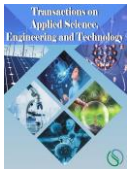

Appendix 1

Here is a Python program that simulates the data transfer process, updates the FSPL, SNR, MODULATION values, and plots the corresponding graphs:

```
python
CopyEdit
import matplotlib.pyplot as plt
import pandas as pd

# Input data
FSPL = [136, 135, 134, 133, 132, 131, 130, 129, 128, 127,
126, 125, 124, 123, 122, 121, 120, 119, 118, 117,
116, 115, 114, 113, 112, 111, 110, 109, 108] # dB

SNR = [5, 5.5, 6.7, 7.3, 8.3, 8.9, 9.8, 10.6, 11.7, 12.8,
13.7, 14.4, 15.8, 16.7, 17.5, 18.8, 19.2, 20.3, 21.7, 22.5,
23.5, 24.6, 25.8, 26.7, 27.8, 28.5, 29.2, 30.5, 31.5] # dB

MODULATION = [1] * len(SNR) # Initial modulation = 1

# Update MODULATION depending on SNR
for i in range(len(SNR)):
if SNR[i] > 12:
MODULATION[i] = 2
if SNR[i] > 13:
 MODULATION[i] = 3
 if SNR[i] > 14:
 MODULATION[i] = 4
 if SNR[i] > 20:
 MODULATION[i] = 5
 if SNR[i] > 25:
 MODULATION[i] = 6
 if SNR[i] > 28:
 MODULATION[i] = 7
 if SNR[i] > 30:
 MODULATION[i] = 8

# Create a results table
data = pd.DataFrame({"Time (min)": list(range(len(FSPL))),
 "FSPL (dB)": FSPL,
 "SNR (dB)": SNR,
 "MODULATION": MODULATION})

# Output table
print(data)

# Building graphs
plt.figure(figsize=(12, 6))

plt.subplot(3, 1, 1)
plt.plot(data["Time (min)"], data["FSPL (dB)"], marker="o", linestyle="-", color="blue", label="FSPL")
plt.xlabel("Time (min)")
plt.ylabel("FSPL (dB)")
plt.title("FSPL vs Time")
plt.legend()
plt.grid()

plt.subplot(3, 1, 2)
plt.plot(data["Time (min)"], data["SNR (dB)"], marker="s", linestyle="-", color="green", label="SNR")
plt.xlabel("Time (min)")
plt.ylabel("SNR (dB)")
plt.title("SNR vs Time")
plt.legend()
plt.grid()
```

```
plt.subplot(3, 1, 3)
plt.plot(data["Time (min)"], data["MODULATION"], marker="^", linestyle="-", color="red", label="MODULATION")
plt.xlabel("Time (min)")
plt.ylabel("MODULATION")
plt.title("MODULATION vs Time")
plt.legend()
plt.grid()

plt.tight_layout()
plt.show()
```

Appendix 2

Python code that uses the Random Forest Classifier classification model to predict the MODULATION value given an FSPL. The model is trained on the known data, then used to make predictions on new FSPL values.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier

# Training data
FSPL_data = np.array([
136, 135, 134, 133, 132, 131, 130, 129, 128, 127,
126, 125, 124, 123, 122, 121, 120, 119, 118, 117,
116, 115, 114, 113, 112, 111, 110, 109, 108
]).reshape(-1, 1)

MODULATION_data = np.array([
1, 1, 1, 1, 1, 1, 2, 2, 2,
3, 4, 4, 4, 4, 4, 5, 5, 5,
5, 5, 6, 6, 6, 7, 7, 8, 8
])

# Train classifier
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(FSPL_data, MODULATION_data)

# New data (FSPL and SNR over time)
FSPL_new = np.array([
129, 128, 127, 126, 125, 125, 114, 111, 109, 108,
119, 118, 126, 129, 136, 135, 134
]).reshape(-1, 1)

# Modulation prediction
MODULATION_pred = model.predict(FSPL_new)

# Time scale
time_minutes = np.arange(len(FSPL_new))

# Result table
results = pd.DataFrame({
"Time (min)": time_minutes,
"FSPL (dB)": FSPL_new.flatten(),
"MODULATION": MODULATION_pred
})

print("Result table:")
print(results)

# Plotting
plt.figure(figsize=(10, 5))
plt.plot(time_minutes, MODULATION_pred, marker='o', linestyle='-', color='green')
plt.title("MODULATION(t) vs. time (classification model)")
plt.xlabel("Time (min)")
plt.ylabel("MODULATION")
plt.grid(True)
plt.xticks(time_minutes)
plt.yticks(range(1, 9))
plt.show()
```