed7s')
def handle(msg):
 chat_id = msg['chat']['id']
 text = msg['text']
 bot.sendMessage(1063718704, text)
MessageLoop(bot, handle).run_as_thread()
print('Listening ...')
while 1:
 time.sleep(1)
```

Fig3a_3. Messages receiver in Raspberry Pi

Task 2. Message transmitter

The code as follow:

```
import telepot
bot = telepot.Bot('970499280:AAGCQ5ooEAJ3pmbIgOE0GsSdggE-weZed7s')
bot.sendMessage(1063718704,'Hello_IOT')
```


iot
File Edit View Sort Go Tools
/home/pi/Desktop/iot
send_messages.py - /home/pi/Desktop/iot - Geany
File Edit Search View Document Project Build Tools Help
Symbols
receive_messages.py
send_messages.py
Variables
bot [2]
Imports
telepot [1]

```
pi@raspberrypi:~ $ sudo scrot -v
scrot version 0.8
pi@raspberrypi:~ $ sudo scrot
pi@raspberrypi:~ $ cd /home/pi/Desktop/iot
pi@raspberrypi:~/Desktop/iot $ python3 send_messages.py
Traceback (most recent call last):
  File "send_messages.py", line 3, in <module>
    bot.sendMessage(IOT_group18_bot,'Testing')
NameError: name 'IOT_group18_bot' is not defined
pi@raspberrypi:~/Desktop/iot $ python3 send_messages.py
Traceback (most recent call last):
  File "send_messages.py", line 3, in <module>
    bot.sendMessage(IOT,'Testing')
NameError: name 'IOT' is not defined
pi@raspberrypi:~/Desktop/iot $ python3 send_messages.py
pi@raspberrypi:~/Desktop/iot $ python3 send_messages.py
pi@raspberrypi:~/Desktop/iot $ python3 Echo_messages.py
Listening ...
^CTraceback (most recent call last):
  File "Echo_messages.py", line 13, in <module>
    time.sleep(1)
KeyboardInterrupt
pi@raspberrypi:~/Desktop/iot $ sudo scrot\
> ^C
pi@raspberrypi:~/Desktop/iot $ sudo scrot
pi@raspberrypi:~/Desktop/iot $ sudo scrot
pi@raspberrypi:~/Desktop/iot $ python3 send_messages.py
```

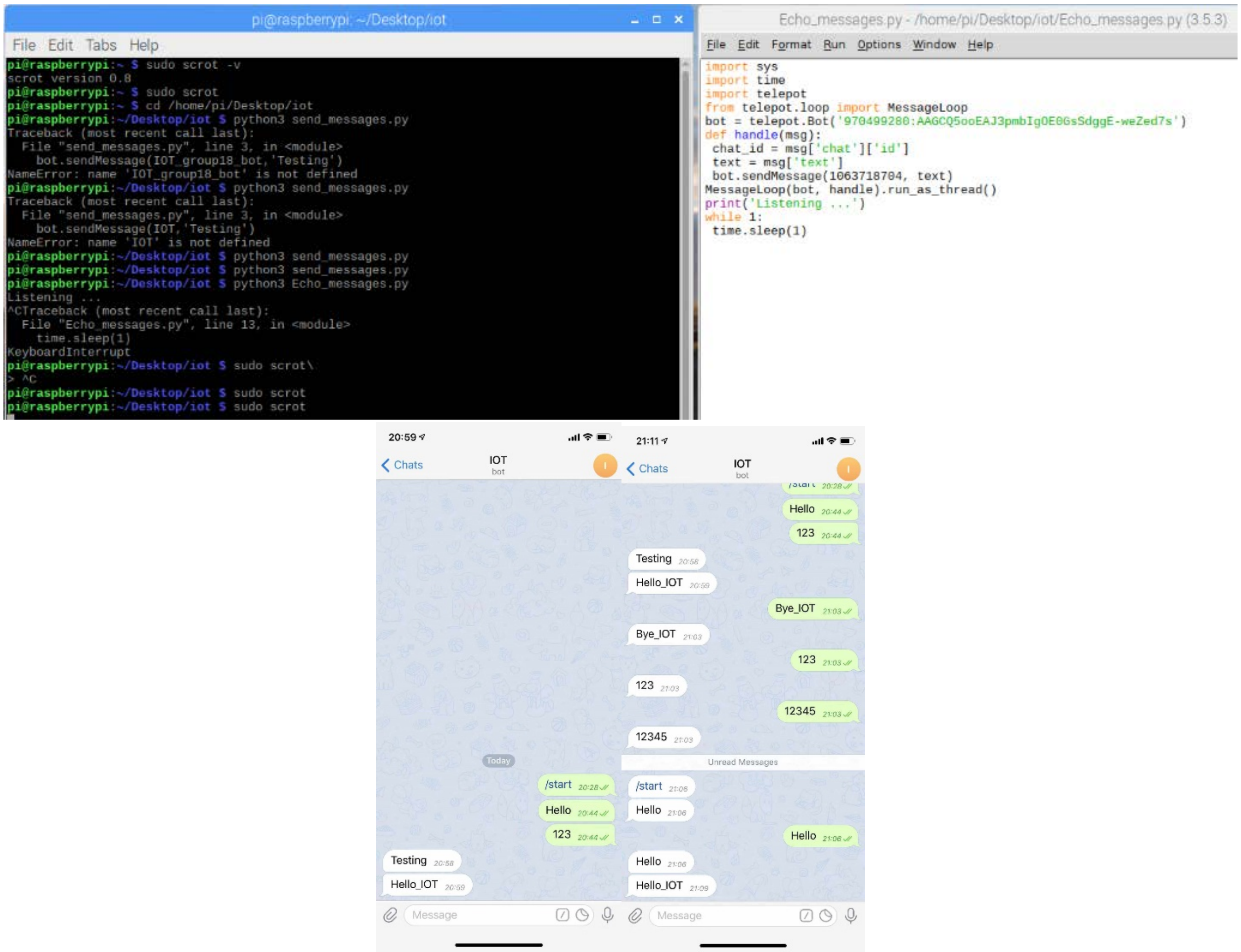
```
import telepot
bot = telepot.Bot('970499280:AAGCQ5ooEAJ3pmbIgOEOGsSdggE-weZed7s')
bot.sendMessage(1063718704,'Hello_IOT')
```

Fig3a_4. Messages transmitter in Raspberry Pi

Message echo:

The code as follow:

```
import sys
import time
import telepot
from telepot.loop import MessageLoop

bot = telepot.Bot('970499280:AAGCQ5ooEAJ3pmbIgOE0GsSdggE-weZed7s')
def handle(msg):
 chat_id = msg['chat']['id']
 text = msg['text']
 bot.sendMessage(1063718704, text)
MessageLoop(bot, handle).run_as_thread()

print('Listening ...')

while 1:
 time.sleep(1)
```


pi@raspberrypi: ~/Desktop/iot
File Edit Tabs Help
Echo_messages.py - /home/pi/Desktop/iot/Echo_messages.py (3.5.3)
File Edit Format Run Options Window Help
20:59
Chats
IOT
bot
21:11
Today
Unread Messages
Message

```
pi@raspberrypi:~ $ sudo scrot -v
scrot version 0.8
pi@raspberrypi:~ $ sudo scrot
pi@raspberrypi:~ $ cd /home/pi/Desktop/iot
pi@raspberrypi:~/Desktop/iot $ python3 send_messages.py
Traceback (most recent call last):
  File "send_messages.py", line 3, in <module>
    bot.sendMessage(IOT_group18_bot,'Testing')
NameError: name 'IOT_group18_bot' is not defined
pi@raspberrypi:~/Desktop/iot $ python3 send_messages.py
Traceback (most recent call last):
  File "send_messages.py", line 3, in <module>
    bot.sendMessage(IOT,'Testing')
NameError: name 'IOT' is not defined
pi@raspberrypi:~/Desktop/iot $ python3 send_messages.py
pi@raspberrypi:~/Desktop/iot $ python3 send_messages.py
pi@raspberrypi:~/Desktop/iot $ python3 Echo_messages.py
Listening ...
^CTraceback (most recent call last):
  File "Echo_messages.py", line 13, in <module>
    time.sleep(1)
KeyboardInterrupt
pi@raspberrypi:~/Desktop/iot $ sudo scrot\
> ^C
pi@raspberrypi:~/Desktop/iot $ sudo scrot
pi@raspberrypi:~/Desktop/iot $ sudo scrot
```

```
import sys
import time
import telepot
from telepot.loop import MessageLoop
bot = telepot.Bot('970499280:AAGCQ5ooEAJ3pmbIgOE0GsSdggE-weZed7s')
def handle(msg):
 chat_id = msg['chat']['id']
 text = msg['text']
 bot.sendMessage(1063718704, text)
MessageLoop(bot, handle).run_as_thread()
print('Listening ...')
while 1:
 time.sleep(1)
```

/start 20:28
Hello 20:44
123 20:44
Testing 20:58
Hello_IOT 20:59

/start 20:28
Hello 20:44
123 20:44
Testing 20:58
Hello_IOT 20:59
Bye_IOT 21:03
Bye_IOT 21:03
123 21:03
123 21:03
12345 21:03
12345 21:03
/start 21:06
Hello 21:06
Hello 21:06
Hello 21:06
Hello_IOT 21:09

Fig3a_5. Messages Connectin between phone and Raspberry Pi

From this Communication, we know that Raspberry Pi and smartphone are very easily to be connected throw Internet. Therefore, we can design some simple system in Pi, when the data is very large, we are able to transmit the dataset to cloud service. In addition, we can send message to the user’s smartphone easily.

Laboratory 3b

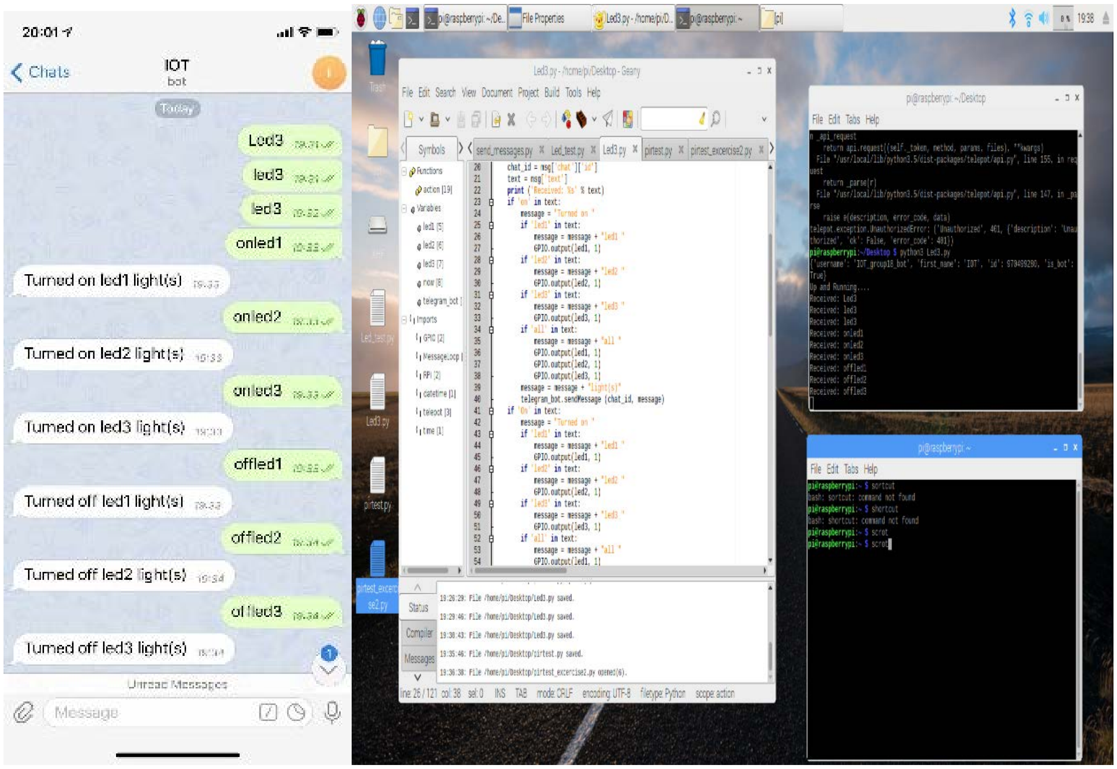
Task 3. GPIO Pins controller

Observation (Led_test.py): After we proceed with the Python program, the program will turn on the LED one by one. The first LED will turn on, then after 1 second, it will turn off. After further 1 second, the second LED turn on, it will also turn off after 1 second. Then the Last LED will turn on and also turn off after 1 second. The proceed will keep looping unless you press CTRL-C to break the program.

Observation (3led.py):

This program will let you control the turn on/off LED with mobile phone.

The communication between mobile phone and Raspberry Pi are shown in picture a. We input particular words and the number of LED in the telegram bot. When the Python program receive the message, it will execute according to the order. However, we added new code of LED called ‘all’, which represented all LED. Finally, we can control the LED on, off with ‘on’ plus the number of led and ‘off’ pulse the number of led. On the other hand, we can control all the LED on, off with ‘all’ plus the light(s) and ‘all’ plus light(s).



Led3
led3
led3
onled1
Turned on led1 light(s)
onled2
Turned on led2 light(s)
onled3
Turned on led3 light(s)
offled1
Turned off led1 light(s)
offled2
Turned off led2 light(s)
offled3
Turned off led3 light(s)

```
    chat_id = msg['chat']['id']
    text = msg['text']
    print ('Received: %s' % text)
    if 'on' in text:
        message = 'Turned on '
        if 'led1' in text:
            message = message + "led1 "
            GPIO.output(led1, 1)
        if 'led2' in text:
            message = message + "led2 "
            GPIO.output(led2, 1)
        if 'led3' in text:
            message = message + "led3 "
            GPIO.output(led3, 1)
        if 'all' in text:
            message = message + "all "
            GPIO.output(led1, 1)
            GPIO.output(led2, 1)
            GPIO.output(led3, 1)
        message = message + "light(s)"
        telegram_bot.sendMessage (chat_id, message)
    if 'On' in text:
        message = 'Turned on '
        if 'led1' in text:
            message = message + "led1 "
            GPIO.output(led1, 1)
        if 'led2' in text:
            message = message + "led2 "
            GPIO.output(led2, 1)
        if 'led3' in text:
            message = message + "led3 "
            GPIO.output(led3, 1)
        if 'all' in text:
            message = message + "all "
            GPIO.output(led1, 1)
```

```
19:26:29: File /home/pi/Desktop/Led3.py saved.
19:29:46: File /home/pi/Desktop/Led3.py saved.
19:30:43: File /home/pi/Desktop/Led3.py saved.
19:35:46: File /home/pi/Desktop/pirtest.py saved.
19:36:38: File /home/pi/Desktop/pirtest_excercise2.py opened(6).
```

```
n _api_request
    return api.request((self._token, method, params, files), **kwargs)
  File "/usr/local/lib/python3.5/dist-packages/telepot/api.py", line 155, in req
uest
    return _parse(r)
  File "/usr/local/lib/python3.5/dist-packages/telepot/api.py", line 147, in _pa
rse
    raise e(description, error_code, data)
telepot.exception.UnauthorizedError: ('Unauthorized', 401, {'description': 'Unau
thorized', 'ok': False, 'error_code': 401})
pi@raspberrypi:~/Desktop $ python3 Led3.py
{'username': 'IOT_group18_bot', 'first_name': 'IOT', 'id': 978499280, 'is_bot':
True}
Up and Running....
Received: Led3
Received: led3
Received: led3
Received: onled1
Received: onled2
Received: onled3
Received: offled1
Received: offled2
Received: offled3
```

```
pi@raspberrypi:~ $ sortcut
bash: sortcut: command not found
pi@raspberrypi:~ $ shortcut
bash: shortcut: command not found
pi@raspberrypi:~ $ scrot
pi@raspberrypi:~ $ scrot
```

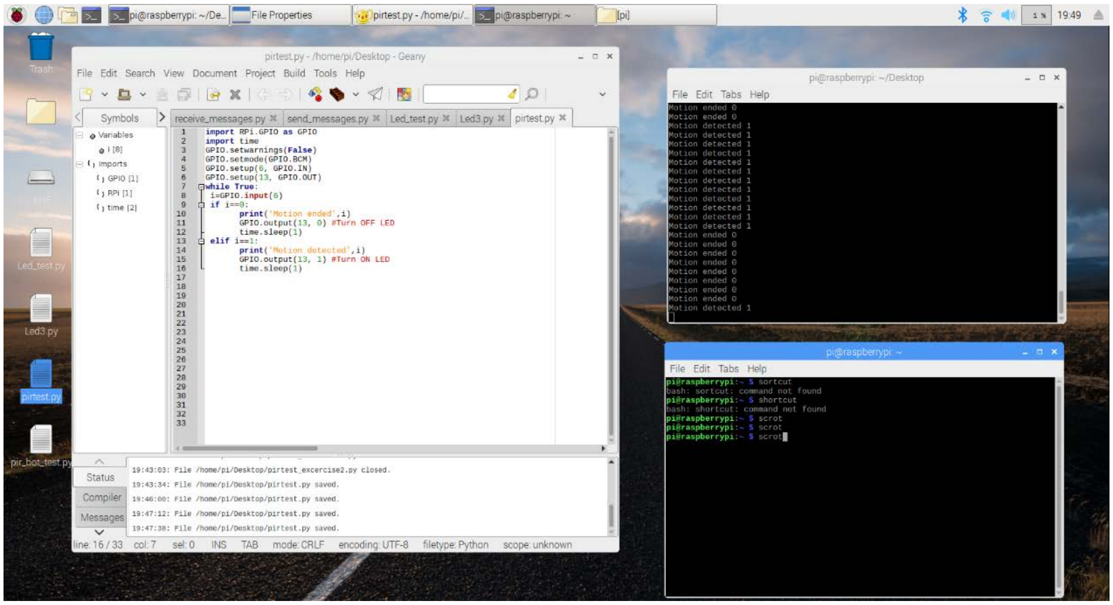
Fig3b_1.The communication between mobile phone and Raspberry Pi

Task 4. Interfacing the PIR Motion Sensor to the Raspberry Pi

Observation (pirtest.py):

PIR (passive infrared) motion sensor mainly function is to detect change in the infrared radiation on it. Once any object or human passes in the front of PIR sensor, the sensor's will detect the temperature rise from ambient temperature to object temperature and then back again to ambient temperature. This change will let the PIR sensor to output corresponding High or Low signals.

After running the program, if any objects moved in front of the PIR sensor, the LED will turn on. The program will detect the signal output from the signal every second. However, the sensor itself build-in a delay for each change.   The code of pirtest.py is shown in picture b.



```
import RPi.GPIO as GPIO
import time
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(6, GPIO.IN)
GPIO.setup(13, GPIO.OUT)
while True:
 i=GPIO.input(6)
 if i==0:
        print('Motion ended',i)
        GPIO.output(13, 0) #Turn OFF LED
        time.sleep(1)
 elif i==1:
        print('Motion detected',i)
        GPIO.output(13, 1) #Turn ON LED
        time.sleep(1)
```

```
19:43:03: File /home/pi/Desktop/pirtest_excercise2.py closed.
19:43:34: File /home/pi/Desktop/pirtest.py saved.
19:46:00: File /home/pi/Desktop/pirtest.py saved.
19:47:12: File /home/pi/Desktop/pirtest.py saved.
19:47:38: File /home/pi/Desktop/pirtest.py saved.
```

line: 16 / 33   col: 7   sel: 0   INS   TAB   mode: CRLF   encoding: UTF-8   filetype: Python   scope: unknown

pi@raspberrypi: ~/Desktop

```
Motion ended 0
Motion ended 0
Motion detected 1
Motion detected 1
Motion detected 1
Motion detected 1
Motion detected 1
Motion detected 1
Motion detected 1
Motion detected 1
Motion detected 1
Motion detected 1
Motion detected 1
Motion detected 1
Motion ended 0
Motion ended 0
Motion ended 0
Motion ended 0
Motion ended 0
Motion ended 0
Motion ended 0
Motion ended 0
Motion detected 1
```

pi@raspberrypi: ~

```
pi@raspberrypi:~ $ sortcut
bash: sortcut: command not found
pi@raspberrypi:~ $ shortcut
bash: shortcut: command not found
pi@raspberrypi:~ $ scrot
pi@raspberrypi:~ $ scrot
pi@raspberrypi:~ $ scrot
```

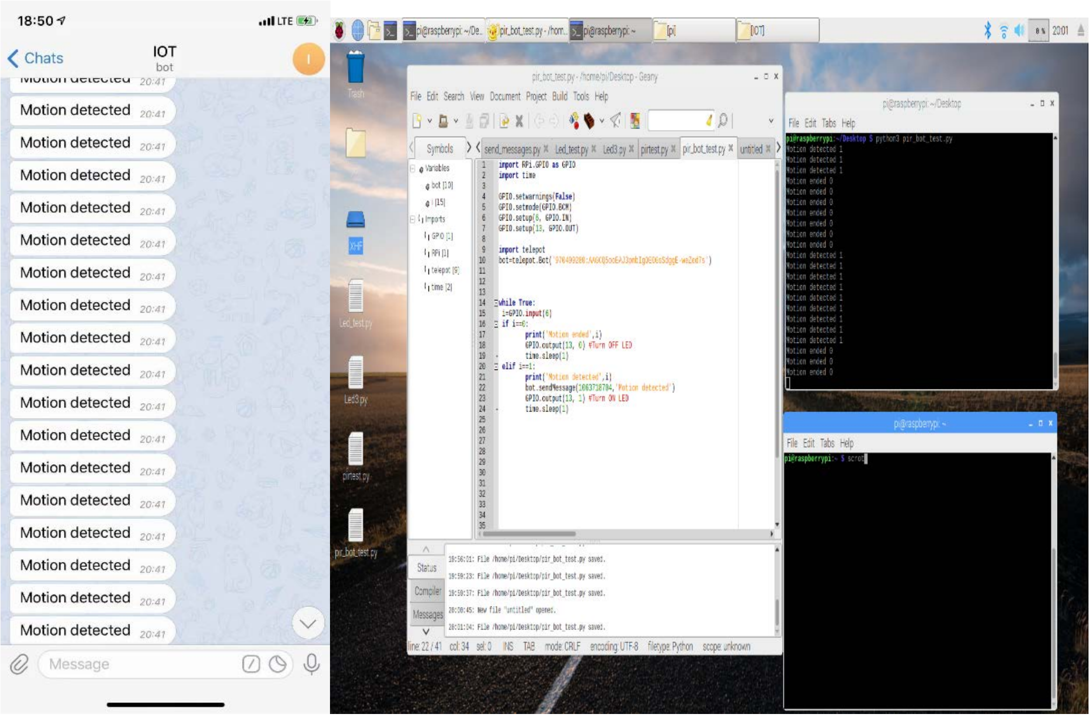
Fig3b_2. pirtest.py

Exercise 2

To enable communication with the mobile phone so that remote users can receive warning when motion is detected, create a file "pir_bot_test.py" and edit it. Run the script and observe the result. Demonstrate your result to the tutor.

Observation (pir_bot_test.py):

The results of pir_bot_test.py are shown in picture b. If there are objects in front of PIR sensor, LED will turn on and the mobile phone will receive the message about 'Motion detected', once no object movement in front of the Sensor, LED will turn off. Please noticed each signal changes from the PIR sensor will have built-in delay.



Motion detected 20:41
Motion detected 20:41
Motion detected 20:41
Motion detected 20:41
Motion detected 20:41
Motion detected 20:41
Motion detected 20:41
Motion detected 20:41
Motion detected 20:41
Motion detected 20:41
Motion detected 20:41
Motion detected 20:41
Motion detected 20:41
Motion detected 20:41
Motion detected 20:41
Motion detected 20:41
Motion detected 20:41
Motion detected 20:41

```
import RPi.GPIO as GPIO
import time

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(6, GPIO.IN)
GPIO.setup(13, GPIO.OUT)

import telepot
bot=telepot.Bot('970499280:AAGCQ5ooEAJ3pmbIgOE0GsSdggE-weZed7s')



while True:
 i=GPIO.input(6)
 if i==0:
        print('Motion ended',i)
        GPIO.output(13, 0) #Turn OFF LED
        time.sleep(1)
 elif i==1:
        print('Motion detected',i)
        bot.sendMessage(1063718704,'Motion detected')
        GPIO.output(13, 1) #Turn ON LED
        time.sleep(1)
```

```
19:56:01: File /home/pi/Desktop/pir_bot_test.py saved.
19:59:23: File /home/pi/Desktop/pir_bot_test.py saved.
19:59:37: File /home/pi/Desktop/pir_bot_test.py saved.
20:00:45: New file "untitled" opened.
20:01:04: File /home/pi/Desktop/pir_bot_test.py saved.
```

line: 22 / 41 col: 34 sel: 0 INS TAB mode: CRLF encoding: UTF-8 filetype: Python scope: unknown

```
pi@raspberrypi:~/Desktop $ python3 pir_bot_test.py
Motion detected 1
Motion detected 1
Motion detected 1
Motion ended 0
Motion ended 0
Motion ended 0
Motion ended 0
Motion ended 0
Motion ended 0
Motion ended 0
Motion detected 1
Motion detected 1
Motion detected 1
Motion detected 1
Motion detected 1
Motion detected 1
Motion detected 1
Motion detected 1
Motion detected 1
Motion ended 0
Motion ended 0
Motion ended 0
```

```
pi@raspberrypi:~ $ scrot
```

Fig3b_4.The results of pir_bot_test.py

pir_bot_test.py

```
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(6, GPIO.IN)
GPIO.setup(13, GPIO.OUT)


import telepot
bot=telepot.Bot('970499280:AAGCQ5ooEAJ3pmbIgOE0GsSdggE-weZed7s')


while True:
 i=GPIO.input(6)
 if i==0:
 print('Motion ended',i)
 GPIO.output(13, 0) #Turn OFF LED
 time.sleep(1)
```

```
elif i==1:
print('Motion detected',i)
bot.sendMessage(1063718704,'Motion detected')
GPIO.output(13, 1) #Turn ON LED
time.sleep(1)
```

Exercise 3

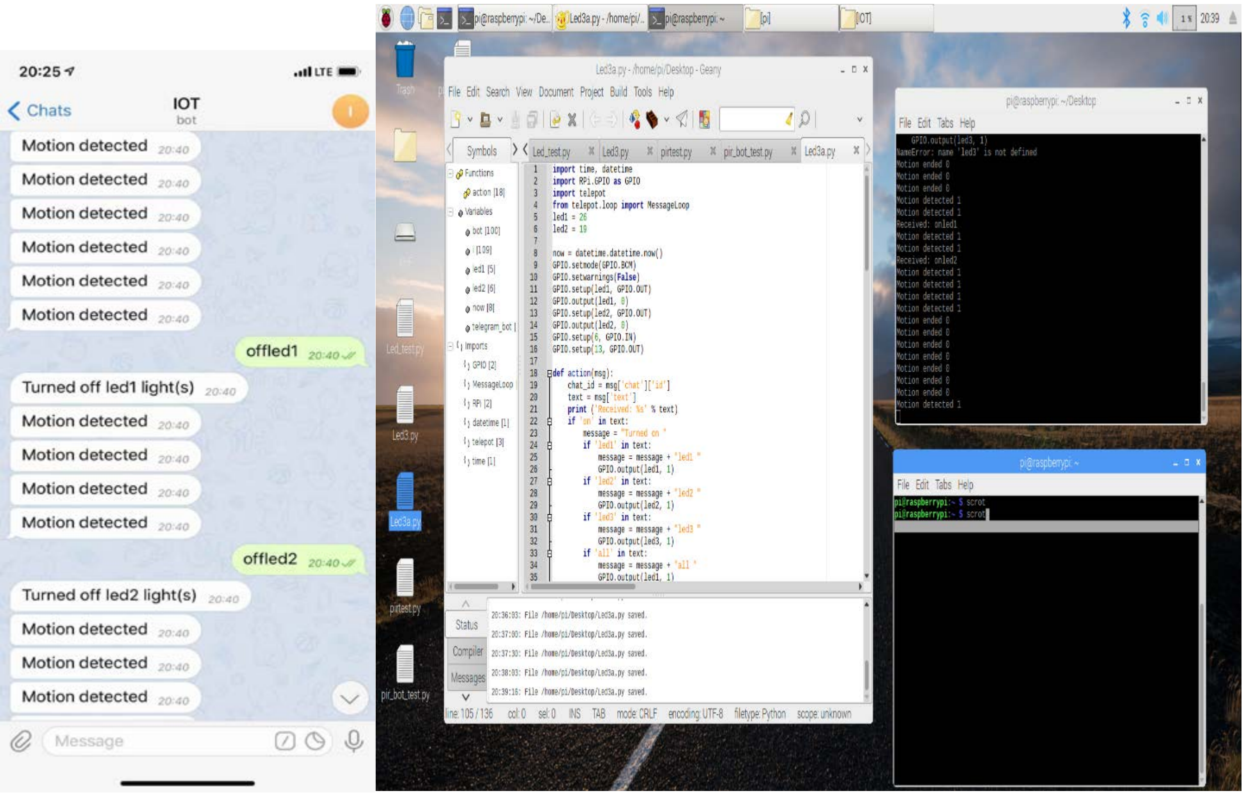
Create a file “pir_bot_led_test.py” and edit it so that one LED will indicate the motion detection while the other LED will indicate the communications. Run the script and observe the result. Demonstrate your result to the tutor.

Observation (pir_bot_led_test.py):

The results of pir_bot_test.py are shown in c.

If there are objects in front of PIR sensor, LED will turn on and the mobile phone will receive the message about ‘Motion detected’). When no movement in front of the sensor, LED will turn off.

Simultaneously, we can control another LED on, off with ‘on’ pulse the number of led and ‘off’ pulse the number of led. On the other hand, we can control all the LED on, off with ‘all’ pulse the light(s) and ‘all’ pulse light(s).



20:25
LTE
Chats
IOT
bot
Message
Led3a.py - /home/pi/Desktop - Geany
File Edit Search View Document Project Build Tools Help
Symbols
Led_test.py
Led3.py
pirtest.py
pir_bot_test.py
Led3a.py
Functions
action [18]
Variables
Imports
Status
Compiler
Messages
line: 105 / 136 col: 0 sel: 0 INS TAB mode: CRLF encoding: UTF-8 filetype: Python scope: unknown
pi@raspberrypi: ~/Desktop
pi@raspberrypi: ~
File Edit Tabs Help
20:39

Motion detected 20:40
Motion detected 20:40
Motion detected 20:40
Motion detected 20:40
Motion detected 20:40
Motion detected 20:40
offled1 20:40
Turned off led1 light(s) 20:40
Motion detected 20:40
Motion detected 20:40
Motion detected 20:40
Motion detected 20:40
offled2 20:40
Turned off led2 light(s) 20:40
Motion detected 20:40
Motion detected 20:40
Motion detected 20:40

```
import time, datetime
import RPi.GPIO as GPIO
import telepot
from telepot.loop import MessageLoop
led1 = 26
led2 = 19

now = datetime.datetime.now()
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(led1, GPIO.OUT)
GPIO.output(led1, 0)
GPIO.setup(led2, GPIO.OUT)
GPIO.output(led2, 0)
GPIO.setup(6, GPIO.IN)
GPIO.setup(13, GPIO.OUT)

def action(msg):
    chat_id = msg['chat']['id']
    text = msg['text']
    print ('Received: %s' % text)
    if 'on' in text:
        message = "Turned on "
        if 'led1' in text:
            message = message + "led1 "
            GPIO.output(led1, 1)
        if 'led2' in text:
            message = message + "led2 "
            GPIO.output(led2, 1)
        if 'led3' in text:
            message = message + "led3 "
            GPIO.output(led3, 1)
        if 'all' in text:
            message = message + "all "
            GPIO.output(led1, 1)
```

```
20:36:03: File /home/pi/Desktop/Led3a.py saved.
20:37:00: File /home/pi/Desktop/Led3a.py saved.
20:37:30: File /home/pi/Desktop/Led3a.py saved.
20:38:03: File /home/pi/Desktop/Led3a.py saved.
20:39:16: File /home/pi/Desktop/Led3a.py saved.
```

```
    GPIO.output(led3, 1)
NameError: name 'led3' is not defined
Motion ended 0
Motion ended 0
Motion ended 0
Motion detected 1
Motion detected 1
Received: onled1
Motion detected 1
Motion detected 1
Received: onled2
Motion detected 1
Motion detected 1
Motion detected 1
Motion detected 1
Motion ended 0
Motion ended 0
Motion ended 0
Motion ended 0
Motion ended 0
Motion ended 0
Motion ended 0
Motion detected 1
```

```
pi@raspberrypi:~ $ scrot
pi@raspberrypi:~ $ scrot
```

Fig3b_5.The results of pir_bot_led_test.py