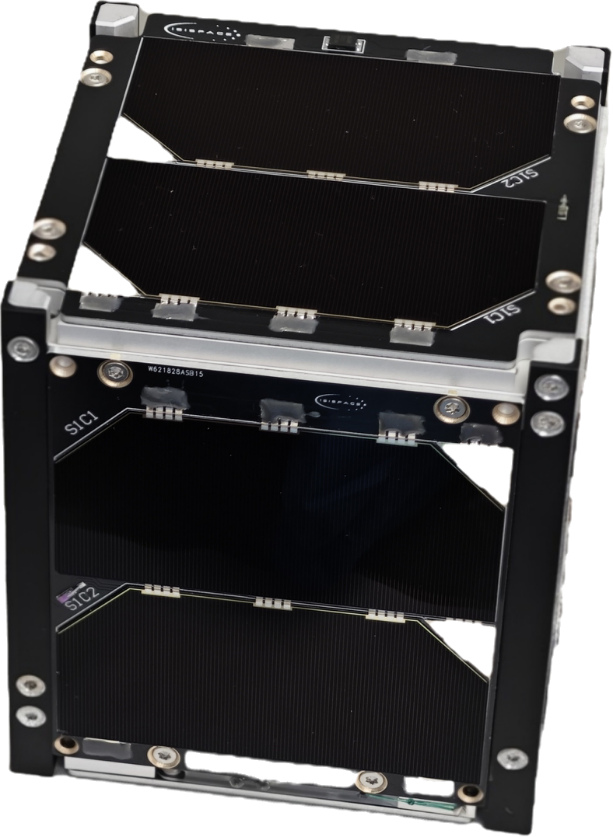
**מסמך אפיון פרויקט גמר בסייבר**

מגישה מעיין כרמי  
ת"ז: 331826966

תיכון הנדסאים הרצליה

A planet earth with a broken face

AI-generated content may be incorrect. 

**HoopoeWorld פיניקס**

# תוכן עניינים

תוכן עניינים

[תוכן עניינים 1](#_Toc218544766)

[פתיחה 3](#_Toc218544767)

[תוכנה מוטסת ללוויין מסמך אפיון 4](#_Toc218544768)

[ראשי תיבות 5](#_Toc218544769)

[מבוא: 6](#_Toc218544770)

[תיאור כללי של הפרויקט 6](#_Toc218544771)

[קהל היעד 6](#_Toc218544772)

[סיבות לבחירה בפרויקט 6](#_Toc218544773)

[מטרות הפרויקט 6](#_Toc218544774)

[ייחודיות וחדשנות 6](#_Toc218544775)

[מגבלות ופיתוח עתידי 7](#_Toc218544776)

[היבטים טכניים 7](#_Toc218544777)

[בדיקות המערכת 8](#_Toc218544778)

[גנט 9](#_Toc218544779)

[פירוט יכולות התוכנה: 10](#_Toc218544780)

[ארכיטקטורת הפרויקט: 12](#_Toc218544781)

[פירוט החומרה 12](#_Toc218544782)

[תיאור טכנולוגי 13](#_Toc218544783)

[תיאור זרימת המידע 13](#_Toc218544784)

[תיאור אלגוריתמים 15](#_Toc218544785)

[תיאור סביבת הפיתוח 17](#_Toc218544786)

[פרוטוקול התקשורת 18](#_Toc218544787)

[תיאור מבני נתונים 27](#_Toc218544788)

[סקירת חולשות 27](#_Toc218544789)

[קוד HoopoeWorld מסמך אפיון 28](#_Toc218544790)

[מבוא: 29](#_Toc218544791)

[תיאור כללי של הפרויקט 29](#_Toc218544792)

[קהל היעד 29](#_Toc218544793)

[סיבות לבחירה בפרויקט 29](#_Toc218544794)

[מטרות הפרויקט 29](#_Toc218544795)

[ייחודיות וחדשנות 30](#_Toc218544796)

[מגבלות ופיתוח עתידי 30](#_Toc218544797)

[היבטים טכניים 30](#_Toc218544798)

[בדיקות המערכת 30](#_Toc218544799)

[גנט – תוכנת ה-HoopoeWorld 31](#_Toc218544800)

[פירוט יכולות התוכנה: 32](#_Toc218544801)

[צד לקוח: 32](#_Toc218544802)

[צד שרת: 32](#_Toc218544803)

[ארכיטקטורת הפרויקט: 34](#_Toc218544804)

[פירוט חומרה 34](#_Toc218544805)

[תיאור טכנולוגי 34](#_Toc218544806)

[תיאור זרימת המידע 34](#_Toc218544807)

[תיאור אלגוריתמים 34](#_Toc218544808)

[פרוטוקול תקשורת 36](#_Toc218544809)

[תיאור מסכי המערכת 39](#_Toc218544810)

[תיאור מבני הנתונים 39](#_Toc218544811)

[סקירת חולשות ואיומים 39](#_Toc218544812)

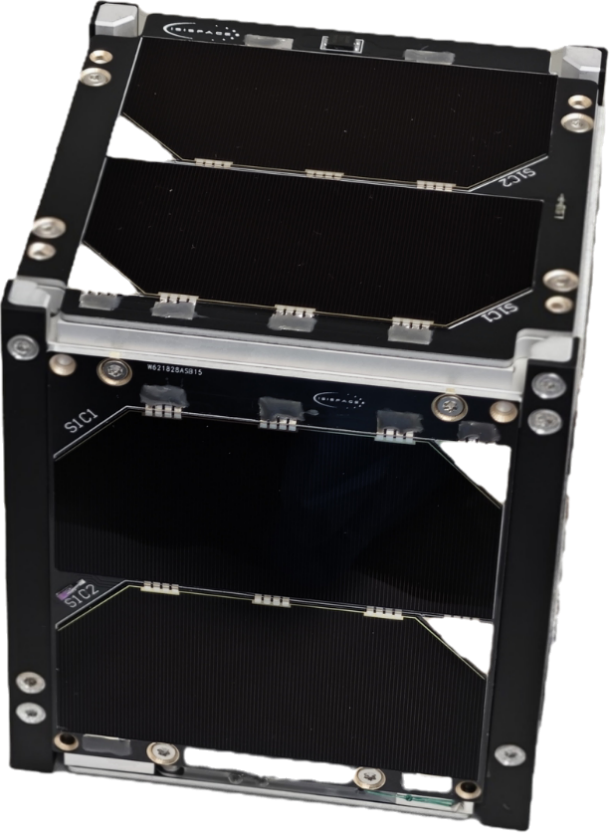
# פתיחה

במסמך הזה יש 2 פרויקטים.

1. קוד תוכנה מוטסת של לוויין
2. תוכנה לתחנת קרקע שנקראת HoopoeWorld

חילקתי את זה כך שיש מסמך אפיון לכל אחד אחד אחרי השני. המלצה שלי לקרוא קודם את ההסבר על התוכנה המוטסת מכיוון שאז ההסבר על התוכנה שמשתמשת בנתונים שנקלטים מהלוויין תשמע יותר הגיוני.

# **תוכנה מוטסת ללוויין מסמך אפיון**



## ראשי תיבות

DPS – Down power safe mode

UC – Up to cruise

DC – Down to cruise

UO – Up to operational

VHF – Very high frequency (130 – 170 MHz)

UHF – Ultra high frequency (400 – 450 MHz)

TRXVU – Transceiver, UHF Transmitter and VHF Receiver

EPS – Electric power system

OBC – On board computer

I2C – Inter integrated circuit communication bus.

ANTS – Antennas System

AX.25 – Amateur X.25

FS – File system

RTC – Real time clock

RFCB – Radio frequency communication branch

SP – Solar Panels

## מבוא:

### תיאור כללי של הפרויקט

הפרויקט הוא תוכנה מוטסת ללוויין. הלוויין היינו לוויין מסוג ננו לוויין בגודל 1CubeSat שזו יחידת מידה של 10cm\*10cm\*10.8cm.

התוכנה כתובה בשפת C ויועדה לשימוש של המרכז המדעים שלנו לניהול ולחקר.

### קהל היעד

קהל היעד העיקרי הוא:

1. **האנשים בתוכנית שלי** (וגם אלו שיצטרפו אליה) – שיוכלו להשתמש בלוויין שכתבתי לו את הקוד, וגם בקוד הלוויין שהוספתי לו הערות כדי שיהיה אפשר לשפר אותו ולהוסיף לו דברים אחרי בלי הצורך להתחיל את הכל מחדש כמו שקרה עד כה.
2. **חובבי רדיו מרחבי העולם** – שיוכלו להשתמש במשיב שעל הלוויין, שהוא אמצעי שמעביר מידע דרך הלוויין לנקודות שונות ברחבי העולם, ובכך יוכלו לתקשר דרכו.

### סיבות לבחירה בפרויקט

אני חושבת שיותר נכון לשאול למה הצטרפתי לתוכנית מכיוון שכבר שהצטרפתי ידעתי שאני ככל הנראה יהיה מפתחת של לוויין. במהלך התוכנית גם ראיתי את האהבה שלי לקוד של מערכת זאת אומרת לא במדויק ללוויין אבל בכללי לתכנת מוצר מאשר תוכנה.

עוד אחת הסיבות שהיא תהיה נכונה גם תוכנית השנייה היא שאני אוהבת ליצור דברים שבאמת ישתמשו בהם ויש בהם צורך. ואני יודעת שבלוויין ישתמשו גם אם לא אנחנו בתוכנית יהיה מקום שישתמש בו.

### מטרות הפרויקט

* חקר החלל והקרינה.
* מתן שירות לחובבי הרדיו ברחבי העולם.
* חקר רכיבי הלוויין, התוכנה ותפקודם בפועל.

### ייחודיות וחדשנות

בעוד שתוכנת לוויין כבר קיימת הייתי צריכה ליצור חדשה עבור הלוויין שלנו.

במהלך התקופה לא הייתה תוכנה ברורה שמתאימה בדיוק ללוויינים שלנו אז יצרתי אחת עבורם. התוכנה הזאת נמצאת על 4 מתוך 9 לוויינים שכרגע נמצאים בחלל.

לקחתי השראה מלוגיקה של לוויינים קודמים של התוכנית ומהקוד שסופק עם הדרייברים (שמסביר איך להשתמש בהם) אבל חלק מהלוגיקה גם ראיתי דרך לשפר אותה אז שיניתי אותה שתהיה למשהו יותר מתאים ויעיל.

### מגבלות ופיתוח עתידי

תוכנת הלוויין מאוד מוגבלת לחומרה הספציפית שלו וכאשר נעבור חומרה נצטרך לשנות חלק מהקוד בין אם ה-drivers שמשתמשים בהם או חלק מהלוגיקה כמו משימת הלוויין שמשתנה. אבל רוב הלוגיקה תישאר אותו הדבר רק צריך להבין איך לשנות את התוכנה כדי להתאים.

זוהי תוכנה שהיא embedded. צריך שתהיה אפשרות לצרוב אותה כדי שתוכל לרוץ כל הזמן וגם היא בסופו של דבר מנהלת את הלוויין.

השינויים בקוד לא חייבים להיות דרסטיים וכתבתי מספיק הערות כדי שאנשים בהמשך ידעו מאיפה להמשיך את מה שעשיתי ויוכלו להבין מה עשיתי. ובו בזמן אני כבר מלמדת את הדור המשך של התוכנית את התוכנה כך שהם ידעו איך להמשיך אותה גם אם לא היו הערות (אבל תמיד טוב שיהיו).

### היבטים טכניים

הפרויקט של הלוויין:

* פיתוח חלק ממערכת הפעלה בזמן אמת (Real-Time) בשפת C עבור הלוויין (התוכנה צרובה עליו).
* פיתוח מערכת הקבצים של הלוויין (וגם כל שאר המערכות).
* כתיבה ויישום של פרוטוקול תקשורת ייעודי.

לא כולל (אך נכלל בתוכנה השנייה):

* עבודה עם מבני נתונים
* שימוש בפרוטוקול תקשורת מוצפן – התקשורת של הלוויין לא יכולה להיות מוצפנת מכיוון שמתוך רישיון חובבות הרדיו אסור לנו לשדר בתדרים שלהם באופן מוצפן.
* מערכת שרת ולקוח – הלוויין הוא יותר גוף שעובד עצמאית אבל הוא גם סוג של שרת שתחנות קרקע ברחבי העולם יוכלו לקבל ממנו נתונים אבל רק מי שאם הפיענוח הנכון יוכל לדעת מה כתוב ומה לשלוח מכיוון שהתקשורת ב-Hexa.

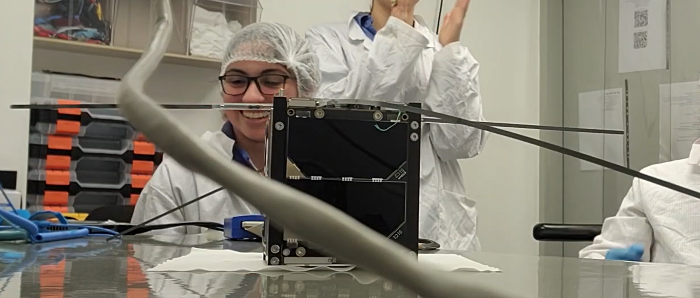
### בדיקות המערכת

מערכת הלוויין כבר מזמן יצאה מאזור שאפשר לבדוק ולתקן את הלוויין אבל אפשר לתקן את התוכנה לפעם הבאה וזה כבר בתהליך או נעשה.

הבדיקות לפני השיגור כללו:

* הרצה של כל הפקודות של הלוויין.
* הרצת בדיקות קצה של פקודות לוויין.
* בדיקה של ריצה של הלוויין.
* בדיקת ניהול הקבצים וניהול מחיקתם.
* פריסת אנטנות – אחד מהחשובים ביותר. אם לא עובד הלוויין לא יוכל לשדר.
* לתת לו לרוץ כמות רבה של זמן בלי הפרעות
* בדיקת התממשקות עם התוכנה שלנו בתחנת הקרקע.
* בדיקת מעבר דרייברים. כל הקוד נבדק על הדרייברים של הלוויין הפרוס אך כאשר הגיע הלוויין הייתי צריכה לשנות את הקוד ולוודא שהכל עדיין עובד.

היו עוד בדיקות שהייתי רוצה לעשות אבל תאריך השיגור כבר הגיע אז הייתי צריכה לשלוח אותו אם רק בדיקות מינימום אפילו שהייתי רוצה יותר זמן. עבור בדיקות נוספות.

אלו שתי תמונות מסרטון שצילמנו בעת פריסת האנטנות.

כמו שאפשר לראות זאת הייתה הקלה מאוד גדולה כאשר זה עבד ועבר חלק מכיוון שזהו החלק שהכי מפחיד למתכנת כאשר הוא מתכנת לוויין.

האנטנות מאוד עדינות ואין לנו ספייר, גם ממה ששומר עליהם היה מספיק רק ל-2 פריסות – אחת במעבדה ואחת בחלל. אז אם זה לא היה עובד זאת הייתה יכולה להיות קטסטרופה.

### גנט

עבר זמן מאז שעשיתי את פרויקט זה אז אני כותבת את ההליך אליו. עד קבלת הלוויין בערך עבדתי עם דרייברים ישנים שעובדים על הלוויין שיש לנו במעבדה אבל הוא לא הלוויין שעלה לחלל. רק אחרי קבלת הלוויין (או בערך שבועיים לפני קבלתו) יכולתי להחליף את הדרייברים.

הגנט מורכב מבערך מה שהיה בפועל, אבל בעיקרון תמיד היו בדיקות לקוד וכשנמצאו שגיאות תיקנתי אותם. התוכנה לא הייתה גמורה עד שהטסנו את הלוויין לארה"ב מכיוון שהיה מידע חדש שקיבלנו על תקלה ברכיב אז ניסיתי לשנות את הקוד כדי שיתאים לבעיה. וכמובן עד הרגע האחרון (לפני הטענת הבטריות) בדקתי את קוד הלוויין.

|  |  |
| --- | --- |
| **שלב** | **תאריך – סיום** |
| לוגיקת לוויין (הבנה וכתיבה) | דצמבר 2023 |
| הבנת כתיבת תוכנת לוויין (גם C נכלל פה אפילו שנלמד גם תוך כדי) | פברואר 2024 |
| כתיבת demo – קוד בדיקות ללוויין שעוזר להבין איך לכתוב לוויין | יוני 2024 |
| כתיבת תוכנת ה-Init System (פעולות האיתחול וקוד הפריסה של האנטנות – בלי לבדוק את הפריסה) TRXVU (מערכת התקשורת) עם הדרייברים הישנים (כולל פקודות ה-TRXVU) | אמצע ספטמבר 2024 |
| Management – קוד של ניהול כללי של הלוויין | אמצע אוקטובר 2024 |
| EPS – מערכת החשמל | תחילת נובמבר 2024 |
| קבלת הלוויין והתחלת החלפת דרייברים | תחילת דצמבר 2024 |
| File System ו-Payload | סוף דצמבר 2024 |
| תיקונים אחרונים ופריסת אנטנות | תחילת ינואר 2025 |

## פירוט יכולות התוכנה:

הלוויין הוא “שרת” שניתן *“להתחבר אליו”* רק במובן הבא:

תחנת הקרקע משדרת פקודה, ואם הלוויין נמצא בקשר (מיקום נכון, אנטנה מופנית- (הוא קולט את האות, מבצע או דוחה פקודה, ומשדר בחזרה תגובה.

כל המערכות הפנימיות להלן כתובות בשפת C, לרוב תוך שימוש בדרייברים ייעודיים, enums, מבני נתונים

קוד הלוויין הוא רק צד שרת שאפשר "להתחבר" אליו באמצעות תחנות קרקע. למה להתחבר בגרשיים? מכיוון שזה יותר ששולחים מתחנת הקרקע ומקווים שהלוויין יקלוט (זאת אומרת שאנחנו עוקבים אחריו נכון) ונקבל תגובה שאנחנו הצלחנו לקלוט.

1. Beacon
   1. שידור בפרק זמן ידוע אות חיים המכיל מידע קריטי על הלוויין:
      1. מצב מערכת החשמל: מתח סוללה (mV), זרם, טמפרטורת תאים
      2. מידע כללי: הזמן בלוויין, מספר ריסטים יזומים, מספר ריסטים כוללים
      3. נתוני payload
      4. מידע תפעולי נוסף הנאסף ממודולים פנימיים
   2. נדרש:
      1. פונקציה שבודקת מעבר זמן
      2. איסוף נתונים מהדרייברים
      3. בניית Packet בהתאם לפרוטוקול שנקבע
      4. שידור דרך מודול התקשורת
   3. נחוץ:
      1. מבנה נתונים union שבתוכו מערך מסוג char ו-struct שמכיל את תוכן ה-beacon.
      2. דרייברים של קריאה מהמערכות ושליחה ב-TRXVU.
      3. הכרת המערכת (זה בהכול אחרת לא היה אפשרי לעשות את הדברים)
2. Transponder
   1. מצב שבו הלוויין הופך ל"מראת רדיו": תחנת אחרת → לוויין → תחנה רחוקה.

הלוויין מפעיל/מכבה את המשיב לפי פקודות שמתקבלות מהקרקע (פקודת הפעלה כוללת כמות זמן).

* 1. נדרש:
     1. קבלת פקודה “הפעל משיב” או “כבה משיב”
     2. במצב הפעלה: שמירת זמן פעולה (Duration)
     3. מנגנון Timeout שמכבה אוטומטית
  2. נחוץ:
     1. enum לכדי להחליף בין מצב משדר לבין מצב משיב.
     2. דרייברים ל-FRAM, ול-TRXVU
     3. תחנת קרקע פעילה

1. Payload
   1. חלקי מדידות שהלוויין אוסף לפי תזמונים שונים (כל 10 שניות אחד ו-15 דקות אחר) ונשמרים ונשלחים לניתוח מדעי. המדידות הן בקשר לקרינה.
   2. נדרש:
      1. יצירת קבצי payload במערכת הקבצים.
      2. קריאה תקופתית של חיישנים
      3. הפעלת דרייבר ייעודי שפותח עבור המשימה (TAU)
   3. נחוץ:
      1. union שבו אנחנו שומרים את המידע
      2. Timestamp – שנדע מתי נשמר
      3. דרייבר של payload.
2. פקודות לוויין
   1. רשימת פקודות קבועה מראש שהלוויין יודע לפרש. לכל מערכת בלוויין ישנה מספר פקודות שאפשר לבצע עבורה. חלק מהפקודות הן ניהול/בקרה, חלק קבלת מידע נוסף, חלק שינוי ערכי משתנים. לדוגמא ביצוע dump ששייך למערכת הקבצים, reset ששייך למערכת הניהול ועוד.
   2. נדרש:
      1. דרך לפרש את הפקודות
      2. מזהה פקודה (יש מספר enums לכך)
      3. switch case עבור כל פקודה
      4. בקרת שגיאת opcode ומערכת
   3. נחוץ:
      1. פונקציות ייעודיות
      2. תיעוד עבור משתמשי התחנה כדי שיוכלו להשתמש בלוויין.
3. מערכת הקבצים
   1. מערכת קבצים פנימית המבוססת FAT32 - דרייבר שמטפל בגישה לקבצים, יצירה, כתיבה ושמירה. התוכנה המוטסת של הלוויין בונה אוטומטית את שמות הקבצים בהתאם ללוגיקה המוגדרת ב-TLM\_management וגם שומרת, משנה ומוחקת את הקבצים בהתאם.
   2. נדרש:
      1. דרייבר FAT32
      2. פרוטוקול יצירת שמות קבצים וסיומות.
      3. ניהול טלמטריה
   3. נחוץ:
      1. מבני נתונים
      2. פונקציות מדרייבר FAT32
      3. תיעוד טוב
4. הפעלה עצמית
   1. מנגנון פנימי שמנתר מצב הלוויין (לדוגמה - חוסר תקשורת לאורך זמן) ומבצע אתחול אוטומטי לשחזור פעיל. אבל גם הפעלה כללית עצמאית של המערכות כך שהכל פועל גם אם לא מתקשרים לאורך זמן.

## ארכיטקטורת הפרויקט:

### פירוט החומרה

כל הרכיבים שאכתוב פה הם של חברה בשם ISISpace חוץ מה-payload שהוא של שורק

OBC – המחשב המוטס.

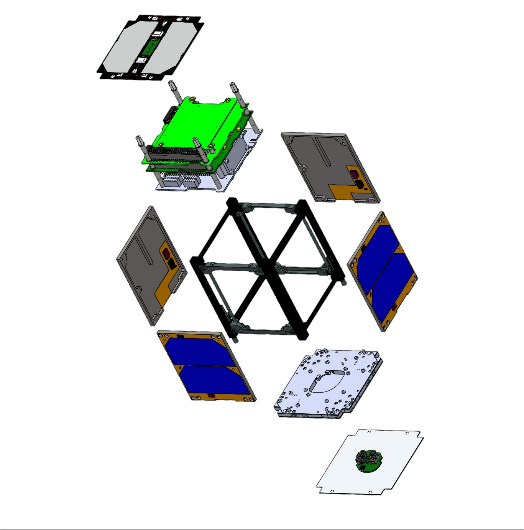
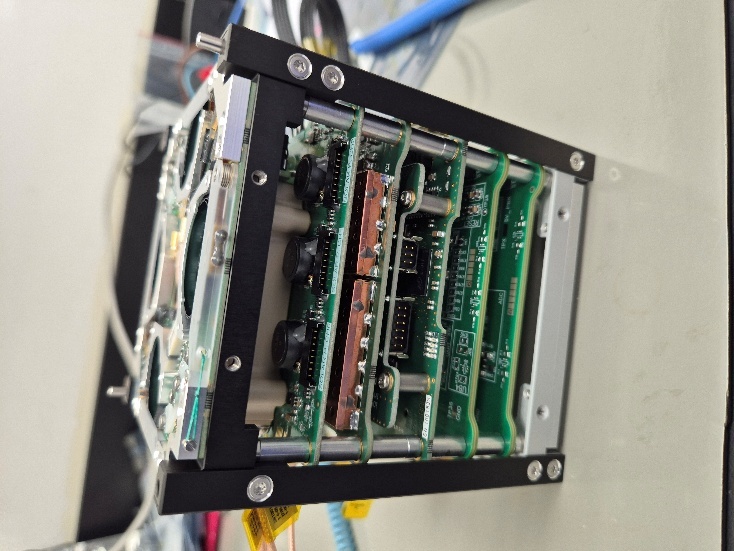
* מחשב בעל ליבה 1 במערכת 32bit.
* המערכת הפועלת עליה היא FreeRTOS והתוכנה שאני כותבת.
* יש עליו גם 2 כרטיסי SD של 2GB שעובד במערכת FAT32.
* חוץ מזה יש שם FRAM, RAM, NORFLASH, supervisor, RTC...
* חיבור שנקרא PC-104 שבאמצעותו מתחברים לרכיבים האחרים.
* לכאן אני כותבת את כל הקוד והוא מתחבר לרכיבים האחרים באחת מדרכי התקשורת (הרוב I2C) וזה מפעיל את הפקודה שאני כותבת מהפונקציות שנתונות.

EPS – מערכת החשמל

* המערכת כוללת בתוכה גם את בטריות
* חיבור לפאנלים הסולאריים (אני לרוב מחשיבה גם את הפאנלים כחלק מהמערכת אבל זה רכיב אחר)
* מנהלת את יציאת החשמל לרכיבים השונים.

TRXVU – מערכת התקשורת

* כוללת מקלט, משדר ומשיב (שהוא אותו הרכיב כמו המשדר).
* אני מחשיבה גם את האנטנות כחלק מהמערכת שכמו הפאנלים הם רכיבים שונים.
* האנטנות מחוברות ל-TRXVU בכבלים קואקסיאליים.

  A close-up of a computer

AI-generated content may be incorrect.

הקרדיט לשרטוט הולך בכולו לרותם אור שהיה עד שנה שעברה חלק מצוות מכניקה של התוכנית. (היינו כ-5 אנשים שהרכיבו את הלוויין שאחת מהן הייתה אני ומתכנתת אחת שהיא גם כן אני)

### תיאור טכנולוגי

* שפת תכנות הלוויין היא שפת C כמו שנאמר, אבל את הפקודות שלחתי ללוויין באמצעות Hexa.
* ה-OS כפי שכבר אמרתי הוא FreeRTOS אבל הקוד שכתבתי משתלב שם כך שהוא בעצם נחשב כחלק ממערכת ההפעלה.
* פרוטוקול התקשורת בין הרכיבים הוא I2C או SPI ופרוטוקול התקשורת עם הקרקע הוא פרוטוקול Ax25. פרוטוקול Ax25 הוא פרוטוקול של שכבת הרשת כך שאין ממש מה להגיד על השכבות מעל.

### תיאור זרימת המידע

נתחיל עם מבנה הפאקטה, מדובר כאן על המבנה בתוך חלק ה-data של ה-AX.25 אז בעיקרון זה פרוטוקול השליחה של ה-packet:

A green and black rectangular sign

AI-generated content may be incorrect.

ID – מייצג גם איזה לוויין זה מתוך כל הלוויינים שאנחנו יכולים לקלוט במבנה הזה וגם את מספר הפקודה אחרי ההפעלה.

type – מייצג את סוג המערכת. TRXVU, EPS, FS וכדומה.

subtype – זה בעצם מספר הפקודה בתוך הקטגוריה. למשל יכול להיות פעמיים subtype = 00 אבל אחד מtype 1 והשני מ type 4.

data length – אורך ה-data שרוצים לשלוח בלי ה-headers שהוספתי. מכיוון שגם אם יש באפר אז הפאקטה תמיד בגודל קבוע אנחנו רוצים לדעת עד איפה אנחנו צריכים לקרוא את המידע.

Data – המידע. יהיה כאן את הפרמטרים של קבלת הפקודה ללוויין או שליחת הפקודה מהלוויין.

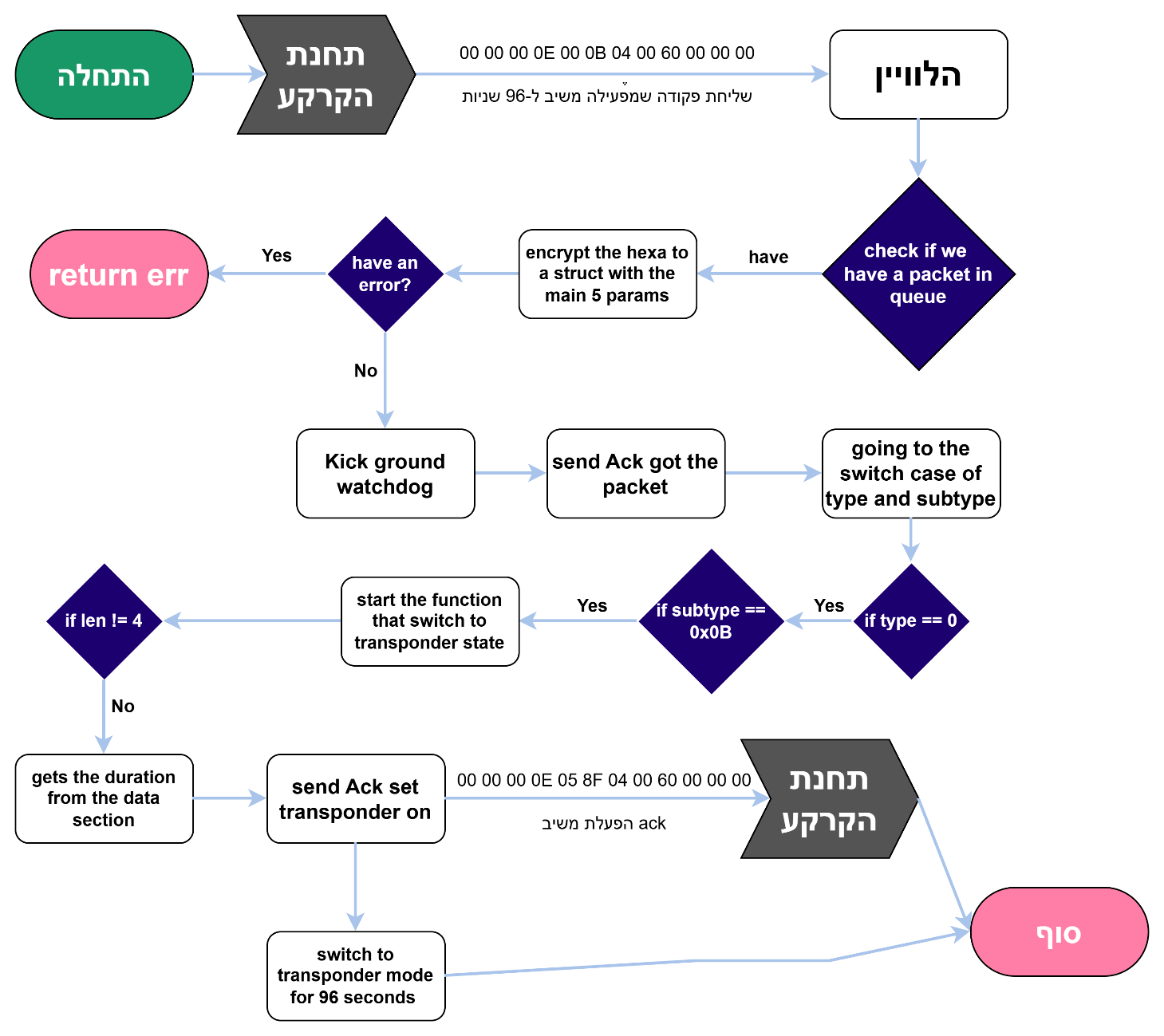
אני אציג את זרימת המידע עם דוגמא של פקודה שאפשר לשלוח ללוויין. וכדי להכניס 2 יכולות ביחד אני אציג את האחת של ה-Transponder. אני גם מתארת אז זה עבור לוויין מספר 14.

השליחה מהקרקע ללוויין וגם הפוך נעשית ב-little endian כל פרמטר לא כל ה-packet ביחד. כך שלמשל int יתהפך כדי שיהיה אפשר לקרוא אותו אבל אם יש אחריו מחרוזת היא לא תתהפך מכיוון שזה בעצם מערך של chars והם מתהפכים אבל בגלל שהם עם byte 1 זה נראה כאילו לא.

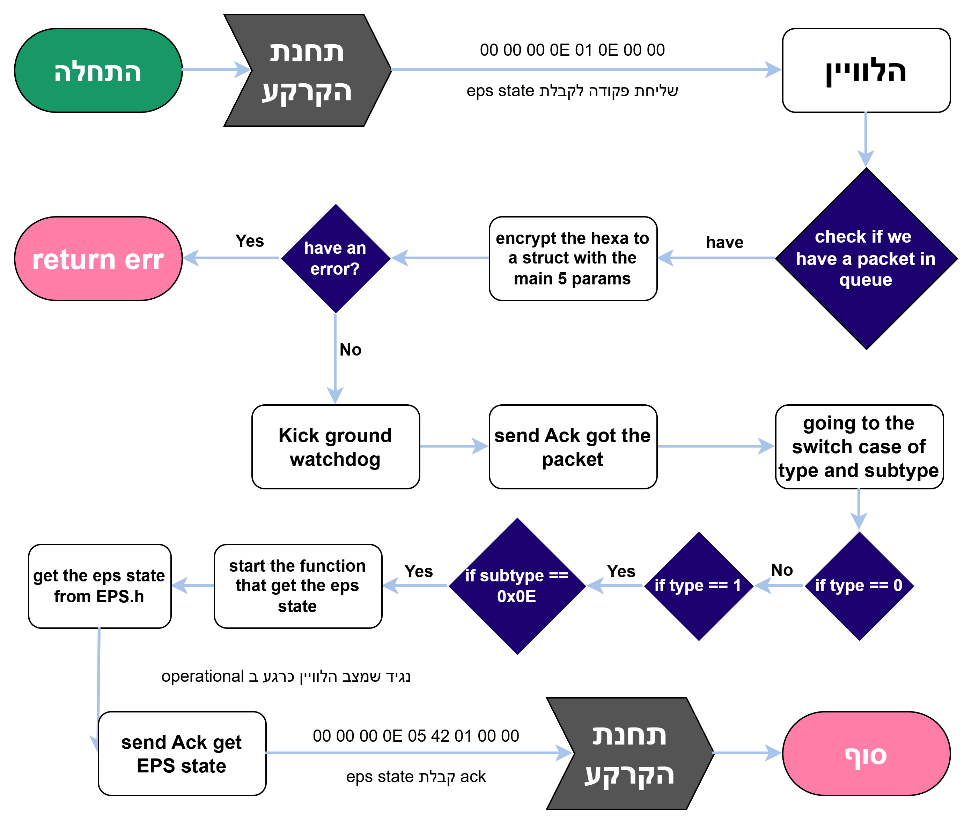
אני לא מכניסה את כל הלוגיקה מכיוון שזה יהיה יותר מדי ארוך אז אני רק כתבתי מה קורה אם הכל עובד כמו שצריך אפילו שיש בדיקות קצה בכל מקום.

האלגוריתם של היכולת נמצא בשקופית הבאה.

לא כתוב כפונקציות מכיוון שזה יהיה פחות מובן אז הכל במילים. האלגוריתם מייצג שליחה של בקשה להפעלת משיב ל-96 שניות.



החצי הראשון של האלגוריתם נכון בעיקרון לכל פקודה שנשלח. מה שמשתנה הוא אחרי בדיקת איזה type ו-subtype הפקודה.

דוגמא נוספת: נשלחת פקודת קבלת eps status.

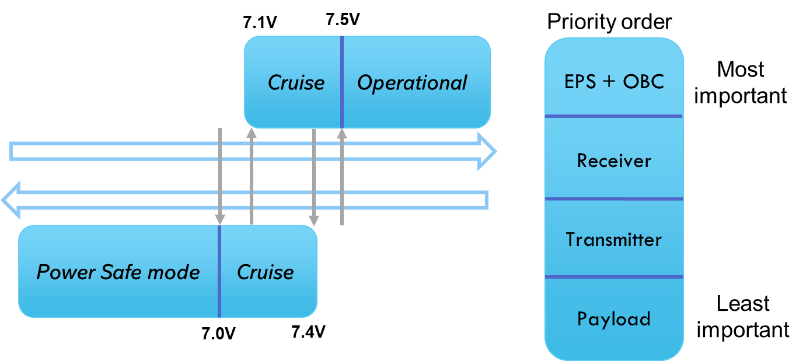
### תיאור אלגוריתמים

זהו האלגוריתם הראשי של התוכנית שנמצא בתוך קובץ main.c.

1. Bootup – מדבר על הפעלה ראשונית של המערכת או על אתחול. אחרי אתחול וגם בהפעלה התוכנה ישירות מתחילה לרוץ.

2. Init system – זוהי פונקציה שנמצאת בקובץ אחר שנקרא InitSystem.c. פקודה זו אחראית על לקשר את כל הרכיבים של המערכת (בין עם על ה-OBC או רכיב נפרד) כך שהתוכנה תוכל להשתמש בהם במהלך הקוד.   
בפקודה זאת יש גם את ההפעלה הראשונית. לפני השיגור לחלל מאתחלים מספר פרמטרים ואחד מהם הוא הפרמטר שזו הפעלה ראשונית ונצטרך לפרוס אנטנות. כאשר הלוויין נמצא בתהליך פריסת אנטנות הוא גם שומר טלמטריה, סופר זמן עד הפריסה ולפני הכל קודם כל מאפס לערכי ה-default את כל הערכים הדינאמיים.

3. EPS\_Conditioning – לוגיקת ה-EPS קובעת את מצב הלוויין לפי כמות הסוללה (יהיה טבלה למטה של הספים). יש 3 מצבים ו-4 ספים. יש בדיקות למעברים כדי שלא יהיה מצב שאנחנו כל הזמן זזים בין 2 מצבים אלה יש שינוי.



4. TRXVU\_Logic – כאן מתרחשת כל לוגיקת התקשורת. בין אם שליחת beacon, קבלת פקודות, תשובות לפקודות וכדומה. לחלק הזה צריך את כל הקבצים שקיימים בתיקייה של ה-TRXVU אבל הפקודה עצמה נמצאת ב-TRXVU.c. הוא בודק אם צריך לשדר ביקון ואם כן משדר, בודק האם קיבלנו פקודה ואם כן מטפל בה וכדומה.

5. Maintenance – פונקציה זו שנמצאת בפונקציית Maintenance.c עושה מספר דברים.

* בודקת מהו זמן הלוויין הכי מעודכן ושומרת אותו על ה-FRAM. למקרה של אתחול שיהיה לנו זמן מעודכן כמה שאפשר.
* בודקת אם יש צורך למחוק קבצים בגלל חוסר מקום. אם יש צורך היא מוחקת את 5% הישנים יותר. (המערכת שומרת תמיד 20% מקום פנוי)
* לאחר מכן מכיוון שה-payload הוא רכיב שלא נוסה לפני בחלל יש מנגנון fail check שאם הלוויין חווה reaset פעמיים ברציפות במהלך פחות מדקה ה-payload נכבה לאלתר עד הפעלה מחודשת דרך פקודה מתחנת הקרקע.
* לבסוף נבדק האם אנחנו צריכים לאתחל את הלוויין בגלל חוסר תקשורת ממושך עם תחנת הקרקע. אם כן מבצעים מספר פקודות ואז עושים hard reset ללוויין.

6. TelemetryCollectorLogic – לוגיקת שמירת מידע במערכת הקבצים. הפונקציה נמצאת ב-TelemetryCollector.c והיא אחראית על שמירת ה-telemetry של כל המערכות. היא בודקת קודם כל האם עבר מספיק זמן מהשמירה האחרונה. אם כן היא לוקחת את ה-data ומכניסה אותו למבנה הנתונים המתאים. המבנה נשלח לפונקציה שכותבת לכרטיס ושם נעשה:

* בודקים מה הזמן הנוכחי.
* משנים את הדרך שהזמן מוצג. (מייצוג של שניות מ-1970 לייצוג רגיל של תאריך ושעה ב-struct)
* נכנסים לפונקציה עם סוג הטלמטריה, ויוצאים עם הסוג שלה, גודל הטלמטריה שרוצים לשמור.
* אחר כך יוצרים את שם הקובץ באמצעות התאריך והסיומת
* מנסים לפתוח את הקובץ ולכתוב לו. אם הוא לא קיים הוא נוצר ואם פשוט אי אפשר הטלמטריה לא נשמרת ויוצאים מהפונקציה.
* לבסוף כותבים לקובץ את המידע יחד עם הזמן שהמידע נלקח בו (כמובן בסוף הכל נשלח ב-hexa)
* וסוגרים את הקובץ.

אחרי כל אלו אנחנו חוזרים להתחלה ומבצעים את כל הלולאה שוב. במקרה של reset חוזרים ממש להתחלה ל-Bootup.

בעיית אלגוריתם

בפונקציית ה- Maintenanceכמו שאמרתי יש אלגוריתם למחיקת הקבצים. במקור חשבתי לעשות את האלגוריתם הזה כמו בלוויינים הקודמים, אך כאשר ראיתי כמה מקום בממוצע נתפס ביום, אחרי ביצוע חישוב מקורב באמצעות גודל המידע, כל כמה זמן הוא נשאר וניחוש ל-log, רציתי לשנות אותו.

חלק מהלוגיקה הכללית מהלוויינים הקודמים של התוכנית נשמרה כך שאולי לא היה קוד לקחת ממנו השראה אבל כן ללוגיקה.

בעבר הלוגיקה של אותה הפונקציה הייתה מחיקה של מסמכים שעברו 30 יום. זאת אומרת שאחרי כחודש הטלמטריה תעלם אם לא הורדנו אותה לתחנה.

בעוד שבלוגיקה עכשיו הלוויין לא מוחק את הטלמטריה שהוא שומר כשנה, מכיוון שאחרי החישוב יצא שהלוויין יכול לשמור בכרטיס SD 1 סביבות השנה וחצי של מידע ואני עשיתי כך שבוודאות לא נגיע ל-0 מקום פנוי אז אחרי שהגענו ל-80% תפוס מפנים את המסמכים הישנים ביותר כך שרק 75% יהיה תפוס.

הפונקציה הזאת נקראת DeleteOldFiles והיא נמצאת במיקום הבא בתוך הפרויקט.

src\SubSystemModles\Maintenance\Maintenance.c

יש לזה גם את קובץ ה-.h באותו המקום.

### תיאור סביבת הפיתוח

פיתוח קוד הלוויין מתבצע בסביבת עבודה ייעודית המבוססת על **Eclipse Kepler (2014).** גרסה זו מותאמת למערכות Embedded ולדרייברים של הלוויין, ולכן מאפשרת שימוש מלא בכל הכלים והיכולות הנדרשים עבור הפיתוח.

בדיקות הלוויין מתבצעות על גבי מספר פלטפורמות שונות — חלקן תוכנות תקשורת, חלקן כלים לצריבה וחלקן מערכות ייעודיות שנבנו עבור התוכנית. להלן פירוט כלי העבודה העיקריים:

1. **PuTTY – תקשורת טורית ובדיקות בסיסיות**  
   בזמן שהלוויין נמצא אצלנו בקרקע, חלק גדול מהבדיקות מתבצע באמצעות חיבור ישיר (Serial) למחשב:
   * ניתן לבצע בדיקות שגרתיות, להדפיס הודעות Debug ולשלוח פקודות.
   * הפקודות ללוויין נשלחות מתוכנת בדיקה אחרת, אך התוצאות מופיעות גם ב־PuTTY. בעוד שבתוכנת הבדיקות הכל (חוץ מבדיקת שידור) נעשה ב-PuTTY.
   * החיבור מתבצע דרך קטגוריית Serial, כאשר מהירות התקשורת משתנה בהתאם לסוג התוכנה שמורצת בלוויין:
     1. Speed 2000000 – לתוכנת לוויין מוטסת
     2. Speed 115200 – לתוכנת ה-demo בה יש את הבדיקות השגרתיות.
2. **RFCB** **ISISpace – מערכת לבדיקת תקשורת רדיו**  
   זהו רכיב חומרה חיצוני המתחבר למחשב ומשמש בעיקר לבדיקת מערכת התקשורת של הלוויין:
   * מאפשר לשלוח פקודות ללוויין ב- Hexa בתדר הנכון (ב-VHF).
   * מאפשר לקבל מהלוויין את ההודעות שהוא משדר בתדר החוזר (ב-UHF).
   * משמש לבדיקה ראשונית של תקינות מערך התקשורת.
   * בשלב זה אין ממשק משתמש גרפי נוח, ולכן נדרש פיענוח ידני של הודעות Hexa לפי פרוטוקול התקשורת, שיצרתי. כך שאחרי מספיק פעמים ידעתי כבר לפענח את ה-Hexa ללא מפענח או הפרוטוקול.
3. **SAM-BA – כלי הצריבה הרשמי**Samba (SAM-BA) משמש לצריבת הקוד למעבד של הלוויין:
   * לאחר קומפילציה, הקוד נכתב לתוך זיכרון ה-NOR Flash.
   * לצורך בדיקות שאמורות לשרוד Reset, חייבים לצרוב ולא רק לטעון ל-RAM.
   * במהלך הפיתוח בוצעו מספר צריבות, אך האחרונה שבהן היא הצריבה הסופית ללוויין, יום לפני שליחתו לארה״ב.
4. **HoopoeNest — ממשק משתמש פנימי לשליחת פקודות**זהו אחד הכלים החשובים ביותר כיום לצוות תקשורת והוא נוצר על ידי מספר תלמידים בתוכנית עבורה.
   * זהו ה-GUI בתחנת הקרקע.
   * מאפשר לשלוח פקודות ולפרש תשובות ללא צורך בכתיבה ידנית ב־Hexa או קריאת ה-Hexa.
   * משמש גם בבדיקות קרקע של הלוויין כדי לוודא שניתן לקלוט נתונים דרך תחנת הקרקע שלנו.
   * מפשט מאוד את תהליך התפעול והבדיקה.
5. **ציוד תקשורת ותוכנות עזר**  
   בנוסף, נדרש ציוד תקשורת ותוכנות תקשורת חיצוניות המשמשות להאזנה, ניתוח, מעקב ושידור. דוגמאות למספר מתוכנות אלו:
   * SatPC32 — מעקב אחרי מסלול הלוויין ואנטנות
   * HDSDR — תוכנת SDR לקבלת אותות מהתדר, מהלוויין גם כן אבל הסינון הוא בתוכנה של soundmodem היא לא מראה שום דבר חוץ מאת מה שאמרנו שאפשר.
   * Soundmodem — מודם תוכנה בו קולטים את ה-packets מהלוויינים.
6. **הלוויין עצמו – סביבת הבדיקה החשובה ביותר (כי בלעדיו אין לוויין)**למרות שניתן לכתוב ולקמפל, **אין** אפשרות לבצע בדיקה עמוקה ואמיתית ללא הרצה על גבי הלוויין עצמו.  
   במסגרת העבודה על תוכנת הלוויין אני השתמשתי ב־Flatsat – לוויין פרוס שנמצא בחדר הנקי ומשמש כסביבת הבדיקות המרכזית לפני הגעת הלוויין שישוגר לחלל.  
   ה־Flatsat כולל את רוב רכיבי החומרה, המודולים והחיבורים, כך שאפשר להריץ עליו כמעט את כל המערכות כפי שיפעלו בפועל. (מכיוון שהוא לוויין שפשוט לא עמד בקריטריון של שיגור לחלל אבל אלו רכיבי לוויין אמיתיים).  
   עם זאת, החומרה בלוויין הסופי שונה, ולכן בעת המעבר ללוויין האמיתי נדרשים עדכונים בדרייברים והתאמות נוספות — אך **מרבית הקוד כבר כתוב ומוכן**.
7. העבודה **בחדר נקי** נדרשת כדי לספק סביבת עבודה סטרילית ובטוחה ללוויין ולמנוע זיהומים, פגיעות סטטיות ונזקים אחרים.

לסיכום: הפיתוח מתבצע ב-Eclipse Kepler, אך הבדיקות הן אלו שמאפשרות לאמת את התנהגות הלוויין במציאות – נעשות רק על גבי ה-Flatsat ועל הלוויין עצמו כאשר הוא מגיע והסביבות תומכות בכך (רואות את התוצאות/שולחות לו וכדומה).

### פרוטוקול התקשורת

פרוטוקול התקשורת שהלוויין משדר בו הוא Ax25. שזהו פרוטוקול תקשורת של חובבי רדיו בתחום של חובבי רדיו.

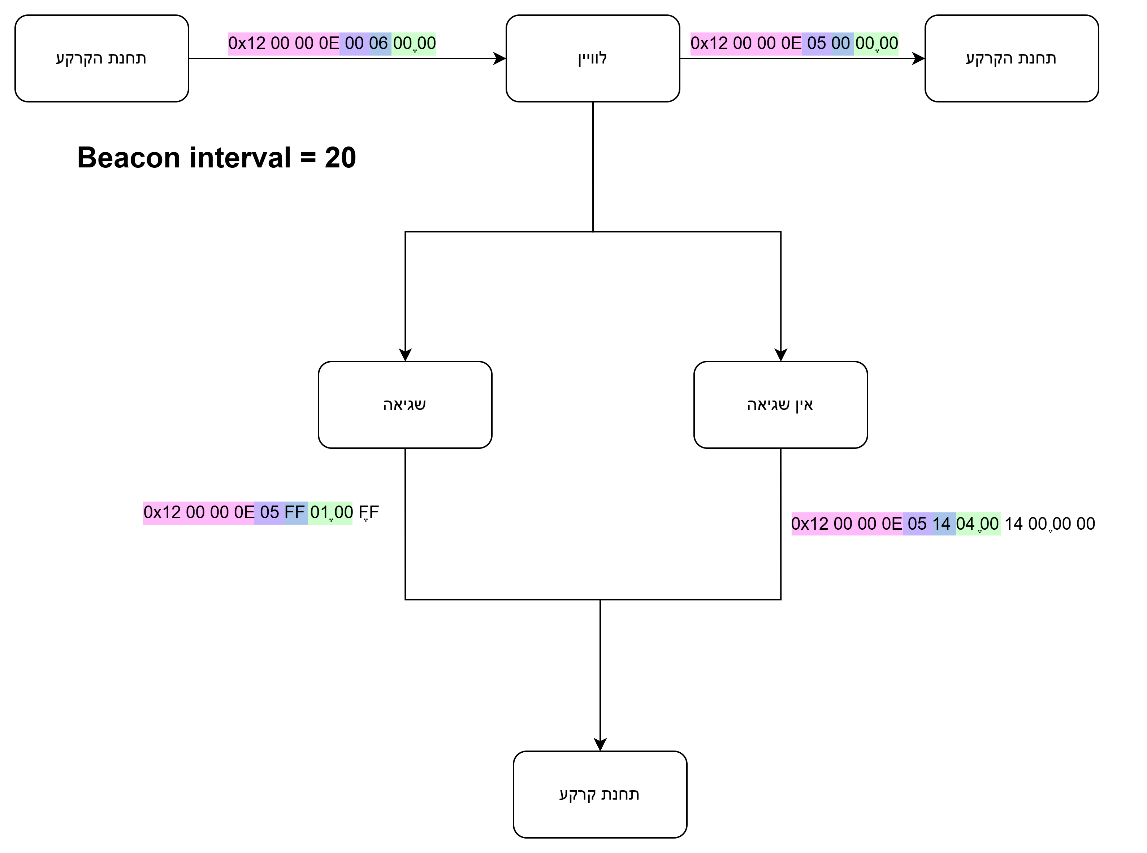
אך הפרוטוקול שכתבנו בתוכנית הוא באמת מה שמשתמשים בתוכנה מכיוון שכאשר אני מפענחת את הקוד בלוויין שאני מקבלת מה-driver אני מקבלת רק את החלק של ה-data בפרוטוקול שם אנחנו מכניסים את כל הלוגיקה שלנו כמו שהוצג לעיל, (אך אני אכניס את המבנה גם כאן).

A green and black rectangular sign

AI-generated content may be incorrect.

כרגע אני אפרט על 3 פקודות, 1 שנשלחת מהלוויין לקרקע ו-2 מהקרקע ללוויין ופקודת החזרה שלהן. הפירוש של כל פקודה יהיה למטה בפרוטוקול, אך גם שם יהיו דברים בהספחים.

בנוסף ה-ID של כל לוויין שונה כך שאני כרגע מציגה ללוויין 14, אבל הקוד פועל גם על 11, 16, 19.

1. שליחת ack אחרי reset. (מהלוויין לקרקע)  
   0x00 00 00 0E 05 7F 04 00 69 29 D1 4D
   1. ID – 00 00 00 0E. שזה מפורש הפוך אבל זה בעצם מראה שזו הפקודה הראשונה שנשלחה מהלוויין לקרקע וגם את זה שאנחנו על לוויין שהמזהה שלו הוא 14 (0E)
   2. Type – 05. זהו סוג של ack, כל פעם שנשלח ack מהלוויין זה בפורמט הזה. (רק הלוויין יכול לשלוח סוג כזה).
   3. Subtype – 7F. מייצג בפרוטוקול ack מסוג של הדלקה אחרי אתחול.
   4. Length – 04 00. נשלח ב-little endian וצריך לתרגם אותו ככזה. אורך ה-data שיבוא אחרי.
   5. Data – 4D D1 29 69. שוב כמו הכל הוא ב-little endian (ב-big: 69 29 D1 4D) מייצג את המספר 1764348237 שזה הזמן שעבר מאז 1.1.1970 00:00:00 בשניות (unix time). מכיוון שהפקודה מחזירה את הזמן על הלוויין לקרקע.
2. שליחת פקודה לקבלת ערך הזמן בין שני beacons (interval)  
     
   כמה הערות חשובות.
   1. הלוויין קודם שולח לקרקע הודעה (הודעת קבלת הפאקטה שכתובה בסוף המשפט) ורק אז מבצע את הלוגיקה. (0x12 00 00 0E 05 00 00 00)
   2. התשובה כאשר אין בעיה מחזירה לנו את המספר 20 כמו שכתוב בצד שזהו ה-interval.
   3. הפעם שלחתי שגיאה כללית שלא מייצגת מה היא אבל יש הרבה סוגי שגיאות שאפשר לראות אך זאת הלוגיקה הבסיסית.
3. שליחת פקודה לקבלת dump (בכלליות מאד)  
   A screenshot of a computer screen

   AI-generated content may be incorrect.  
   הסבר קצר.  
   ה-dump היא פונקציה של הורדת הרבה מידע מהלוויין בפעם אחת. אז זה למה יש סימני שאלה מכיוון שאני לא יודעת מה המידע (אני כן יודעת את הגודל אבל הוא משתנה מ-dump ל-dump. הפונקציה מחזירה מספר רב של מידע.  
   הפקודה הראשונה שמבצעים לאחר מציאת בחוסר שגיאה היא פקודת "התחלתי dump" הפקודה שחוזרת היא פקודת "סיימתי dump".  
   הפקודה ששולחים בקרקע מכילה: שם מערכת, זמן התחלה, זמן סיום, תדירות ה-data.  
   במהלך ה-dump יש גם אפשרות לשלוח הודעה שעוצרת אותו למקרה שהוא נמשך יותר זמן ממה שרצית.   
   גם כאן הסדר הוא קודם כל שליחת – קבלתי את הפאקטה, אחר כך ack של התחלת הפעולה ובגלל שזהו dump יש גם את כל הפעולה באמצע ואז שליחת סיום dump.

**הפרוטוקול המלא:**

יש אותו יותר ברור ב-excel חיצוני.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **פרוטוקול תחנת קרקע -> לוויין** | | | | | | |
| מערכת | | שם הפקודה | מספר מערכת | מספר פקודה | אורך [bytes] | פרמטרים | תיאור | |
| TRXVU | | Mute | 0x00 | 0x02 | 4 | duration [sec] | מפעיל את דגל ההשתקה (max: 7 days) | |
| Unmute | 0x03 | 0 | - | מכבה את דגל ההשתקה | |
| Set beacon interval | 0x04 | 4 | cycleTime [sec] | קובע קצב חדש בשביל שליחת הפקטה.  (min: 5 sec, max: 60 sec) | |
| Restore default beacon interval | 0x05 | 0 | - | מחזיר את הקצב המקורי. Interval = 20 [sec] | |
| Get beacon interval | 0x06 | 0 | - | מבקש את beacon interval | |
| Get Tx uptime | 0x07 | 0 | - | מבקש כמה זמן הרכיב פועל (Tx) | |
| Get Rx uptime | 0x08 | 0 | - | מבקש כמה זמן הרכיב פועל (Rx) | |
| Turn on idle | 0x09 | 4 | duration [sec] | מפעיל מצב idle בלוויין (max: 40 min) | |
| Turn off idle | 0x0A | 0 | - | מכבה מצב idle בלוויין | |
| Set transponder on | 0x0B | 4 | duration [sec] | מפעיל מצב משיב בלוויין (max: 14 days) | |
| Set transponder off | 0x0C | 0 | - | מכבה מצב משיב | |
| Set RSSI transponder | 0x0D | 2 | rssiValue [short] | קובע גודל rssi (בעיקרון כמה קולטים).  גודל: 0-4095 | |
| Set RSSI default | 0x0E | 0 | - | מחזיר לברירת מחדל:  rssi = 10 | |
| Get Ant uptime | 0x10 | 1 | antSide [0/1] | מבקש כמה זמן צד אנטנות היה פועל | |
| Ping | 0x13 | 0 | - | מחזיק "pong", רק כדי להראות תקשורת | |
| Get RSSI | 0x14 | 0 | - | מחזיר את ערך RSSI | |
| Off ants deploy | 0x15 | 0 | - | עוצר את ניסיונות פריסת האנטנות | |
| EPS | | Reset EPS WDT | 0x01 | 0x01 | 0 | - |  | |
| Update alpha | 0x02 | 4 | newAlpha [float] | מספר בין 0-1  smoothing factor | |
| Restore deafult alpha | 0x03 | 0 | - | ברירת מחדל 0.3 | |
| Get alpha | 0x04 | 0 | - | מבקש לדעת מה הערך של alpha | |
| Update thresholds | 0x05 | 16 | DPS, DC, UC, UO \* | שולחים בקשה לעדכון ספי מתח. יש טווחים אפשרים. [mV] | |
| Restore deafult thresholds | 0x06 | 0 | - | DPS = 6900, DC = 7300, UC = 7000, UO = 7400 [mV] | |
| EPS | | Get thresholds | 0x01 | 0x07 | 0 | - | מבקש קבלת ספי המתח. [mV] | |
| Get heaters value | 0x0C | 0 | - | מבקש את הספים להפעלת מחממים | |
| Set heaters value | 0x0D | 8 | lowThreshold, highThreshold | מספר עשרוני. מתי מפעילים מחממים. נוצר מתוך דרישה לא מומלץ להשתמש. | |
| Get EPS status | 0x0E | 0 | - | מבקש לדעת באיזה מצב הלוויין | |
| File System | | Delete all files | 0x02 | 0x01 | 0 | - | מוחק את כל הקבצים השמורים על הלוויין. | |
| Delete TLM | 0xA1 | 9 | type, start, end | מוחק לפי סוג ותאריך.  (סוג - byte) | |
| Delete TLM by type | 0xB1 | 1 | type | מספר בין 0-10. כמו הסוג של dump | |
| Get last FS error | 0xBB | 0 | - | מבקש את השגיאה האחרונה ב-FS | |
| Set TLM cycle | 0xCC | 5 | type [byte], newCycle [int] | סוג מ-0-6. תדירות של שמירה. | |
| Restore default TLM cycle | 0xCE | 0 | - | מחזיר את הכל ל-default. \*\*\* | |
| Get TLM cycle | 0xCD | 0 | - | מקבל את הרשימה של כל כמה זמן נשמר data מכל סוג | |
| Dump | 0x0E | 13 | type [byte], startDate [int], endDate [int], resolution [int] | שולח בקשה להוריד מידע שנשמר. סוג אפשר לראות בפרוטוקול לוויין-> תחנת קרקע ב-dump לפי מספר פקודה | |
| Get amount of free space | 0x0B | 0 | - | כמה מקום פנוי יש בלוויין (ב-bytes) | |
| Switch SD card | 0x0A | 1 | SD\_toSwitch | מספר בין 0-1. שולח בקשה להחליף | |
| Abort dump | 0x1E | 0 | - | כאשר רוצים לעצור dump באמצע | |
| Payload | | Turn off payload | 0x04 | 0x11 | 0 | - | מכבה payload עד להודעה חדשה | |
| Turn on payload | 0x20 | 0 | - | מדליק payload ומוריד את הדגל | |
| Maintenance | | Get ground WDT duration | 0x03 | 0x01 | 0 | - | מבקש את הזמן שבו חוסר תקשורת תגרום ל-reset ללוויין | |
| Set ground WDT duration | 0x02 | 4 | newDuration [sec] | שינוי אותו הזמן לזמן אחר. (min: 3 days) | |
| Update sat time | 0xDD | 4 | newTime [sec] | פקודה חשובה אך קטלנית. מינימום לתחילת שנת 2000. משנה את זמן הלוויין. | |
| Maintenance | | Reset component | 0x03 | 0xDE | 1 | type | שולח פקודת reset. \*\*\* | |
| Get sat uptime | 0xE3 | 0 | - | מבקש את הזמן שבו הלוויין היה דלוק בשניות | |
| Get sat time | 0xE4 | 0 | - | מבקש את זמן הלוויין ב-unix | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **פרוטוקול לוויין -> תחנת קרקע** | | | | | | | |
| מערכת | | שם הפקודה | מספר מערכת | מספר פקודה | אורך [bytes] | פרמטרים | תיאור | מערכת שאליה זה קשור |
| TRXVU | | Beacon | 0x00 | 0x01 | 141 | יהיה בנספחים json שיסביר | שולח לקרקע כל interval של שניות אות חיים בעל פרמטרים חשובים של הלוויין. | כללי |
| ACK | | FS reset | 0x05 | 0x70 | 0 | - | נשלח לפני שהולכים לרסט את אחת המערכות. לכל מערכת יש את המספר שלה ששולחים  מה כל מספר:   1. Soft 2. Hard 3. TX 4. RX 5. Ants 6. Soft payload 7. Hard payload 8. FRAM 9. File system | Maintanance |
| Payload soft reset | 0x71 | 0 | - |
| Payload hard reset | 0x72 | 0 | - |
| TX hard reset | 0x73 | 0 | - |
| RX hard reset | 0x74 | 0 | - |
| ANTS reset | 0x75 | 0 | - |
| EPS reset | 0x80 | 0 | - |
| Soft reset | 0x81 | 0 | - |
| Hard reset | 0x82 | 0 | - |
| FRAM reset | 0x83 | 0 | - |
| Get ground WDT duration | 0x48 | 4 | groundWDT |  |
| Set ground WDT duration | 0xA5 | 0 | - | שולח ACK שהפקודה בוצעה |
| Update sat time | 0x12 | 4 | newSatTime | שולח ACK שהפקודה בוצעה (וגם למה השתנה) [sec - unix] |
| Get sat uptime | 0x47 | 4 | satUptime | שולח את התשובה לבבקשה [sec] |
| Get sat time | 0x46 | 4 | satTime | שולח את התשובה לבבקשה [unix] |
| Kill payload | 0x50 | 0 | - | שולח אישור לבקשת הכיבוי עד להודעה חדשה | Payload |
| Turn on pyload | 0x51 | 0 | - | שולח אישור על הדלקה (גם מוריד את הדגל של עד להודעה חדשה) |
| ACK | | Delete TLM | 0x05 | 0xC0 | 0 | - | בכל סוג של מחיקה זה מה שנשלח | File System |
| FS last error | 0xC1 | 4 | FS\_lastError | מחזיר את השגיאה האחרונה ב-FS |
| Set TLM cycle | 0x85 | 0 | - | שולח שהפקודה התבצעה |
| Get TLM cycle | 0x45 | 28 | eps, trxvu, ants, SP, wod, radfet, sel\_seu | שולח את כל התדירויות של שמירה למערכת הקבצים של data [sec] |
| Get free space | 0x44 | 4 | freeSpace | שולח מקום פנוי על הכרטיס הנוכחי |
| Dump start | 0x90 | 0 | - | נשלח בתחילת dump |
| Dump finished | 0x92 | 0 | - | נשלח בסיום dump |
| switch SD card | 0xC0 | 0 | - | בכל סוג של מחיקה זה מה שנשלח |
| Reset EPS WDT | 0x15 | 0 | - | מצביע על כך שהפקודה התבצעה | EPS |
| Update alpha | 0x17 | 0 | - | שולח שבוצעה הפקודה |
| Get alpha | 0x40 | 4 | alpha | מחזיר את ערך ה-alpha |
| Update thresholds | 0x16 | 0 | - | שולח שבוצעה הפקודה |
| Get thresholds | 0x41 | 16 | DPS, DC, UC, UO \*\* | מחזיר את ספי המתחים של הלוויין |
| Get heaters value | 0x43 | 8 | lowThreshold, highThreshold | מחזיר את ערכי הסף של הדלקה וכיבוי מחממים |
| Set heaters value | 0x55 | 0 | - | מודיע ששונו |
| Get EPS status | 0x42 | 1 | state | מספר בין 0-2 מציג את מצב הלוויין |
| ACK | | Mute | 0x05 | 0x8D | 4 | duration [sec] | ack הופעל, לכמה זמן הופעל | TRXVU |
| Unmute | 0x8E | 0 | - | פקודה בוצעה |
| Set beacon interval | 0x14 | 4 | newCycle [uint] | הצליח, המחזור של שליחה החדש. |
| Get beacon interval | 0x4A | 4 | interval [uint] | מחזיר את המחזור של שליחה. |
| Get Tx uptime | 0x4B | 4 | TxUptime [sec] | כמה זמן ה-Tx פועל מאז reset |
| Get Rx uptime | 0x4C | 4 | RxUptime [sec] | כמה זמן ה-Rx פועל מאז reset |
| Turn on Idle | 0xA1 | 4 | duration [sec] | ack הופעל, לכמה זמן הופעל |
| Turn off Idle | 0xA2 | 0 | - | פקודה בוצעה |
| Set transponder on | 0x8F | 4 | duration [sec] | ack הופעל, לכמה זמן הופעל |
| Set transponder off | 0x9F | 0 | - | פקודה בוצעה |
| Set RSSI | 0x18 | 2 | newRssi | הצליח, מחזיר את ערך ה-rssi בלוויין |
| ACK | | Get Ant uptime | 0x05 | 0x4D | 4 | AntUptime | השניות. כמה זמן פועל | TRXVU |
| ping | 0xAA | 4 | "pong" | שולח pong אחרי שקיבל ping |
| Get RSSI | 0x49 | 2 | rssi | מחזיר את ערך ה-rssi בלוויין |
| Receive com | 0x00 | 0 | - | כל פעם שהתקבלה פקודה בלוויין | כללי |
| Commend exec | 0x01 | 0 | - | כאשר אין ack מובנה לכך |
| reset wakeup | 0x7F | 4 | satTime | שולח שבוצע ריסט ואת זמן הלוויין |
| Finish first active | 0x35 | 0 | - | נשלח כשהסתיימה ההפעלה ה-1 |
| Ready for takeof | 0xFB | 0 | - | נשלח כאשר מכינים את הדגלים. |
| Unknown subtype | 0xBB | 0 | - | למקרה של מספר פקודה לא ידוע |
| Unknown type | 0xCC | 0 | - | למקרה של מספר מערכת לא ידוע |
| Error msg | 0xFF | 1 | errorType | שולח מספר שגיאה, פירוט בהמשך. |
| Dump | | EPS TLM | 0x06 | 0x00 | 120 | \*\*\*\* | dump הוא מספר שליחות. כאן יש שליחה אחת של כל מערכת. כל אחת מהמערכות כאשר עושים לה dump שולחת מספר עצום (לרוב) של packets כאלו.  המספר של פרמטרים הוא לרוב גדול לכאן הוא בנספחים. הסבר גם על המשתנה יהיה שם. | EPS |
| TX TLM | 0x01 | 22 | \*\*\*\* | TRXVU |
| Ants 0 TLM | 0x02 | 13 | \*\*\*\* | TRXVU |
| Solar Panels TLM | 0x03 | 28 | \*\*\*\* | EPS |
| WOD TLM | 0x04 | 145 | \*\*\*\* | TRXVU |
| Radfet TLM | 0x05 | 20 | \*\*\*\* | Payload |
| SEL TLM | 0x06 | 16 | \*\*\*\* | Payload |
| SEU TLM | 0x07 | 8 | \*\*\*\* | Payload |
| RX TLM | 0x08 | 26 | \*\*\*\* | TRXVU |
| Ants 1 TLM | 0x09 | 13 | \*\*\*\* | TRXVU |
| Log TLM | 0x0A | משתנה | \*\*\*\* | כללי |

\*ראשי תיבות בתחילת המסמך

\*\* אפשר לראות בטבלה dump type את סוגי הקבצים

\*\*\*יהיה רשום בהמשך. בטבלה של reset type.

\*\*\*\*תיאור לערכים ב-dump יהיו בנספחים

טבלאות עזר לפרוטוקול:

|  |  |  |
| --- | --- | --- |
| number | type | default |
| 0 | EPS | 10 [sec] |
| 1 | TRXVU | 10 [sec] |
| 2 | ANTS | 10 [sec] |
| 3 | Solar panels | 10 [sec] |
| 4 | WOD | 10 [sec] |
| 5 | Radfet | 15 [min] |

טבלה: Type TLM period

|  |  |
| --- | --- |
| number | type |
| 0 | EPS |
| 1 | TX |
| 2 | Ant 0 |
| 3 | Solar Panels |
| 4 | WOD |
| 5 | Radfet |
| 6 | SEL |
| 7 | SEU |
| 8 | Rx |
| 9 | Ant 1 |
| 10 | log |
| טבלה: Dump type | |

|  |  |
| --- | --- |
| number | type |
| 0 | soft |
| 1 | hard |
| 2 | Tx |
| 3 | Rx |
| 4 | ants |
| 5 | payload |
| 6 | payload hard |
| 7 | FS |
| 8 | FRAM |
| טבלה: Reset type | |

|  |  |
| --- | --- |
| Error number | Error msg |
| 1 | בעיה בקריאה מה-FRAM |
| 2 | בעיה בכתיבה ל-FRAM |
| 3 | נכתב ב-FRAM שגוי |
| 10 | שגיאה בכתיבה ל-I2C |
| 15 | לא מצליח לקבל זמן |
| 17 | שליחת גודל לא נכון לפקודה |
| 20 | לא מצליח לקבוע זמן סיום |
| 23 | לא צד אנטנות קיים |
| 24 | לא מצליח לקבל uptime |
| 26 | אין אפשרות לאתחל WDT |
| 27 | בעיה בלקבל משתנים מה-struct |
| 28 | ערך alpha בלתי אפשרי |
| 29 | ערך threshold לא אפשרי |
| 30 | אי אפשר לעשות את מה שרוצים |
| 40 | לא מצליח לעשות אתחול |
| 50 | לא מצליח לעדכן זמן לוויין |
| 60 | לא הצליח למחוק את כל הקבצים |
| 69 | לא קיים כרטיס SD כזה |
| 70 | אותו כרטיס שכבר משומש |
| 255 | שגיאה לא ידועה |

טבלה: error type

### תיאור מבני נתונים

בלוויין אין שימוש במבני נתונים כפי שאנחנו מכירים (חוץ ממבני נתונים של התוכנה כמו struct, union וכדומה), אך קיימים קבצים שנוצרים לשם אחסון נתונים. הקבצים הללו מגיעים עם סיומת המותאמת למערכת. לדוגמה, מידע על מערכת ה-EPS נשמר בקובץ עם הסיומת "eps.". זה מתאים משום שהנתונים שמתקבלים מהלוויין מגיעים בקטע Hexadecimal, ולכן נשמרים באופן זהה, והסיומת מאפשר להבחין איזו מערכת היא זאת שאנחנו קוראים.

בנוסף, קיים גם קובץ log, אשר נועד לעקוב אחרי שגיאות שמתרחשות בלוויין, כך שנוכל לדעת בדיוק מה היו הבעיות ומתי.

**מבנה הקבצים ומידע בתוכם:**

* **שם קובץ:** {date}.{subsystem}
* **תאריך יצירה:** כל יום נוצר קובץ חדש שמכיל את המידע של אותה מערכת, עבור כל אותו היום.
* **מבנה שורה בקובץ:** {time}{telemetry at that time}

### סקירת חולשות

בלוויין אין אפשרות להפעיל הצפנה, שכן אנחנו עובדים בתדרי חובבים, ובאופן כללי אסור לבצע תקשורת מוצפנת בתדרים אלו. זהו אחד התנאים שנדרשים כאשר מקבלים רישיון חובבות וכדי להשתמש בתדרים אלו.

המערכת בלוויין מוגנת בצורה מיטבית עד שלב של שליחת פקודה חוקית. ברגע ששולחים פקודה חוקית, אין למעשה שום אמצעי למנוע את ביצועה. מעבר לכך, אין חשש שנוכל לשלוח פקודות שגויות או בטעות לגרום ללוויין לבצע פעולה לא נכונה, שוב מכיוון שהוא יודע להבדיל וגם עליו יש הגבלות אבל יכולים לשנות לנו דברים בלי שנדע אם למישהו יש את הפקודות. כל פקודה חייבת להיות מדויקת ומובנת לחלוטין.

המערכת של הלוויין עצמאית ואינה דורשת מעקב צמוד. כל פקודה שיש לשלוח אליה חייבת להיות מאוד מדויקת, אך מעבר לכך לא נדרשות התערבויות נוספות.

# **HoopoeWorld מסמך אפיון**

A planet earth with a broken face

AI-generated content may be incorrect.

## מבוא:

### תיאור כללי של הפרויקט

הפרויקט הוא תוכנה ללוויין ותוכנה נוספת (בשם HoopoeWorld) שיכולה לרוץ על כל מחשב (בהעדפה על שרת חזק עם הרבה מקום אחסון) שתשמש כשרת לאתר https.

התוכנה תשמש כשרת וכלקוח שיכול לגשת ל-Beacon, אות חיים מהלוויין שמכיל את כל המידע החשוב ביותר, שנקלט מתחנות קרקע ברחבי העולם.

באמצעותה ניתן יהיה לצפות בפיענוח הפקאטות ולקבל תמונה כוללת של מצב הלוויין.

### קהל היעד

קהל היעד העיקרי הוא **האנשים בתוכנית שלי** (וגם אלו שיצטרפו אליה) – שיוכלו להשתמש ב"אפליקציה", בשרת ובאתר כדי לקבל מידע על הלוויינים מכל רחבי העולם ולא רק מתחנת הקרקע שלנו. וגם יוכלו להשתמש בלוויין שכתבתי לו את הקוד. כאן ספציפית זה בעיקר תלמידי צוות תקשורת.

* הם יוכלו להוסיף לתוכנה לוויינים חדשים לבדוק.
* ארצה שגם יהיה מספיק תיעוד כדי שיוכלו להמשיך ולשפר את הפרויקט אחרי.

### סיבות לבחירה בפרויקט

בחרתי בפרויקט הזה כי רציתי לעסוק בתחום שמעניין אותי ואני מרגישה שיש צורך במוצר.

אני אוהבת ליצור דברים שיש בהם צורך אמיתי, ובמקרה הזה אני מביאה תוכנה חדשה שתשמש את התוכנית שלי.

כבר זמן רב חשבנו על רעיון של מערכת שתאסוף את כל המידע ותמיין ותציג אותו בצורה ברורה ונוחה, וזאת מטרת התוכנה. בעוד שבלוויין נכנסתי לתוכנית כי רציתי לעשות את זה ובסוף באמת עשיתי.

### מטרות הפרויקט

* הצגת מידע שנאסף אך כיום לא נעשה בו שימוש.
* ניתוח המידע כדי לאתר שגיאות ותקלות בזמן אמת באופן יותר יעיל ומהיר.
* מעקב צמוד יותר אחרי הלוויינים.
* אפשרות להפיק גרפים ונתונים ויזואליים נוספים לניתוח.

### ייחודיות וחדשנות

נכון לעכשיו, אין תוכנה אחרת שעושה בדיוק את מה שאני מתכננת (התוכנה של ה-HoopoeWorld).

התוכנה הקיימת בתחנת הקרקע שלנו מסוגלת רק לקלוט ולפענח מידע שנקלט אצלנו ולשמור אותו, אך לא לאסוף מידע מכל העולם. ומצד שני האתר שממנו אני רוצה לקחת נתונים מציג מידע בפורמט גולמי (Hexa), ללא מיון לפי לוויין או הצגה קריאה.

אני מתכננת ליצור תוכנה שתדע לאסוף את המידע מהאתר, למיין אותו, להציג את המידע בצורה נוחה וברורה, במספר מחשבים דרך שרת עם אפשרות להרחבה בעתיד.

### מגבלות ופיתוח עתידי

בעיקרון ה-HoopoeWorld הוא בלי הגבלות רק צריך מחשב מספיק חזק כדי שיוכל לשמש כשרת. בהגבלות לתוכנה יכול לקרות מצב בו האתר שהתוכנה מנסה לקרוא ממנו יקרוס ואז לא נקבל מידע חדש עד שהאתר יעלה מחדש ואם הלוויין לא נמצא באתר לא נוכל לקבל אליו מידע נוסף. התוכנה נועדה להיבנות כך שיהיה ניתן להוסיף בקלות לוויינים חדשים וסוגים שונים של Beacons, ולהתאים אותה בהמשך – בניגוד לתוכנה הנוכחית שלנו בתחנת הקרקע שמקבלת מידע רק מהתחנה, שאף אחד לא יודע לערוך אותה ולא מנסים כדי לא להרוס. היא גם נבנתה כבסיס מספיק ברור שיהיה אפשרות להגדיל אותה וכאשר יוצרים אצלנו תוכנת קרקע חדשה יוכלו לחבר אותה לאתר שלי.

### היבטים טכניים

הפרויקט כולל בתוכו:

* עבודה עם מבני נתונים.
* פיתוח מערכת שרת ומספר לקוחות.
* שימוש בפרוטוקול תקשורת מוצפן (HTTPS).
* כתיבה ויישום של פרוטוקול תקשורת.
* הורדת מסמכים משרת ללקוח.
* שרת דינאמי הפועל לפי json.

### בדיקות המערכת

* לראות שאין כפילויות ב-database בגלל שזה אומר שיש לי בעיה בקוד. תתבצע על ידי הכנסת הנתונים כאשר אני יודעת שיש כפילויות ואז לשלוח שאילת שסופרת את הכל ואחת שסופרת רק את המיוחדים. כאשר אני רואה שהמספרים שווים אני יודעת שזה עובד כראוי.
* שהלקוח באמת מקבל כל פעם שהוא גולל למטה את המידע הנוסף. יתבצע על ידי כניסה למערכת ולראות שבאמת מקבלים עוד דברים.
* שכאשר הוא למעלה הוא מתעדכן על המידע החדש כאשר עוברות שעתיים וגם בשרת התווסף מידע.
* לראות שאם אני משנה את המקום ממנו אני לוקחת הכל עדיין עובד ורץ. יעשה באמצעות פנייה לתוכנה ובקשת לוויין אחר ולקחת מעט ואז אני אראה שהכל טוב (וגם אראה שזה מתרגם נכון לפי פרוטוקול Beacon שונה).
* כאשר לוחצים על הלוויינים השונים באמת משתנה המידע ללוויין הנכון.

### גנט – תוכנת ה-HoopoeWorld

זה רק בשרת כי אני משתמשת בפרוטוקול HTTPS. הלמידה קוראת תוך כדי עבודה. מה שצריך ללמוד כדי לעשות את זה אני עושה.

תוך כדי גם כותבים ומשכתבים את הספר.

|  |  |
| --- | --- |
| **שלב** | **תאריך – סיום** |
| אפיון פרויקט | 30.11.25 |
| להכין את אופי ה-json ואת ה-config | 10.12.25 |
| להכין את התוכנה שיוצרת את הטבלאות | 20.12.25 |
| להכין קובצי אתר שיראו ערכים מפוברקים (מבחינת html, css, js) | 1.1.26 |
| להתחיל להכין את השרת – פיענוח הפאקטות לפי ה-json וה-SQL. | 25.1.26 |
| לשלב בין קבצים של האתר לפיענוח של השרת (שיהיה בפורמט שנוכל להכניס את זה כמידע לאתר) | 30.1.26 |
| להכין את ה-class לקוח בצד שרת | 10.2.26 |
| יצירת האפשרות לחיבור באמצעות HTTPS | 1.3.26 |
| הורדת המידע מהלוויין ב-excel למחשב הלקוח לפי כמות, זמן, או הכל. | 10.3.26 |
| סיום כתיבת ספר פרויקט. | 1.4.26 |

## פירוט יכולות התוכנה:

### צד לקוח:

הלקוח הוא אתר אינטרנט המשמש ממשק להצגת נתונים ולביצוע בקשות לשרת.

* בחירת לוויין.
  + המשתמש יכול לבחור מתוך רשימת לוויינים קיימים במערכת על מנת להציג את הנתונים שלהם.
  + נדרש:
    - כפתורי בחירה או רשימת Select ב-HTML/JS
    - שליחת בקשה מסוג GET לשרת לקבל את הדפים
    - שליחת GET נוסף לקבלת נתוני לוויין שנבחר
  + נחוץ שיהיה בצד השרת את ה-JSON.
* הצגת ביקון
  + הצגת המידע שנאסף מה-Beacon של הלוויין באופן קריא. במקום Hexa גולמי, מוצג מידע מפוענח לפי הפורמט שהוגדר בשרת וגם מקבל את המידע שהוא כבר לא גולמי מהשרת (מקבל מחדש את העמוד).
  + נדרש:
    - שליחת בקשת GET לשרת לקבלת נתוני ה-Beacon העדכניים
    - הצגת הנתונים בדפדפן בפורמט נוח לקריאה
    - יכולת לקבל נתונים חדשים בזמן שהמשתמש נמצא באתר (עדכון תקופתי וכדומה)
  + נחוץ:
    - פונקציית JS להצגת וקבלת הנתונים
    - אזור תצוגה ב-HTML (טבלה / כרטיסיית נתונים)
* מנגנון בקשת נתונים אוטומטית
  + על ידי גלילה של העכבר / כמות זמן שעבר הלקוח מקבל עוד נתונים על הלוויין.   
    גלילה מטה, מקבל עוד למעלה כל שעתיים, אך רק עם יש עוד
  + נדרש:
    - פונקציות JS המטפלות במנגנון
    - פרמטרים שנשמרים ב-JS
    - החזרה מסודרת של הפרמטרים
* בקשת הורדת excel
  + המשתמש יכול להוריד קובץ Excel של נתוני הלוויין — את כל העבר, לפי זמנים או רק מספר פאקטות אחרונות.
  + נדרש:
    - כפתור הורדה ב-HTML
    - שליחת GET לשרת עם פרמטרים
    - ניהול הורדת קובץ ידידותית למשתמש

### צד שרת:

צד השרת בתוכנה הזאת הרבה יותר מורכב מכיוון שהוא עושה הרבה פעולות ובסוף שולח לדפדפן את האתר.

* מבני נתונים (JSON, SQL)
  + שמירת נתונים של כל לוויין בטבלת SQL. קיימת טבלה שממפה בין לוויין לבין הטבלה המתאימה (יכול להיות שיהיה ב-config של ה-json אבל במהלך הזמן אדע)
  + נדרש:
    - יצירת בסיס נתונים SQL
    - יצירת טבלאות דינמיות בהתאם לנתוני כל לוויין
    - מנגנון הכנסת נתונים ל-SQL על בסיס קובצי JSON המגדירים פורמט
    - קבצי JSON של מבנה הנתונים
* Config
  + קבצי JSON המגדירים את פורמט ה-Beacon עבור כל לוויין, וכן את כתובתו. בקובץ ה-config יהיה את מזההו של הלוויין, כתובת ה-json, וקישור לטבלת הנתונים שלו שבאתר (ואולי גם בשרת עצמו).
  + נדרש:
    - שמירת קבצי JSON נפרדים לכל לוויין
    - הגדרת פרוטוקול כתיבה אחיד (schema) שמאפשר להוסיף או להוריד לוויינים בקלות
    - תוכנה שתפרש שתבין את ה-JSON ותתרגם Hexa לנתונים.
* בניית אתר ושליחתו ללקוח
  + השרת יוצר את דפי ה-HTML/JS/CSS באופן דינמי בהתאם לנתונים הקיימים בבסיס הנתונים, ומגיש אותם בפרוטוקול HTTPS.
  + נדרש:
    - תבניות HTML (או יצירה דינמית)
    - הפעלת שרת HTTPS
    - קריאת הנתונים מבסיס הנתונים והזרמתם לדפדפן
    - מנגנון גלילה דינמית / טעינה מתמשכת אם נדרש
* שליחת כל המידע או חלק ב-excel
  + השרת אוסף את כל נתוני הלוויין מה-SQL (או רק מספר השורות שהלקוח ביקש), מייצר מהם קובץ Excel ושולח ללקוח. \* מביא את החדשים ביותר.
  + נדרש:
    - שאילתת SQL מבוקרת
    - יצירת Excel באמצעות ספרייה מתאימה
    - שליחת הקובץ בתגובה ל-GET
  + למידה של משהו חדש. (אני לא יודעת איך אפשר לעשות את זה אז אני לא יודעת מה זה מכיל. כאשר אגיע לשם אראה ואוסיף)

## ארכיטקטורת הפרויקט:

### פירוט חומרה

* מחשב חזק עם הרבה מקום לשמש כשרת.
* לוויין (שיהיה מה לקלוט)
* מחשבים להתחבר דרכם.

### תיאור טכנולוגי

* בתוכנה הזאת יש מספר שפות תוכנה. שפת השרת תהיה python אך כדי ליצור את האתר אצטרך שפות אתרים כגון: html, css, js.
* ה-OS עליו התוכנה פועלת יהיה windows.
* פרוטוקול התקשורת בין הרכיבים הוא HTTPS.

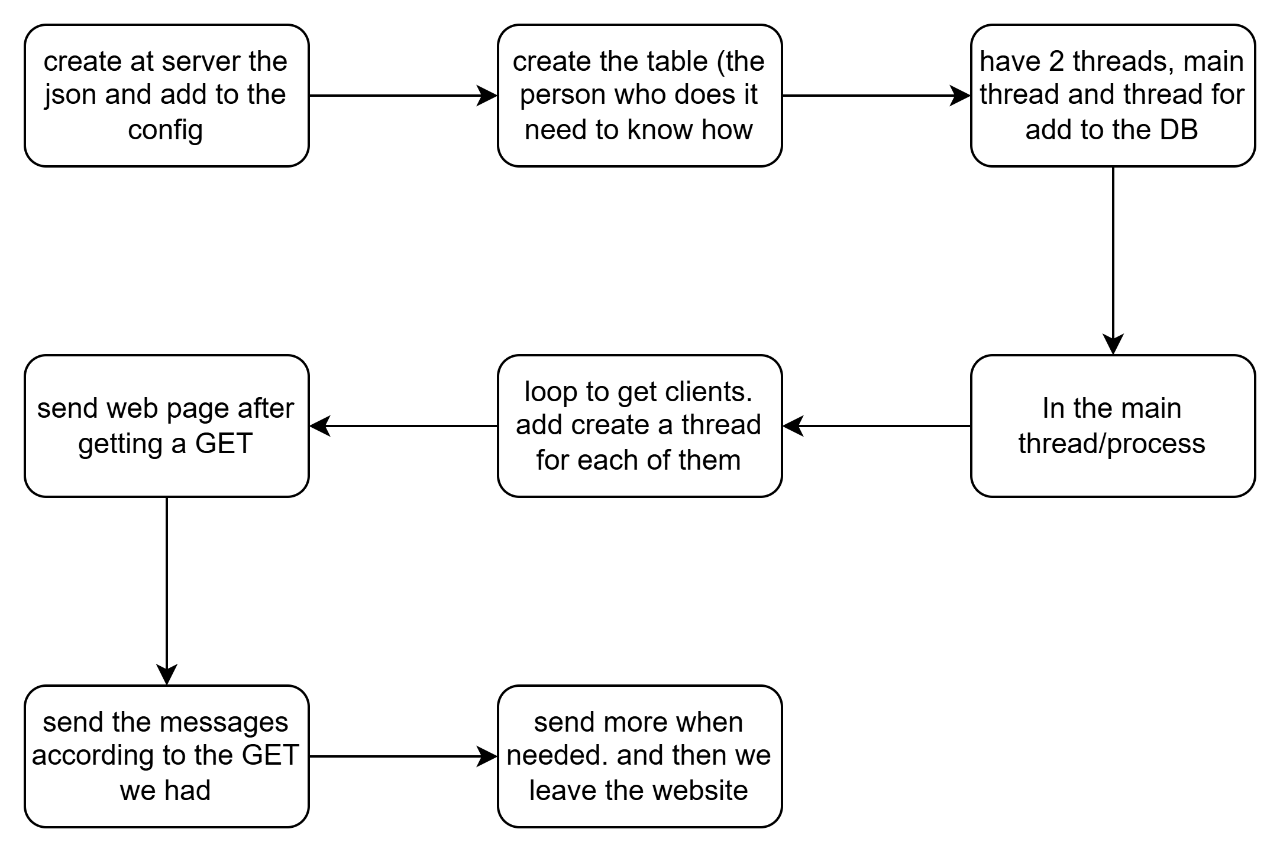
### תיאור זרימת המידע

A screenshot of a cell phone

AI-generated content may be incorrect.זה עבור היכולות הדינאמיות של מבני הנתונים.

דברים אחרים אצרך להוסיף בהמשך. עדיין אין לי רעיון בטוח על איך יעשו.

### תיאור אלגוריתמים



זה בעיקרון מה שאמור לקרות באלגוריתם של התוכנה.

### פרוטוקול תקשורת

פרוטוקול ב-HTTPS וכל הפקודות הן פקודות GET.

הפירוט שתראו יהיה בקשר לתוכן של פקודת ה-GET.

**לקוח -> שרת**

התקשורת בין הלקוח לשרת מתבצעת באמצעות בקשות HTTPS.

כל הנתונים נשלחים דרך ה-url.

**בחירת לוויין:**

Endpoint: /chooseSatellite/{satelliteName}

בעת בחירת לוויין בפעם הראשונה, נשלחת בקשה זו לשרת.

השרת מחזיר ללקוח את 25 ה-packets האחרונות של הלוויין שנבחר בצורה הנראית ידידותית למשתמש.

**קבלת נתונים חדשים (למעלה):**

Endpoint: /addTop/?satName={}&mostResent={}

כל שעתיים התוכנה בודקת מול השרת האם קיים מידע עדכני יותר מהמידע שנמצא אצל הלקוח. אם קיים מידע חדש – הוא נשלח ללקוח.

הקבלה מתבצעת על פי:

* שם הלוויין – satName
* זמן הלוויין העדכני ביותר הקיים אצל הלקוח – mostResent

**קבלת נתונים חדשים (למטה):**

Endpoint: /addBottom/?satName={}&leastResent={}

כאשר המשתמש גולל כלפי מטה והגיע כמעט לסוף הנתונים המוצגים, התוכנה מבקשת מהשרת פקאטות נוספות. גם כאן 25 מהן.

הקבלה מתבצעת על פי:

* שם הלוויין – satName
* זמן הלוויין הכי פחות עדכני הקיים אצל הלקוח – leastResent

**הורדת נתונים כ-excel:**

Endpoint: /downloadData/?type={typeOfDownload}&satName={}

זהו Endpoint כללי לכל סוגי ההורדות.

בהתאם לסוג ההורדה, מצורפים פרמטרים נוספים:

* &limit={} – חיפוש לפי כמות מקסימלית של נתונים.
* &end={}&start={} – חיפוש בטווח זמן מוגדר
* &start={} – חיפוש מזמן מסוים ועד הנתון העדכני ביותר
* ללא פרמטרים – קבלת כל הנתונים על הלווין הספציפי.

סוגי ההורדה השונים:

|  |  |
| --- | --- |
| שם סוג | פרמטרים להוספה |
| All | אין |
| Limit | limit |
| StartTime | start |
| StartToEndTime | start, end |

### תיאור סביבת הפיתוח

במהלך הפרויקט נעשה שימוש במספר סביבות וכלים משלימים, שלכל אחד מהם תפקיד ייעודי בתהליך הפיתוח:

1. **PyCharm** – שימש לפיתוח ולתכנות צד שרת בשפת Python.
2. **Visual Studio** – שימש לפיתוח קבצי האתר. כולל CSS, HTML, JavaScript.
3. **Visual Studio Code** – שימש ליצירה ולעריכה נוחה של קובצי JSON.
4. **SQLiteStudio** – שימש לבדיקת בסיס הנתונים ולאימות שהמידע נוצר ונשמר כראוי.
5. **Google Developer Tools** – שימש לבדיקת קוד JaveScript ולתצוגה ועיצוב בזמן אמת של רכיבי האתר.
6. **SatNOGS** – משמש כמקור הנתונים המרכזי של המערכת, ממנו מורד וממוין המידע. כלי זה אינו סביבת פיתוח כשלעצמו, אך מהווה רכיב מהותי בפעולת התוכנה.
7. **CMD** – בו נוצר ה-self-signed certificate עבור תקשורת HTTPS.

**שרת -> לקוח**

בכל בקשה השרת פועל לפי הסדר הבא:

1. בדיקה האם מדובר בפעולה.
2. אם לא – ניסיון לשלוח קובץ.
3. במקרה של כישלון, מוחזרת שגיאת 404.

**בחירת לוויין:**

בעת בחירת לוויין, השרת מחזיר ללקוח:

* קטע HTML המכיל את ה-packets בפורמט הצגה (25 packets)
* כלל הנתונים נשלחים בפורמט JSON, כולל גם את חותמות הזמנים.

שדות ב-JSON:

1. **mostRecent** – זמן הלוויין העדכני ביותר שנשלח ללקוח.
2. **leastResent** – זמן הלוויין הכי פחות עדכני שנשלח ללקוח.
3. **data** – קטע ה־HTML המכיל את הנתונים להצגה.

**קבלת נתונים חדשים (למעלה):**

כאשר מתקבלת פקודה לכך השרת מחזיר

* קטע HTML המכיל את ה-packets בפורמט הצגה (כל אלו שיותר גדולים מהזמן שבלקוח)
* כלל הנתונים נשלחים בפורמט JSON, כולל גם את חותמת הזמן החדשה.

שדות ב-JSON:

1. **mostRecent** – זמן הלוויין העדכני ביותר שנשלח ללקוח.
2. **data** – קטע ה-HTML המכיל את הנתונים להצגה

**קבלת נתונים חדשים (למטה):**

כאשר מתקבלת פקודה לכך השרת מחזיר

* קטע HTML המכיל את ה-packets בפורמט הצגה (25 packets)
* כלל הנתונים נשלחים בפורמט JSON, כולל גם את חותמת הזמן המאוחרת ביותר החדשה.

שדות ב-JSON:

1. **leastResent** – זמן הלוויין הכי פחות עדכני שנשלח ללקוח.
2. **data** – קטע ה-HTML המכיל את הנתונים להצגה

**הורדת נתונים כ-excel:**

בעת בקשה להורדת נתונים, השרת מחזיר קובץ Excel בפורמט ‎.xlsx,‎המכיל את כל הנתונים בהתאם לפרמטרי החיפוש שנשלחו בבקשה.

הקובץ נשלח ללקוח בקידוד Hex.

**דוגמאות פשוטות להעברה:**

A screenshot of a phone

AI-generated content may be incorrect.

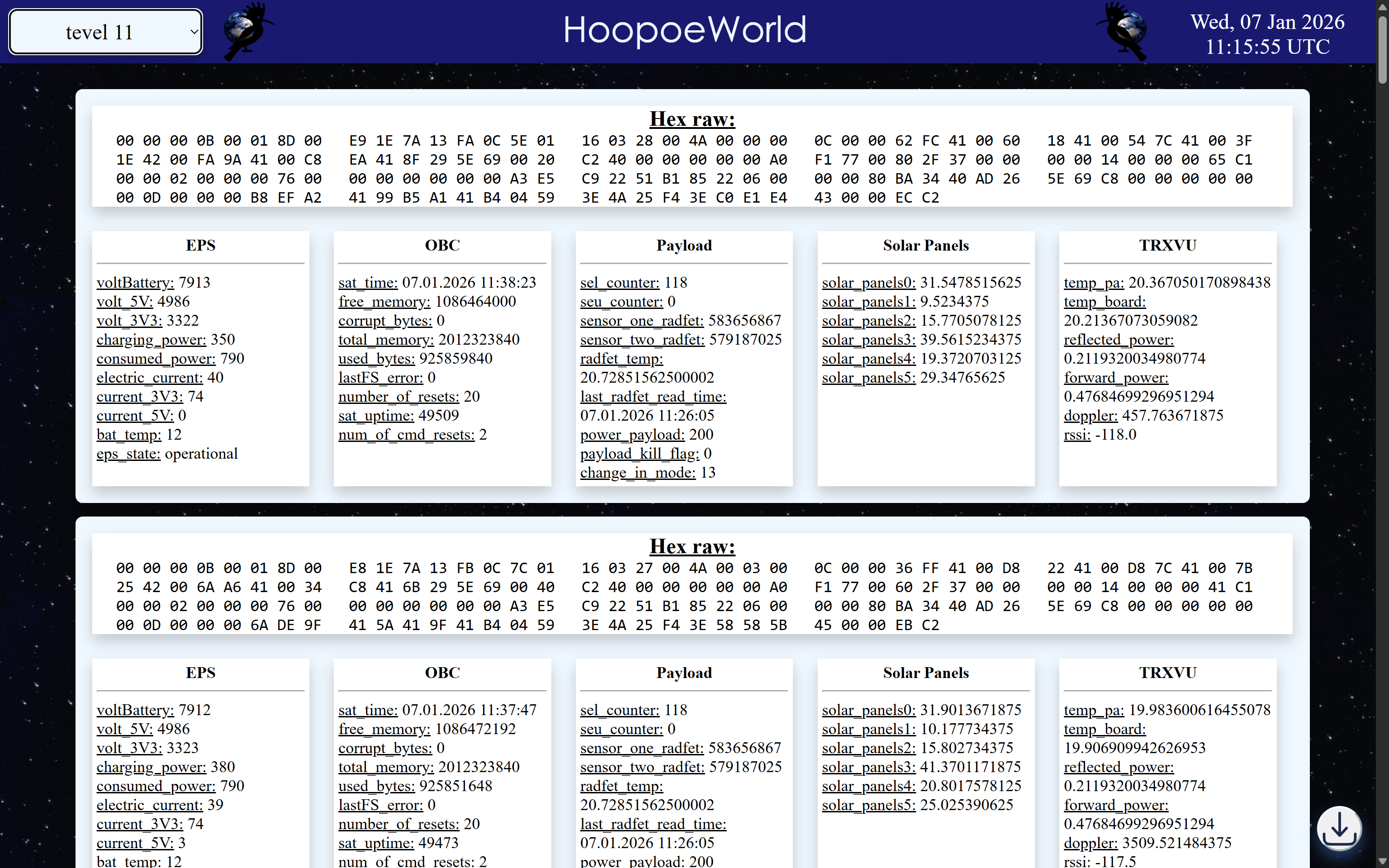
### תיאור מסכי המערכת

יש מסך אחד שהוא המסך הראשי אבל אפשר לראות עליו דברים שונים באמצעות הכפתורים.

כך נראה המסך ההתחלתי. ב-select של בחירת לוויין נמצאים כל הלוויינים שכתובים ב-config ואפשר לבחור ביניהם.

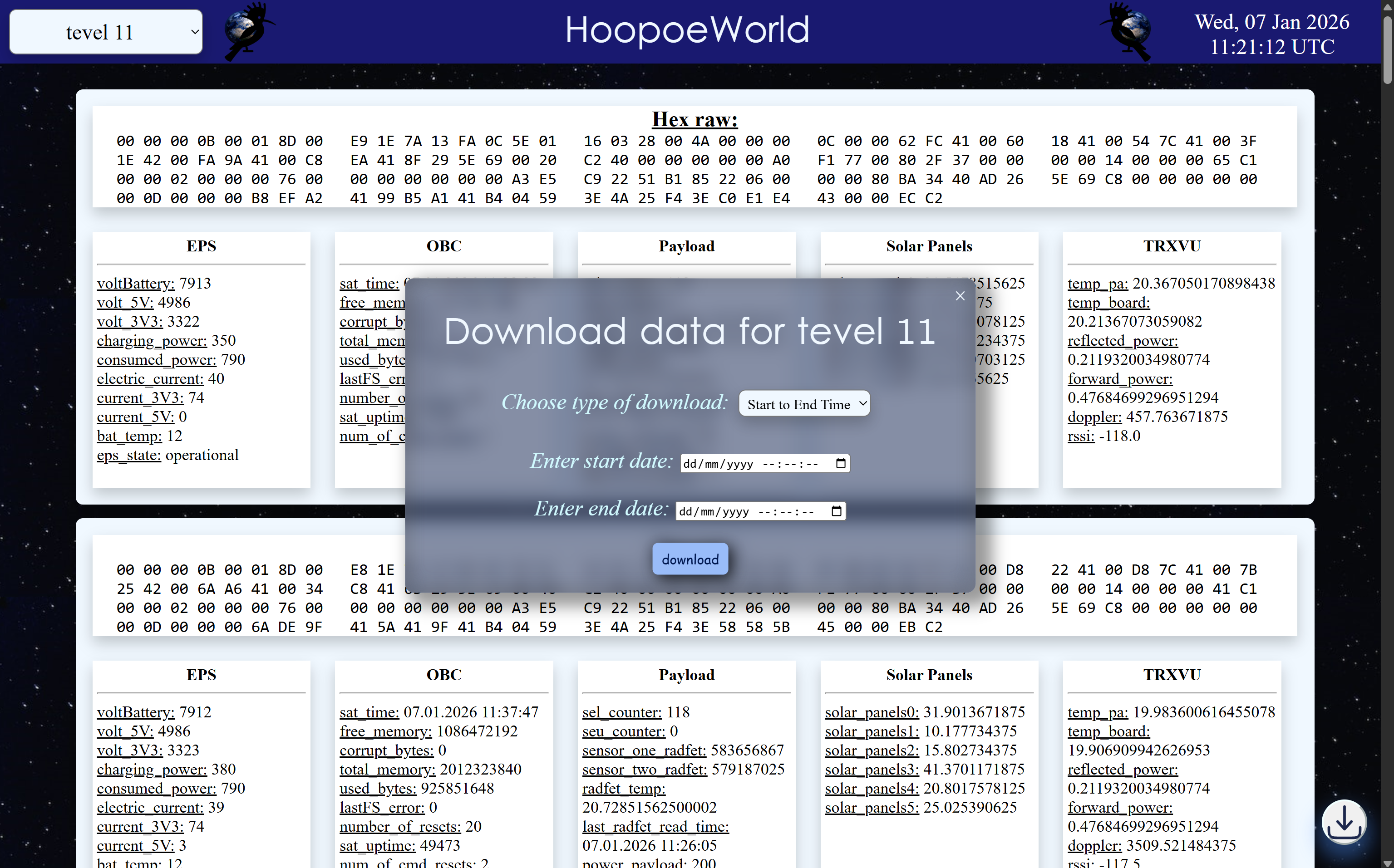
הבחירה של לוויין תוסיף בחלק התחתון את ה-data ואת האפשרות להוריד מידע כ-excel.

בצד יש זמן ב-UTC מכיוון שלרוב מכוונים את הלוויינים לזמן הזה אז שתהיה נקודת זמן.



המסך הזה שנמצא למעלה מראה מה קורה במקרה שנבחר לוויין Tevel 11 ומה יוצג. יעלו ה-packets הכי חדשות שנמצאות ב-database שפוענחו לפי ה-json. כל שעתיים יש ניסיון לעלות עוד פאקטות, שוב את הכי חדשות. כאשר גוללים מספיק למטה הלקוח מבקש אוטומטית עוד פאקטות להצגה והם נוספות בסוף הגלילה.

בצג גם התווסף הכפתור שיראה את האפשרות להורדה ל-excel.



המסך הזה קופץ כאשר לוחצים על הכפתור. אפשר לבחור בו בין אפשרויות שונות של הורדות ללוויין ההוא. האפשרויות הן: start-end, start, limit, all. כל אפשרות תיצור בסוף שאילת שונה עבור השרת לשלוח ממנה.

הבקשה בסופו של דבר תוריד קובץ excel למחשב לפי המיון מה-SQL שנשמר בשרת.

### תיאור מבני הנתונים

בתוכנה יהיו מספר מבני נתונים אך הם דינאמיים אבל בכללי:

* Json – config – מחבר בין ה-SQL, הכתובת ממנה לוקחים את הדברים ללוויין הספציפי, שם הלוויין, כתובת ה-json של הפיענוח.
* Json – ספציפי לכל פיענוח לוויין – מחזיק את הפיענוח של ה-Beacon. מה אמור להיות בו, את מפתח ה-SQL, גודל ה-header, מיקום ה-opcode ב-header והאם big/little endian.
* SQL – אחד לכל לוויין – כל המידע המפוענח נשמר שם. כאשר קוראים ממוין לפי זמן לוויין.

ל-jsons יש פורמטים נחוצים. הפורמט לכל json יהיה בעמוד הבא.

newestTime:

(not a must at start but it’s recommended)

{

{satNogs\_name}: “*most current time got from there*”

}

הקובץ ייווצר בכל מקרה אם הוא לא קיים אך העדפה שיהיה התחלה ולא מ-0, או שיהיה ישירות אחרי שיגור הלוויינים מכיוון שיכול להיות שהחיבור שלנו לאתר יקרוס ולא נוכל לקבל עוד לתקופה.

אם רוצים מההתחלה ממליצה להוריד ב-csv, לקחת ממנו את התאריך הכי מאוחר ולהשתמש בפונקציה שנמצאת בהערה כדי להוסיף את זה למקום הרצוי.

Config:

{

    "{satName}": {

        "callsign": "{sat\_callsign}",

        "tableName": "{what the sql table will be called}",

        "satnogs\_name": "{how the sat is called in satnogs}",

        "beacon\_json": "{the json of the data/beacon format}"

    },

    "{satName2} ": {

        "callsign": "{sat\_callsign}",

        "tableName": "{what the sql table will be called}",

        "satnogs\_name": "{how the sat is called in satnogs}",

        "beacon\_json": "{the json of the data/beacon format}"

    }

}

אפשר להוסיף כמה לוויינים שרוצים, זה דוגמא עם 2, אך כל הדברים כאן נחוצים שיהיו בשניהם. (ה-beacon\_json יכול להיות אותו אחד).

Beacon\_json:

{

“setting”: {

"isDefaultBigEndian": false,

 "opcode": "*opcode in hex (type + subtype)*",

 "sizeof\_header": *full size of header it’s an int*,

 "place\_start\_opcode": *counter start at 1. It’s an int*,

 "prime\_key": "*which param is the primary key*"

}

"subType": {

"name": "*name of function (most likely beacon)*",

"params": [

{

"name": "{name of param}",

"type": "{type of param}",

"subSystem": "{if we have a system it is about} - from here included not a must",

"enum": "{if the type is byte and want to display enum}",

"format": "{in case we have changes for the param (x). can do +, -, \*, /, \*\* (for power)}",

"isBigEndian": false

},

{

"name": "*second param, going according to the order in packet*",

"type": "{param type}"

}

]}}

יכולים להיות כמה פרמטרים שרוצים (יותר נכון כמה שיש בפאקטה), בפעם השנייה מוצגים כל הדברים שחובה שיהיו בכתיבה של פרמטר.

### סקירת חולשות ואיומים

1. **תלות במערכת SatNOGS**  
   במקרה של קריסה באתר SatNOGS, לא ניתן יהיה לקבל נתונים חדשים. מאחר שמדובר בגורם חיצוני, אין אפשרות לטפל בתקלה זו באופן ישיר, מלבד טיפול בשגיאות בקוד (כגון שימוש במנגנוני ‎try-catch) על מנת למנוע קריסת המערכת המקומית.
2. **טעויות אנוש בקבצי ה-JSON**המערכת אינה תמיד מסוגלת לזהות טעויות הנובעות מהזנה ידנית של נתונים. כתוצאה מכך, ייתכן שקובצי ‎JSON הכוללים מבנה שגוי או נתונים לא תקינים יתקבלו על ידי המערכת ויגרמו לבעיות בפעולתה.
3. **שימוש בתעודת HTTPS עצמית (self-signed)**מאחר והמערכת משתמשת בתעודת HTTPS מסוג self-signed, נדרש חידוש ידני של מפתח ההצפנה אחת לשנה. תהליך זה מהווה נקודת חולשה פוטנציאלית, הן מבחינת אבטחה והן מבחינת תחזוקה.
4. **קבלת נתונים שגויים מצד גורמים חיצוניים**במידה וגורם חיצוני מבין את פרוטוקול התקשורת, קיימת אפשרות לשליחת נתונים שגויים למערכת. עם זאת, איום זה צפוי להשפיע בעיקר על חוויית אותו הגורם והצגת המידע אצלו אך לא אצל כלל המשתמשים, ולא לגרום לקריסת השרת, מאחר ואין אפשרות להעלות מידע מהלקוח לשרת.

התוכנה משתמשת בפרוטוקול מוצפן HTTPS, כך ששכבת התעבורה היא TCP.