פרויקט

איסוף נתוני

זיהום אוויר

תוכן עניינים

קווים כללים

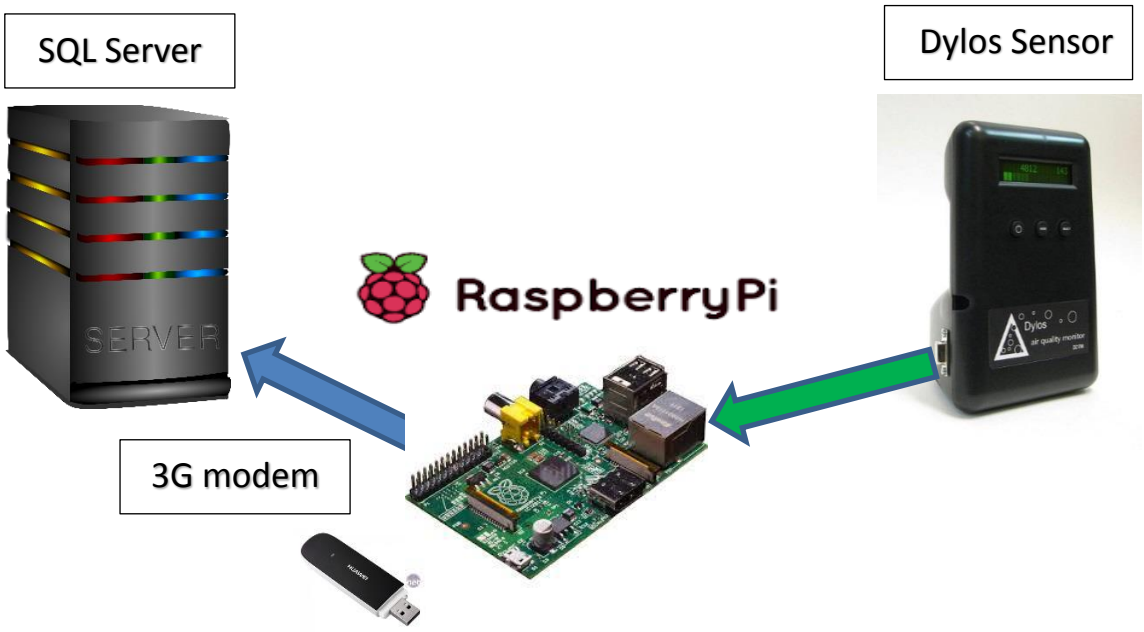
מדריכים

תיעוד חיצוני

<u>נספחים</u>

תיאור כללי:

איסוף נתוני זיהום אויר מחיישנים ואחסון המידע בבסיס נתונים לצורך מחקר עתידי.



רכיבי הפרויקט:

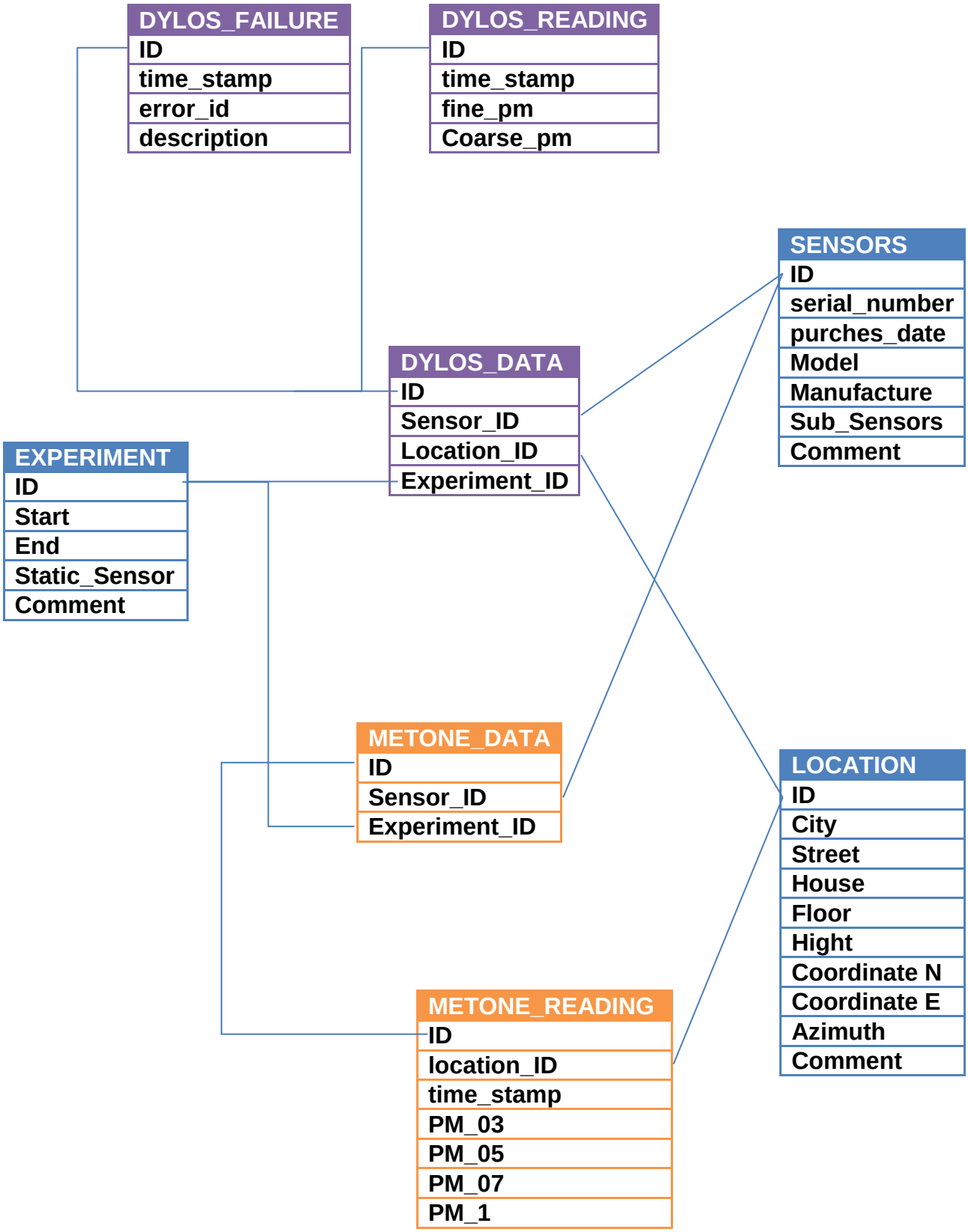
- <u>חיישנים</u>
  - חיישן dylos:

    דוגם נתוני זיהום משני סוגים בזמן אמת, בעל יציאה טורית.
  - בהמשך שימוש בחיישן metone (יתכנו חיישנים נוספים)
- <u>בקר</u>
  - Raspberry pi תחת מערכת הפעלה NOOBS.
- <u>מודם סלולרי</u>
  - T & A
  - בהמשך שימוש במודמים נוספים (עדיפות למודם ללא מצב אחסון)

בסיס הנתונים- שרת SQL (הוקם ע"י שחר)

כניסה דרך: http://www.phpmyadmin.co/
פרטי השרת-
Server: 132.68.226.244
User name: APMoD
Password: AirPol2015

סכמת בסיס הנתונים



מטרות:

- קריאת נתונים מחיישן מסוג dylos
- יצירת תקשורת סלולרית עם הבקר
- שליחת נתונים מהבקר לשרת SQL באמצעות תקשורת סלולרית

מטרות עתידיות:

- יצירת קוד מקבילי עבור קריאה מהחיישן ושליחה לשרת (פסיקות)
- הוספת חיישנים נוספים (metone)
- יצירת דוחות וניתוחים על בסיס המידע בשרת
- תמיכה בשליחת נתונים במקרה של הפעלה מחדש באופן לא מבוקר
- בדיקות ברמת השרת (לדוגמא: בדיקת נפילת רשת)
- הוספת מצב טסט לבדיקת תקינות ראשונית בזמן יציאה לשטח

תיאור הקבצים:

קריאת הנתונים מהחיישן ושליחתם לשרת באמצעות הבקר נעשים באמצעות תכנית הכתובה בשפת python השמורה בקובץ **DylosToSerial.py**.

DylosConf.py- סקריפט אינטראקטיבי המאפשר שינוי/אתחול הגדרות בטרם הוצאת המערכת לשטח.

setting.py- קובץ המכיל הגדרות לצורך שימוש התוכנית הראשית (DylosToSerial.py)

אופן פעולת התוכנית הראשית:

קריאת נתונים מהחיישן כל דקה ע"י קריאה והמתנה לקריאה הבאה.
לאחר מספר קבוע של קריאות (update period המוגדר ב-DylosConf.py),
ננצל את זמן ההמתנה להתחברות ושליחת הנתונים לשרת באמצעות תקשורת סלולרית (או כל חיבור אינטרנטי אחר).

גיבוי נתונים:

נתוני הקריאה נשמרים על גבי הבקר בקבצי excel יומיים (פורמט csv) בהתאם לתיקיית backup folder המוגדרת בקובץ הקונפיגורציה.
מיד לאחר הקריאה היא נכתבת לקובץ הגיבוי היומי.
לאחר מספר ימים: save period (המוגדר בקובץ הקונפיגורציה) קובץ הגיבוי ימחק.

העברת קריאות לשרת:

הקריאות מהחיישן נשמרות ברשימה זמנית ולאחר מספר קריאות (update period) בזמן ההמתנה לקריאה הבאה ניצור שאילתת SQL מתאימה, נבצע התחברות לשרת ושליחה של השאילתה.
קריאות שנשלחו בהצלחה לשרת ימחקו מהרשימה.

<u>שיקולים להגדרת update period=15:</u>

בכל packet ניתן להעביר כ400 תווים.
מצד אחד, לא נרצה לספוג overhead גדול בשליחת שאילתה בודדת
ומצד שני, נרצה לשמור על בסיס הנתונים עדכני ככל האפשר.
על כן, בכל 15 דקות נשלח 15 קריאות בכך נשמור על עדכניות בבסיס הנתונים וגם נשמור על overhead סביר.
(מצורף סיכום פגישה עם איתי דברן בנושא)

עדכון/הוספת רכיבים:

1. **הוספת מודם**

- מודם בעל מצב תקשורת בלבד:

אין צורך לבצע כלום, בקובץ הקונפיגורציה להגדיר את ה-Modem כother.

- מודם בעל שני מצבים (תקשורת ואחסון):

יש ליצור קובץ קונפיגורציה מתאים בשם <name>.conf
ולשמור ב-/home/pi/Dylos
בהתאם לשלבים 3-6 מדריך ליצירת תקשורת סלולרית ראשונית ב- raspberry pi

2. **הוספת סוג חיישן**

יש לשנות בקובץ הקונפיגורציה sensor type ולהוסיף קוד מתאים לקריאה מהחיישן החדש בסקריפט DylosToSerial.py.

פרוטוקול שגיאות

שגיאה	סיבות אפשריות	תגובת התוכנית	תגובת השרת	פתרונות אפשריים
לא נמצא חיבור סריאלי לחיישן	1.אין חיבור סריאלי לבקר (הכבל לא מחובר לבקר/הכבל לא תקין) 2.הגדרה שגויה בשם החיבור הסריאלי בקובץ הקונפיגורציה (sensor serial name)	יתבצע ניסיון חיבור נוסף כל SERIAL_CONNECT_PERIOD (המוגדר בקובץ הקבועים) תופיע הודעת שגיאה מתאימה על המסך. לא יתבצעו קריאות	לא יתקבלו קריאות	1. חיבור הכבל מחדש 2. הגדרת שם החיבור הסריאלי מחדש (הרצת קובץ קונפיגורציה לצורך עדכון) 3. החלפת כבל והגדרת שם חיבור סריאלי.
קבלת קריאה ריקה מהחיישן	1.החיישן כבוי 2.החיישן מנותק (אך הכבל מחובר לrp) 3.יציאה מסנכרון (יסתדר בקריאה הבאה)	בקובץ הcsv תישמר קריאה עם ערכי "none"	תשלח לטבלת DYLOS_FAILURE שגיאה מספר 1: NO AVAILABLE DATA הערה: מופיע לעיתים קרובות בקריאה הראשונה	בדיקה כי החיישן דלוק ומחובר היטב
קבלת קריאה לא תקינה מהחיישן	1. הכבל אינו מחובר היטב מצד החיישן 2. הכבל תקול (יש שם צ׳יפ FTDI)	בקובץ הcsv תישמר קריאה עם ערכי "error"	תשלח לטבלת DYLOS_FAILURE שגיאה מספר 2: INVALID DATA	1. בדיקה כי הכבל מחובר היטב 2. החלפת כבל

שגיאה	סיבות אפשריות	תגובת התוכנית	תגובת השרת	פתרונות אפשריים
אין חיבור לאינטרנט	1. לא חובר לבקר אמצעי תקשורת (מודם, כבל רשת וכו) 2. יש בעיה בהגדרת ספק השירות (בקובץ הקונפיגורציה) 3. יש בעיה בספק השירות	תופיע הודעת שגיאה מתאימה על המסך. <u>פרוטוקול טיפול:</u> 1. ניסיון חיבור לרשת לאחר כל קריאה. 2. במידה ולא נוצר חיבור לאחר FIRST_REBOOT_TIME (המוגדר בקבועים) תתבצע בצורה מבוקרת הפעלה מחדש של מערכת ההפעלה: הקריאות שטרם נשלחו יגובו בקובץ זמני וישוחזרו בהפעלת המערכת מחדש. 3. במידה ולא נוצר חיבור לאחר הפעלת המערכת מחדש תתבצע הפעלה מבוקרת כל 24 שעות.	לא יתקבלו קריאות. כאשר החיבור יתחדש ישלחו כל הקריאות שנדגמו בזמן שלא הייתה תקשורת	1. בדיקה כי מחובר אמצעי תקשורת. 2. בדיקה כי הגדרות הקונפיגורציה תקינות. (ספק, הגדרת מודם) 3. בדיקה מול הספק אם יש תקלה
לא ניתן להתחבר לשרת	1. הגדרות השרת אינן תקינות בקובץ הקונפיגורציה 2. עומס על השרת 3. השרת נפל	תופיע הודעת שגיאה מתאימה על המסך. <u>פרוטוקול טיפול:</u> 1. נשלח מייל המתריע כי יתכן וישנה בעיה בשרת (בהתאם לכתובת המייל המופיעה בקובץ הקונפיגורציה) 2. ניסיון חיבור לשרת לאחר כל קריאה 3. במידה ולא נוצר חיבור במשך SEND_MAIL_PERIOD (המוגדר בקובץ הקבועים) ישלח מייל נוסף באופן סידרתי.	לא יתקבלו קריאות. כאשר יתאפשר חיבור עם השרת ישלחו כל הקריאות שנדגמו בזמן שלא התאפשר חיבור.	1. בדיקת הגדרות השרת בקובץ הקונפיגורציה 2. בדיקת השרת 3. לבחון אופציה להגדיל את הזמן הניתן לחיבור לשרת: CONNECTION_TIMEOUT (המוגדר בקובץ הקבועים)

כדאי לדעת:

1. שעון הבקר מסונכרן לשעון GMT, השעה נדגמת מהרשת.

2. ניתוק המודם כאשר מחובר לאינטרנט יגרום לתקיעה של מערכת ההפעלה.
יש לנתק ולחבר מחדש את הבקר.

3. באתחול לא מבוקר של מערכת ההפעלה הקריאות שטרם נשלחו לשרת לא ישלחו בריצת התוכנית מחדש אך ישמרו בקובץ הגיבוי היומי.

4. המודם TP-LINK MA180 לעיתים לא מצליח לבצע mode-switch ולכן רצוי להשתמש במודם אחר. מחיפוש ברשת עולה כי התופעה ידועה.

5. קיימים כבלים סריאליים שהעברת המידע דרכם לא מתבצעת כראוי ולכן מומלץ לוודא את סוג הכבל לפני הוצאתו לשטח.

6. על מנת להעלות ממשק גרפי (שקול ל-**desktop** של חלונות) יש להיכנס תחת משתמש: pi, סיסמא: raspberry.
ולאחר מכן, לכתוב את הפקודה – startx. (יציאה מהממשק הגרפי וחזרה למסך הראשי תתבצע ע"י ALT+CTRL+F1)

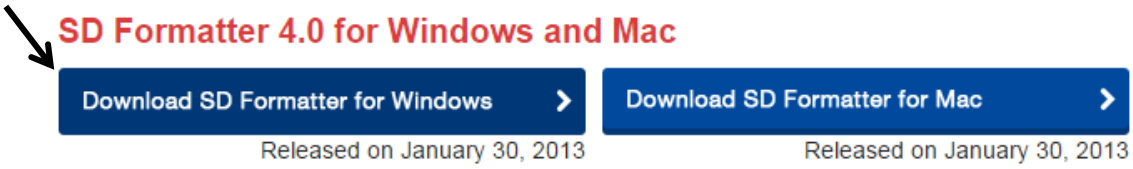
7. על מנת לעצור את ריצת התכנית שפועלת ברקע, יש להיכנס לממשק הגרפי ובשורת הפקודות להכניס את הפקודה **ps –A | grep python**.
פקודה זו תדפיס את פרטי התהליך שמריץ את התכנית שלנו, נזכור את הID של התהליך (עמודה ראשונה).
נכניס את הפקודה **sudo kill -9 <ID>**. כעת הרגנו את התהליך ועצרנו את ריצת התכנית.

8. יצירת העתק של כרטיס SD קיים, מדריך מפורט ניתן למצוא באתר הרשמי של raspberry pi:
https://www.raspberrypi.org/documentation/installation/installing-images/

מדריך להתקנת מערכת

1. פרמט את כרטיס הSD

- הורד תוכנה לצורך הפרמוט מהאתר:
https://www.sdcard.org/downloads/formatter_4/



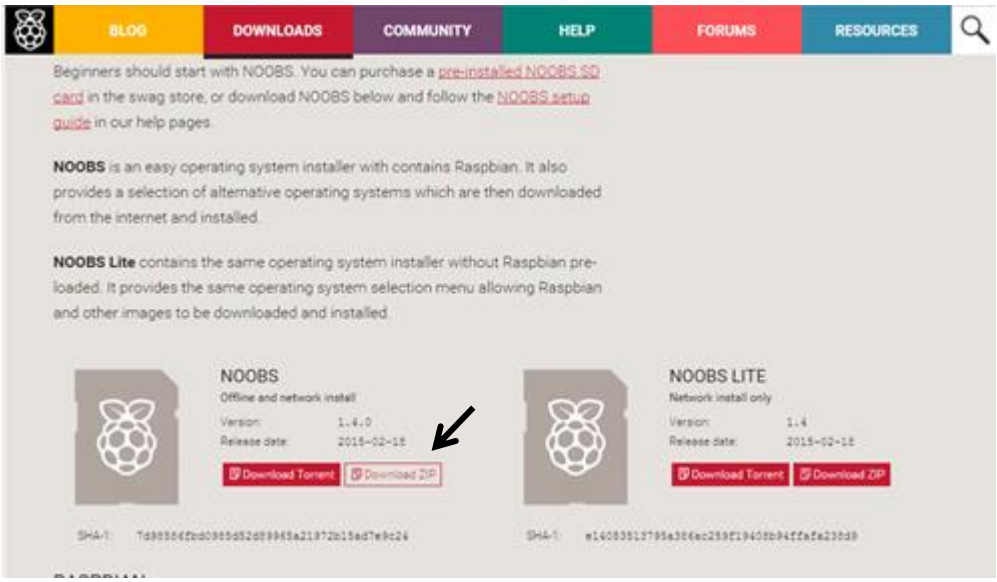
- בצע התקנה של התוכנה

- הכנס את כרטיס הSD למחשב ובצע פרמוט שלו.
ניתן למצוא הסבר מפורט בקישור ההורדה.

2. הורדת מערכת הפעלה

- הכנס לאתר: http://www.raspberrypi.org/downloads/

- לחץ להורדת NOOBS בקובץ ZIP



- בצע Extract מה-ZIP לתוך כרטיס הSD

3. התקנת מערכת הפעלה

- הכנס את כרטיס הSD ל-raspberrypi

- בצע התקנה של מערכת ההפעלה
הסבר מפורט ניתן למצוא ב: http://www.raspberrypi.org/help/noobs-setup/

4. העברת הקבצים ל-raspberrypi

- צור תקיית Dylos: פתח טרמינל ורשום את הפקודה **mkdir Dylos**

- התחבר לאינטרנט (חיבור קווי או wifi)

- הורד את הקבצים הדרושים:
DylosToSerial.py, DylosConf.py, setting.py, wvdial.conf
וקבצי הקונפיגורציות של המודמים למשל: TA.conf, MA180.conf

- העבר/העתק את הקבצים לתקיית Dylos:
הורד ישירות לתקייה במידת האפשר או העבר מdownloads ע"י הפקודה:
cp downloads/<file_name> Dylos

5. הורדת תוספים נדרשים

רשום בטרמינל את הפקודות הבאות (יש צורך בחיבור לאינטרנט):

- **sudo apt-get install python-pip**
- **sudo pip install pyserial –upgrade**
- **sudo apt-get install python-mysql**
- **sudo pip install mysql-connecctor-python**
- **sudo apt-get update**
- **sudo apt-get install ppp usb-modeswitch wvdial**

6. שינוי הקובץ wvdial.conf

- החלף את הקובץ wvdial.conf בקובץ (שהורדנו):

ע"י הפקודה: **sudo cp Dylos/wvdial.conf /etc/wvdial.conf**

7. הרצת קובץ קונפיגורציה לפני יציאה לשטח

מדריך ליצירת תקשורת סלולרית ראשונית ב-raspberry pi

1. **הורדת תוכנה**
   - חבר את הpi raspberry לאינטרנט ע"י wifi/חיבור קווי
   - פתח טרמינל ובצע את הפקודות הבאות:

```
sudo apt-get update
sudo apt-get install ppp usb-modeswitch wvdial
```

2. **נתק את החיבור לאינטרנט ובצע הפעלה מחדש (reboot בטרמינל)**

3. **חבר את המודם ל-raspberry pi**

4. **מצא את קוד המצב של המודם במצב אחסון**

   - פתח טרמינל ובצע את הפקודה הבאה:

```
lsusb
```

   - התבונן בפלט ושמור את המספר עבור חיבור המודם (המסומן בצהוב) יוגדר Default Product

```
pi@raspberrypi ~ $ lsusb
Bus 001 Device 002: ID 0424:9514 Standard Microsystems Corp.
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 003: ID 0424:ec00 Standard Microsystems Corp.
Bus 001 Device 015: ID 2357:0200
Bus 001 Device 013: ID 046d:c52e Logitech, Inc.
Bus 001 Device 014: ID 045e:078c Microsoft Corp.
```

5. **יצירת קובץ קונפיגורציה**

   - בצע את הפקודות הבאות בטרמינל:
     החלף את המספר 2357:0200 במספר שנשמר בשלב 3
     (לשים לב להוסיף את ה\)

```
cd /tmp
Tar –xzvf /usr/share/usb_modeswitch/configPack.tar.gz 2357\:0200
```

   - פתח את הקובץ באמצעות עורך טקסט (למשל leafpad) ע"י הפקודה:
     החלף את המספר 2357:0200 במספר שנשמר בשלב 3

```
leafpad 2357:0200
```

- נעתיק מקובץ זה חלק מהמידע לקובץ הקונפיגורציה
דוגמא עבור הקובץ שיפתח:

```
# TP-Link MA180

TargetVendor=  0x2357
TargetProduct= 0x0201

MessageContent="55534243123456780000000000000061b000000020000000000000000000000"
```

- צור קובץ קונפיגורציה חדש בשם: <modem_name>.conf בתיקיית /home/pi/Dylos
ע"י הפקודה:

leafpad /home/pi/Dylos/<modem_name>.conf

- כתוב אליו את הנתונים באופן הבא:

```
DefaultVendor= 0x2357
DefaultProduct= 0x0200

TargetVendor= 0x2357
TargetProduct= 0x0201

CheckSuccess= 10
InquireDevice= 0

MessageContent="55534243123456780000000000000061b000000020000000000000000000000"
```

כאשר השדות בצבע ורוד לקוחים משלב 4, השדות בצבע ירוק לקוחים מהקובץ המופיע בתחילת העמוד ואילו השדות בצבע שחור קבועים.

השלבים הבאים מיועדים רק עבור חיבור ידני ומידי לרשת.
בהרצת התוכנית לאחר הגדרת קובץ הקונפיגורציה יתבצעו באופן אוטומטי.

6. **העתקת קובץ הקונפיגורציה**
החיבור לאינטרנט באמצעות המודם הסלולרי דורש העברה למצב "מודם".

על מנת שהפקודה להעברת מצב תתבצע בהצלחה נדרש להעתיק את קובץ הקונפיגורציה הנ"ל ל- /etc/usb_modeswitch.conf

ניתן לבצע זאת ע"י הפקודה:

sudo cp /home/pi/Dylos/<modem_name>.conf /etc/usb_modeswitch.conf

7. **העברת המודם ממצב אחסון למצב תקשורת**

יש לבצע את הפקודה:

sudo usb_modeswitch –c /etc/usb_modeswitch.conf

8. **ערוך את קובץ הקונפיגורציה wvdial עבור חיבור לרשת:**

- פתח את הקובץ בעורך טקסט (למשל leafpad) ע"י הפקודה:
- **sudo leafpad /etc/wvdial.conf**

- החלף את "internet", בספק הAPN

```
[Dialer Defaults]
Init1 = ATZ
Init2 = ATQ0 V1 E1 S0=0 &C1 &D2 +FCLASS=0
Init3 = AT+CGDCONT=1,"IP","internet"
Stupid Mode = 1
Modem Type = Analog Modem
ISDN = 0
Phone =*99#
Modem = /dev/gsmmodem
Username = { }
Password = { }
```

9. **התחבר לאינטרנט ע"י הפקודה:**

sudo wvdial 3gconnect

מדריך הגדרת קונפיגורציה לפני הוצאת המערכת לשטח

1. **חבר את המערכת.**
   - חיבור החיישן והמודם לבקר.
   - חיבור הבקר למסך, מקלדת ועכבר.
   - העלאת ממש גרפי של הבקר (בהתאם לסעיף 6 בכדאי לדעת)

2. **יש לבדוק כי כל השלבים במדריך ההתקנה בוצעו.**
   בפרט כי הקבצים הועברו לבקר ה-raspberry pi ובוצעה הורדה של התוספים הנדרשים. (שלבים 4-6 ב- מדריך להתקנת מערכת)

3. **יש לבדוק כי קיים קובץ קונפיגורציה מתאים עבור המודם שברצונך לשלוח לשטח.**
   במידה ואין ניתן ליצור אחד בהתאם לשלבים 3-6 ב- מדריך ליצירת תקשורת סלולרית ראשונית ב- raspberry pi.

4. **מציאת שם החיבור הסריאלי של החיישן.**
   - חבר את החיישן
   - פתח טרמינל והכנס את הפקודה **lsusb**
   - חפש את שם החיבור הסריאלי וזכור אותו על מנת להגדיר בהמשך את sensor serial name (ניתן לשמור חלק מהשם בתנאי שהוא ייחודי. לדוגמא: Prolific בלבד)

```
pi@raspberrypi ~/Dylos $ lsusb
Bus 001 Device 002: ID 0424:9514 Standard Microsystems Corp.
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 003: ID 0424:ec00 Standard Microsystems Corp.
Bus 001 Device 012: ID 067b:2303 Prolific Technology, Inc. PL2303 Serial Port
Bus 001 Device 009: ID 1bbb:0000 T & A Mobile Phones
Bus 001 Device 004: ID 046d:c52e Logitech, Inc.
```

5. **הרצת קובץ הקונפיגורציה.**
   - פתח טרמינל הכנס לתיקיית Dylos והכנס את הפקודה:

python DylosConf.py

כעת, ירוץ הסקריפט בצורה אינטראקטיבית ויש לאתחל/לשנות את ההגדרות.

<u>הגדרות עבור השרת:</u>

- **user**- שם משתמש
- **password**- סיסמא
- **host**- IP של שרת הSQL
- **database**- שם ה-data base אליו כותבים

<u>הגדרות כלליות:</u>

- **ID** - מספר החיישן (ספרות בלבד)
- **Network**- רשת אלחוטית
  (GSM network: Orange, Cellcom, Pelephone, Hot, Golan, 012)

- **Modem**- שם המודם
  הכנס:
  Other במידה וקיים למודם רק מצב תקשורת
  <Name> במידה וקיימים לו שני מצבים: אחסון ותקשורת
  (ויש קובץ קונפיגורציה מתאים עבור המודם בהתאם לשלב 2)
- **Mail**- כתובת מייל לשליחת הודעות שגיאה
- **update period**- מספר קריאות שלאחריהן ישלח עדכון לשרת
  (רצוי להגדיר 15 הסבר ניתן למצוא: שיקולים להגדרת update period=15)
- **save period** - מספר הימים עבורם ישמרו קבצי הגיבוי (למשל 30)
- **backup folder**- (path מלא עבור תיקיית גיבוי הקבצים)
- **sensor type**- סוג החיישן (כרגע רק dylos)
- **sensor serial name**- שם החיבור הסריאלי של החיישן בהתאם לשלב 3.

בנוסף, יש לאשר/לבטל את הרצת התוכנית הראשית עם העלאת המערכת.

הערה: במידה ולא יאושר/יבוטל ניתן יהיה להריץ את התוכנית ע"י הפקודה:

python DylosToSerial.py

אך במקרה שהמערכת תופעל מחדש הרצת התוכנית לא תחודש.

6. הוצאת המערכת לשטח

עם העלאת המערכת בשטח התוכנית הראשית תתחיל לרוץ ברקע.
ניתן לעצור את התוכנית בהתאם לסעיף 7 ב-כדאי לדעת.

עבור פרטים נוספים בנוגע לריצת קובץ הקונפיגורציה ניתן לעבור על:
תיעוד הקובץ DylosConf.py

תיעוד הקובץ DylosConf.py

מטרת הסקריפט: לאתחל/לשנות הגדרות בטרם הוצאת המערכת לשטח.

הסקריפט יוצר את הקבצים:

1. DylosConf.txt- המכיל הגדרות כלליות לריצת המערכת הנשמרות ב-**USER_DATA** .
2. ServerConf.txt- המכיל את הגדרות החיבור לשרת הנשמרות ב-**SERVER_CONF**.
3. SensorBasicConf.txt- המכיל הגדרות עבור החיישן בלבד הנשמרות ב-**SENSOR_DATA** .
   (נקבעים כתוצאה מהגדרת המשתנה sensor_type כ-dylos ותומכים בקריאת הנתונים מהחיישן לRP)
4. DylosDesc.txt- המכיל פירוט באשר לשדות **USER_DATA**.

השדות ב-USER_DATA : (פירוט נוסף בקובץ DylosDesc.txt)

ישנם **משתנים שחובה להגדירם** לפני חיבור המערכת
(ניתן לראות כי מוגדרים כdefault באתחול הUSER_DATA) הגדרות אלו מסומנות באדום

- **ID** - מספר החיישן
- **Network**- רשת אלחוטית (GSM network: Orange, Cellcom, Pelephone, Hot, Golan, 012)
- **Modem**- שם המודם
  Other במידה וקיים למודם רק מצב תקשורת
  <Name> במידה וקיימים לו שני מצבים: אחסון ותקשורת
  (יש ליצור קובץ קונפיגורציה מתאים בהתאם למדריך בשם <name>.conf ולשמור ב- /ome/pi/Dylos
- **Mail**- כתובת מייל לשליחת הודעות שגיאה
- **update period**- מספר קריאות שלאחריהן ישלח עדכון לשרת (למשל 15)
- **save period**- מספר הימים עבורם ישמרו קבצי הגיבוי (למשל 30)
- **backup folder**- (path עבור תיקיית גיבוי הקבצים)
- **sensor type**- סוג החיישן (כרגע רק dylos)
- **sensor serial name**- שם החיבור הסריאלי של החיישן.

ניתן להשיג את שם החיישן בצורה הבאה:
יש לחבר את החיישן ולאחר מכן לפתוח טרמינל ולהכניס את הפקודה lsusb כעת לחפש את שם החיבור הסריאלי. ניתן להכניס את השם המלא או חלקו (בתנאי שיהיה יחודי)

```
pi@raspberrypi ~/Dylos $ lsusb
Bus 001 Device 002: ID 0424:9514 Standard Microsystems Corp.
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 003: ID 0424:ec00 Standard Microsystems Corp.
Bus 001 Device 012: ID 067b:2303 Prolific Technology, Inc. PL2303 Serial Port
Bus 001 Device 009: ID 1bbb:0000 T & A Mobile Phones
Bus 001 Device 004: ID 046d:c52e Logitech, Inc.
```

השדות ב- SERVER_CONF:
(נכון לעכשיו מוגדרים פרטי ההתחברות שנמסרו ע"י שחר והם ניתנים לשינוי בריצת הסקריפט)

- **user**- שם משתמש
- **password**- סיסמא
- **host**- IP של שרת הSQL
- **database**- שם ה-data base אליו כותבים

בנוסף מוגדרים:
- **conf_location**- path עבור הקובץ DylosConf.txt
- **sensor_basic_conf**- path עבור הקובץ SensorBasicConf.txt
- **dylos_reader**- path עבור הקובץ DylosToSerial.py (הסקריפט הראשי)
- **server_conf**-path עבור הקובץ ServerConf.txt

(אלו **אינם** ניתנים לשינוי במסגרת הסקריפט ויש לשנות את הסקריפט עצמו לצורך שינויים)

פעולת הסקריפט:

1. מחיקת קובץ rebootFlag הנוצר בריצת התוכנית הראשית לאחר פעולת reboot יזומה (מהווה דגל).

2. פתיחת הקובץ DylosConf.txt אם קיים והעברת תוכנו ל-**USER_DATA** או יצירתו במידה ולא קיים והגדרת ערכי **USER_DATA** לפי ערכי ברירת המחדל.

3. פתיחת הקובץ ServerConf.txt אם קיים והעברת תוכנו ל-**SERVER_CONF** או יצירתו במידה ולא קיים והגדרת ערכי **SERVER_CONF** לפי ערכי ברירת המחדל.

4. מעבר על שדות **USER_DATA** ויצירת קובץ הפירוט DylosDesc.txt.

5. מעבר על שדות **SERVER_CONF** ועדכונם/הגדרתם לפי קלט מהמשתמש.

6. מעבר על שדות **USER_DATA** ועדכונם/הגדרתם לפי קלט מהמשתמש.

7. יצירת פקודת updateInternetStr וביצועה.
הפקודה מעדכנת את קובץ הקונפיגורציה wvdial בהתאם לרשת הסלולרית שנבחרה.

8. העתקת קובץ הקונפיגורציה של המודם (במידה ואינו other) שסופק למיקום המתאים על מנת שפקודת usb_modeswitch
(המעבירה ממצב אחסון למצב תקשורת) תתבצע בהצלחה.
(במידה ולא קיים למודם קובץ קונפיגורציה תואם תודפס שגיאה והתוכנית תסתיים)

9. הגדרת **SENSOR_DATA** ע"י קבועים במידה והחיישן מסוג dylos
(במקרה של הוספת סוגי חיישנים נוספים יש לעדכן בשלב זה קונפיגורציה עבורם)

10. הדפסת ההגדרות החדשות, פרטי **USER_DATA** ו-**SERVER_CONF**.

11. המשתמש ישאל האם להריץ את התוכנית הראשית בהעלאת המערכת בהתאם לתשובתו יערך הקובץ /etc/rc.local.

תיעוד הקובץ setting.py

מכיל הגדרות לצורך שימוש בתוכנית הראשית (DylosToSerial.py המבצע לו import):

- תבניות להדפסת הודעות שגיאה

- **<u>הגדרות קבועים:</u>**

- **FIRST_ERROR_ID**=1: מזהה שגיאת קריאה ריקה מהחיישן.
- **FIRST_ERROR_MSG**="NO_AVAILABLE_DATA":
הודעת שגיאת קריאה ריקה.
- **SECOND_ERROR_ID**=2: מזהה שגיאת קריאה לא תקינה מהחיישן.
- **SECOND_ERROR_MSG**="INVALID_DATA":
הודעת שגיאת קריאה לא תקינה.
- **MAIN_TABLE**=DYLOS_READING :
שם טבלת קריאות חיישני הdylos בשרת.
- **ERROR_TABLE**=DYLOS_FAILURE: שם טבלת השגיאות בשרת.
- **SERIAL_CONNECT_PERIOD** = 10:
פרק הזמן בשניות בין ניסיון מציאת חיבור סריאלי.
-
- **SLEEP_BETWEEN_READS** = 40:
פרק הזמן בשניות בו התוכנית ממתינה עד לקריאה הבאה במידה ולא מתבצעת שליחת נתונים לשרת.
- **MINUTE** = 60: שניות בדקה.
- **DAY** = 24: שעות ביממה.
- **FIRST_REBOOT_TIME** = 3: פרק הזמן בשעות שלאחריו במקרה בו אין חיבור לאינטרנט תתבצע פעולת reboot יזומה (לאחר הפעם הראשונה הפעולה תתבצע כל 24 שעות).
- **SEND_MAIL_PERIOD** = 13: פרק הזמן בשעות (+1 – כלומר 12) לשליחת מייל במקרה של בעיות עם השרת.
- **CONNECTION_TIMEOUT** = 5: פרק הזמן המקסימלי בשניות לניסיון יצירת קשר עם השרת.
- **RESTORE_FILE** = '/home/pi/Dylos/restore.txt': כתובת הקובץ שיאחסן את המידע במקרה של פעולת reboot יזומה
- **REBOOT_FLAG** = '/home/pi/Dylos/rebootFlag': כתובת הקובץ שישמש כדגל שבוצע reboot יזום ראשוני

תיעוד הקובץ DylosToSerial.py

התוכנית הראשית.

מטרת הסקריפט: קריאת נתונים מהחיישן כל 60 שניות ושליחת נתונים לשרת ע"י תקשורת סלולרית לאחר update_period של קריאות (אשר מוגדר בקובץ הקונפיגורציה- default-כ 15).
בנוסף נוצרים קבצי גיבוי וטיפול בשגיאות השונות.

הפונקציות:

-import_configuration(file_name)

קלט- שם של קובץ הקונפיגורציה (לא כתובת מלאה)
פלט- הגדרות הקונפיגורציה הנמצאות בקובץ ששמו התקבל בקלט או [ ] במידה והקובץ אינו קיים.

-find_available_serial_ports()

קלט-אין
פלט- שם של serial_port עבור החיישן או [ ] במקרה של שגיאה.
במקרה של שגיאה יודפס: "No serial port available"

-connectToInternet()

קלט ופלט- אין
הפונקציה יוצרת חיבור לתקשורת סלולרית.
אם מדובר במודם בעל שני מצבים (אחסון ותקשורת, לא other) משנה את המודם למצב תקשורת ע"י קובץ הקונפיגורציה של המודם וממתינה 5 שניות על מנת שהשינוי יתבצע כראוי.
ביצוע פקודת החיבור לאינטרנט ברקע .

-open_serial_connection(serial_port=str)

קלט- שם ה-serial port
פלט- אין אך המשתנה הגלובלי ser מתעדכן.
יצירת תקשורת בין החיישן לRP
המשתנה ser מתעדכן לfalse רק במידה והחיבור נכשל.
במקרה של הצלחה יודפס: "The serial connection is now open"
במקרה של שגיאה יודפס: "The serial connection has failed"

-create_file(folder,date)

קלט- תיקייה בה ישמר הקובץ ותאריך הקובץ
פלט- שם הקובץ
יוצרת את התיקייה והקובץ במידה ואינם קיימים.
שם הקובץ נקבע לפי תבנית בהתאם לתאריך בקלט ומזינה את כותרות העמודות בהתאם לקובץ הקונפיגורציה (SensorBasicConf.txt).

-restoreData()

פלט- רשימה המכילה את כל המידע השמור בקובץ האחזור או רשימה ריקה במידה ולא קיים.
בודקת אם קיים קובץ אחזור לפי הגדרת RESTORE_FILE (בקובץ setting.py)
במידה וקיים, שופכת את המידע לרשימה ומוחקת את הקובץ.

-removeOldFiles(backupFolder, daysAgo, today)

קלט- path של תיקיית הגיבוי, מספר ימים שקבצים בעלי ותק גדול מהם ימחקו, תאריך של היום.
פלט- אין
הפונקציה מוחקת את כל הקבצים מתיקיית הגיבוי בעלי ותק הגדול מ-daysAgo.

-write_data()

קלט ופלט- אין
בדיקה שהחיבור הסריאלי פתוח הדפסת הודעה בהתאם.
בקשת קריאה מהחיישן.

-request_data()

פלט- שורת קריאה מהחיישן
בקשת קריאה מהחיישן ע"י הפונקציה write_data() קריאת מידע מהחיישן.
הפרדת המידע מתווים לא רצויים והפרדתו בפסיקים.
במקרה של שגיאה תחזור שורת קריאה ריקה.

-write_data_to_csv(filename,dylos_data)

קלט- שם הקובץ, ושורת הקריאה
פלט- שורת הקריאה הערוכה
הוספת מידע נוסף לשורת הקריאה: חתימת הזמן ו-id עבור החיישן וכתיבת השורה לקובץ הcsv.
במקרה של קריאה לא תקינה יודפס:
עבור קריאה ריקה: "the data was null!, Dylos is disconnected or off"
ובמדדים יופיע none.
עבור קריאה שאינה תקינה: "there was an error!, maybe the cable is unplugged"
ובמדדים יופיע error.

-sendMail()

קלט ופלט- אין
מגדירה את פרוטוקול SMTP בהתאם לחשבון המייל שפתחנו ב-gmail (קבוע בקוד אין צורך לשנות משמש לשליחת מייל בלבד).
יצירת אובייקט טקסט המתריע על בעיה בחיבור לשרת ושליחתו למייל המוגדר בקובץ הקונפיגורציה.

-storeData(list)

קלט- רשימה של קריאות לאחסון
יוצר קובץ בשם RESTORE_FILE (המוגדר בקובץ setting.py) וכותב אליו את שורות הקריאה מהרשימה.

-handleNoConnection(tmpList)

קלט- רשימת הקריאות שטרם נשלחו
פלט- 0 אם יש בעיה בחיבור לשרת, 512 אם יש בעיה בחיבור לאינטרנט

בדיקת תקשורת לאינטרנט ע"י ביצוע ping ל-google.
במידה והתקבל pong (יש תקשורת אינטרנט ויש בעיה בחיבור לשרת):

- יאופס המשתנה DISCONNECT_TIME (המחזיק את חותמת הזמן בו התגלה כי אין חיבור לאינטרנט)
- יעודכן המשתנה SEND_MAIL (המחזיק את מספר השעות שעברו מאז נשלח מייל)
- במידה וטרם נשלח מייל או שחלפו SEND_MAIL_PERIOD (המוגדר בsetting.py) שעות מאז שליחת המייל האחרון ישלח מייל המודיע על בעיה בחיבור לשרת לכתובת המוגדרת בקובץ הקונפיגורציה ע"י קריאה לפונקציה sendMail()
- אחרת, (לא התקבל pong- אין תקשורת אינטרנט או ש-google קרס ☺)
- יאופס המשתנה SEND_MAIL
- במידה וקיים הקובץ REBOOT_FLAG וגם עברו 24 שעות (לפי המוגדר בDAY) או שעברו FIRST_REBOOT_TIME שעות בצע:
  - שמירה של הקריאות שטרם נשלחו ע"י קריאה לפונקציה storeData()
  - Reboot
- אחרת, בצע ניסיון נוסף להתחבר לאינטרנט ע"י קריאה לפונקציה connectToInternet()

-connectToServer(config, tmpList)

קלט- הגדרות החיבור לשרת, רשימת הקריאות שטרם נשלחו
פלט- המילה Connect במידה ונוצר חיבור אחרת 0 או 512

ניסיון חיבור לשרת הSQL ע"י פונקציות מערכת של python.
במידה ונכשל קריאה לפונקציה handleNoConnection().
במידה והצליח יעודכנו המשתנים הגלובליים SEND_MAIL ו-DISCONNECT_TIME (יאופסו)

-createAndSendQuery(tmpList)

קלט- רשימת הקריאות שטרם נשלחו
פלט- חותמת הזמן ביציאה מהפונקציה

- יצירת אובייקט ביצוע (curser) המשמש ליצירת קשר עם השרת.
- אתחול שלוש רשימות:
  backupData– גיבוי הרשימה המלאה
  readerData- עבור הקריאות התקינות
  failureData- עבור קריאות לא תקינות
- נגדיר במשתנה size את המספר המינימלי בין מספר הקריאות שטרם נשלחו לבין update period.

- ונבצע size פעמים:
  - הוצאת קריאה מרשימת הקלט שמירתה ברשימה הגיבוי וסיווגה באופן הבא:
    - אם השדה האחרון בשורת הקריאה הינו מספר הקריאה תקינה
    - אם השדה האחרון הינו "none" הקריאה אינה תקינה ותקבל את השגיאה "NO AVAILABLE DATA" שמספרה 1.
    - אם השדה האחרון הינו "error" הקריאה אינה תקינה ותקבל את השגיאה "INVALIDE DATA" שמספרה 2.
  - בהתאם לסיווג הקריאה תוכנס לרשימה המתאימה (readerData או failureData).
- עבור הרשימות readerData ו- failureData במידה ויש בהן מידע ניצור שאילתה מתאימה לשליחת הקריאות בהן ונשלח אותה לביצוע.
- אם נכשלה שליחת השאילתה נבצע שחזור של הרשימה באמצעות רשימת הגיבוי ונצא מהפונקציה.

-sendData(tmpList,config)

קלט- רשימת הקריאות שטרם נשלחו, הגדרות החיבור לשרת
פלט- אין

דגימת חתימת זמן בעת הכניסה לפונקציה.
כל עוד קיימות קריאות ברשימה וזמן השהות בפונקציה לא חרג מזמן ההמתנה SLEEP_BETWEEN_READS (המוגדר בsetting.py) בצע:
- במידה ולא קיים חיבור לשרת יצירת חיבור ע"י קריאה לפונקציה connectToServer().
- אם אין חיבור לאינטרנט או תגובה מהשרת יציאה מהפונקציה
- שליחת update period (המוגדר בקובץ הקונפיגורציה) של קריאות לשרת ע"י קריאה לפונקציה createAndSendQuery()

בצע המתנה לפרק זמן הנותר מ-SLEEP_BETWEEN_READS

<u>פעולת הסקריפט:</u>

1. יבא את ההגדרות מקבצי הקונפיגורציה ע"י קריאות לפונקציה import_configuration().
2. קריאה לפונקציה find_available_serial_ports() המוצאת את ההתקן הסריאלי של החיישן.
3. קריאה לפונקציה connectToInternet() - היוצרת חיבור לאינטרנט ע"י תקשורת סלולרית.
4. בדיקה כי אכן נמצא בשלב 2 התקן סריאלי במידה ולא ממשיכה לנסות כל SERIAL_CONNECT_PERIOD (המוגדר בקובץ setting.py)
5. קריאה לפונקציה open_serial_connection() הפותחת ערוץ תקשורת עם החיישן.
6. מציאת תאריך של היום הנוכחי (ע"י פונקציות מערכת מהספרייה datetime).
7. קריאה לפונקציה create_file() היוצרת קובץ excel (csv) יומי (לגיבוי נתונים) בתיקיית הגיבוי שהוגדרה בקובץ הקונפיגורציה.
8. קריאה לפונקציה restoreData() - המאחזרת מידע במידה ובוצעה פעולת reboot יזומה.

9. לולאה (אינסופית) המבצעת:
   9.1 במידה וחלף יום:
      - יצירת קובץ גיבוי עם התאריך החדש ע"י קריאה לפונקציה **create_file()**
      - קריאה לפונקציה **removeOldFiles()** – המוחקת מתיקיית הגיבוי קבצים ישנים בהתאם ל-save period המוגדר בקובץ הקונפיגורציה.
   9.2 בדיקה האם ההתקן הסריאלי מחובר
   9.3 קריאה לפונקציה **request_data()**- המחזירה שורת קריאה מהחיישן.
   9.4 קריאה לפונקציה **write_data_to_csv()**- עריכת שורת הקריאה מהחיישן לשליחה לשרת וכתיבתה לקובץ הגיבוי היומי (csv)
   9.5 במידה ומספר הקריאות שטרם נשלחו גדול מ-update period (המוגדר בקובץ הקונפיגורציה)
      - קריאה לפונקציה **sendData()**- שליחת הקריאות שנצברו לשרת
   9.6 אחרת, המתן SLEEP_BETWEEN_READS שניות.

סיכום פגישה עם מר איתי דברן

מסקנות:

- בפיתון החיבור לרשת נעשה על גבי פרוטוקול FTP מעל TCP.

- קיים פרוטוקול נוסף TFTP מעל UDP.

- עמדנו על ההבדלים בין הפרוטוקולים השונים ובאופן כללי ההבדל העיקרי הינו שפרוטוקול UDP אומנם מהיר יותר אך אינו מבטיח בדיקת שגיאות ושמירה על סדר העברת הנתונים בעוד שפרוטוקול TCP מעט איטי יותר (לא באופן משמעותי) אך מבטיח תקינות העברת מידע ושמירה על סדר ההעברה.
בעצה אחת עם איתי דברן ובהתאם לצרכינו הגענו למסקנה כי פרוטוקול TCP מתאים יותר.

- כמות העברת מידע בשאילתה:
לפי מה שהבנו מאיתי בכל packet ניתן להעביר כ400 תווים.
מספר התווים בשאילתה ממוצעת בשליחת שורת קריאה בודדת הינה כ35 תווים.
על כן, עבור שליחת שורה בודדת נקבל overhead גדול ולכן הוחלט להעביר 15 שורות קריאה בכל שאילתה.
בכך נקטין את ה-overhead מחד ומאידך נשמור על תקינות העברת המידע.

- בנוסף, שליחת המידע לשרת באמצעות פיתון במקרה של חריגה מגודל packet בודד תדע לחלק את המידע לכמות packet מתאימה בצורה אופטימלית ותקינה.