

המחלקה להנדסת תוכנה

פרויקט גמר – יוני 2018

פיתוח פרוטוקול כללי להעברת מידע ברכיבי IOT
Development of data transfer protocol for IOT components

מאת: אריק לוי וגיא מימוני

1



יוני, 2018

תמוז, תשע"ח

המחלקה להנדסת תוכנה

פרויקט גמר – תשע"ו

פיתוח פרוטוקול כללי להעברת מידע ברכיבי IOT
Development of data transfer protocol for IOT components

מאת: אריק לוי 303175640

גיא מימוני 200481638

מנחה אקדמי: דר' גיא לשם אישור: גיא לשם תאריך: 13.6.18

רכז הפרויקטים: דר' אסף שפיינר אישור: תאריך:

תקציר

הפרויקט הוא פרויקט מחקרי בתחום עיבוד נתונים ורכיבי IOT .

הפרויקט יועד למחקר ופיתוח פרוטוקול כללי לניתוח, סינון ושידור נתונים ברכיבי IOT אשר קולטים נתונים מחיישנים פיזיים.

מטרתו לסייע לחברות וארגונים המיישמים את טכנולוגיית ה IOT במערכות שלהם ורוצות לצמצם את שימוש המשאבים והעלויות שלהם בעת תפעולם.

בהמשך המסמך יוצג המבוא לעולם ה IOT והמהפכות הטכנולוגיות שייגרמו באמצעות שימוש בתחום הזה, תוצג הבעיות הקיימות שהובילו למחקר ופיתוח פרויקט זה, לאחר מכן נציג את הדרכים המוצעות לפתרון הבעיות באמצעות הפרוטוקול שלנו.

לסיום נציג את מהלך ההתקדמות שלנו במשך השנה לגמר המחקר והפיתוח שלנו בנושא שבחרנו.

הצהרה

העבודה נעשתה בהנחיית ד"ר גיא לשם
עזריאלי – המכללה האקדמית להנדסה
בירושלים
המחלקה להנדסת תוכנה.

החיבור מציג את עבודתנו האישית
ומהווה חלק מהדרישות לקבלת תואר ראשון
בהנדסה.

תודות

ברצוננו להודות בראש ובראשונה למנחה
האקדמי

שלנו, ד"ר גיא לשם, על ההכוונה והסיוע
המפורטים

במהלך תכנון ומימוש הפרויקט.

תודה רבה לאסף שפיינר, רכז הפרויקטים על
עזרה

במינוף הפרויקט ושיפורו.

הצדקה

בחירתנו לעשות פרויקט זה בזוג נבעו מכמה סיבות:

- נפח הפרויקט ורמתו מצריכה עבודה של שני מהנדסים. טרם התחלת הפרויקט נחשפנו לנושא ה IOT ולעבודה עם ממשק המותאם לשבב ה Linkit, שניהם היו זרים לנו ולכן נדרשנו לבצע מחקר רב וכמובן גם פרקטיקה. כזוג יכולנו לבצע חלוקת עבודה, להתייעל בחיפוש חומרים וקריאתם, להתייעץ ולדון על כל רעיון/ממצא ולהתקדם הן בפן התיאורטי והן בפן המעשי בו זמנית.
 - לגיא היה ניסיון במסגרת לימודים והשלמת בגרות אלקטרוניקה וזה סייע לנו רבות בפרויקט הנ"ל בנושאים של בניית חיבורים פיזיים באמצעות כבלים מהחיישן אל רכיב ה IOT. לאריק היה ידע וניסיון קל בעבודה עם שפת Python וממשק לינוקס המותאם לשפה הזו (בשפה זו פיתחנו את הפרוטוקול). עבודה כזוג נועדה לייצל ולשפר את המחקר והפיתוח של הפרוטוקול.
- חשוב לציין שחלוקת התפקידים בינינו נעשתה באופן שוויוני – שנינו ביצענו מחקר מעמיק ושנינו עבדנו על פיתוח האלגוריתם לסירוגין.

תוכן עניינים

3.....	תקציר
4.....	הצהרה
5.....	תודות
6.....	הצדקה
7.....	תוכן עניינים
8.....	מילון מונחים
9.....	מבוא
12.....	תיאור הבעיה
12.....	דרישות ואפיון הבעיה
14.....	האתגרים הטכנולוגיים
15.....	הבעיה מבחינת הנדסת תוכנה
16.....	תיאור הפתרון
18.....	הסבר כללי על הפרוטוקול
19.....	תיאור האלגוריתם
21.....	תהליכים ונתוני המערכת
22.....	תיכון המערכת – תרשים מחלקות
23.....	תיאור הכלים המשמשים לפתרון
25.....	תיאור הפרוטוקול שמומש
26.....	תכנית בדיקות
26.....	בדיקות הפרוטוקול במצב שגרה
31.....	בדיקות הפרוטוקול במצב תקלה
34.....	סקירת עבודות הקשורות והשוואה לספרות
38.....	סיכום מסקנות
39.....	נספחים
39.....	רשימת ספרות \ ביבליוגרפיה
40.....	קישורים למערכות ניהול הפרויקט ובקרת תצורה וסרטון
41.....	תרשימים וטבלאות
43.....	תכנון הפרויקט
44.....	טבלת סיכונים
45.....	טבלת דרישות
46.....	Abstract

מילון מונחים

IoT – "האינטרנט של הדברים", הוא רשת של חפצים פיזיים, או "דברים", המשובצים באלקטרוניקה, תוכנה וחיישנים המאפשרים תקשורת מתקדמת בין החפצים ויכולות איסוף והחלפת מידע. רשת זו צפויה להוביל לאוטומציה בתחומים רבים. האינטרנט של הדברים כולל בין השאר את תחומי " הבית החכם "ו"העיר החכמה".

כמו כן, האינטרנט של הדברים כולל טכנולוגיות שמאפשרות למשל: ניטור שתלי לב, שבבים המותקנים על חיות משק לצורכי ניטור ומעקב, כלי רכב המצוידים בחיישנים מובנים, התקני שטח המסייעים לכבאים בפעילויות חילוץ והצלה ועוד.

השאיפה הגדולה של הטכנולוגיה היא שבעתיד ניתן לחבר כל התקן חשמלי לאינטרנט וליצור מערכת ורשת חכמה אשר תנתר ותפקח על כל המכשירים.

"האינטרנט של דברים מבטיח מהפך הרבה יותר נרחב, לא רק שימוש באינטרנט כלשלו רחוק אלא יכולת אוטונומית של מכשירים לתקשר אחד עם השני. דוגמה טובה להתפשטות של חיישנים חכמים היא מוני החשמל החדשים המאפשרים תקשורת דו כיוונית בין הצרכנים לרשת החשמל. בעזרת חיישנים אלו יוכלו הצרכנים לדעת כאשר ישנו מחסור בחשמל ואף לכבות באופן אוטומטי מכשירים זוללי אנרגיה בשעות בהן יש עומס על הרשת."

8

מערכת - המונח הזה מוגדר בשבילנו כמערכת עליה יופעל ויבוצע שימוש בפרוטוקול הכללי שלנו. המערכת יכולה להיות רשת של שבבי IOT המחוברים לחיישני לחץ ומד טמפרטורה בצינור גז או במאגר מים ממוחשב.

בדו"ח זה נשתמש במילה הזו מספר פעמים רבות וכוונתנו תהיה ההסבר הנ"ל.

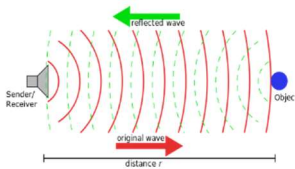
Linkit smart 7688 duo – שבב IOT של חברת Mediatek, השבב בעל מעבד, זיכרון Ram ומטמון, כמו כן לשבב קיימים חיבורי micro-usb, Gpio pins ועוד. מפרט המכשיר:



- זיכרון של 128 מגה בייט,
- מעבד Single core , 580 MHz ,MIPS24KEc – 32 bits
- אפשרות להרחבת זיכרון עד 512 מגה בייט – Micro SD Slot
- חיבור USB 2.0
- תומך ב – Wi-Fi
- מחיר: 16\$

על שבב זה אנו נמקד את בניית האלגוריתם שלנו.

WinSCP - פלטפורמה נוחה וקלה לשימוש המשמשת אותנו להתחברות מהירה עם השבב דרך רשת מקומית משותפת והעברת קבצים לתוכו (בעיקר קבצי תכנית בשפת פיתון). אנו נפרט עליה בהמשך המסמך.



HC sr04 – חיישן מרחק, החיישן עובד בכך שהוא שולח גלי סונר על ידי מרכיב ה Output אל השטח הפתוח וכאשר הוא נתקל באובייקט, גלי הסונר חוזרים חזרה ונקלטים במרכיב ה Input. (להלן תמונה של החיישן כאשר גליל מתכת אחד משדר גל סונר והשני קולט את הגל החוזר, להלן גם המחשה של התהליך).



השתמשנו בחיישן זה לחיבור עם רכיב ה IOT ומדידה של דגימות למטרת הפרויקט. החיבור נעשה באמצעות לוח ניסוי (מטריצה) לחיבור רכיבים אלקטרוניים באמצעות כבלי מתח ופינים.

מבוא

9

עד לפני עשור נעשה החיבור לאינטרנט מהמחשב הביתי בלבד. בשנים האחרונות גדלה עשרות מונים כמות ה"דברים" המחוברים לאינטרנט: הטלפון החכם, הטלוויזיה, שעונים חכמים, משקפיים, מכוניות ומוצרי חשמל ביתיים. כולם הופכים את עולמנו לעתיר מידע, לעולם שבו ניתן לדעת הכול על הכול בכל רגע ובכל מצב. כיום מחוברים לאינטרנט של הדברים כ-6 מיליארד חפצים, וההערכות הן שמספרן יגיע עד שנת 2020 לכ-21 מיליארד, מדובר בכשלושה חפצים בממוצע לאדם באוכלוסיית כדור הארץ.

המהפכה החלה לפגוש אותנו בתחומים שונים ומגוונים בחיינו, בין היתר, בית חכם, רפואה ורכבים אוטונומיים.

רוב רכיבי ה IOT המחוברים לרכיבים פיזיים כגון מכונות כביסה, צינורות גז, מכוניות ועוד, משמשים לניטור וניתוח נתונים כמובן על מנת לייעל את התפוקה ולהוריד מנטל הפיקוח והניטור האנושי, כדי שזה יקרה בדרך כלל מחוברים ישירות לרכיבי ה IOT חיישנים המודדים נתונים בזמן אמת ומשדרים את הנתונים לרכיבים.

כל חיישן פועל בצורה שונה, בעל ממשק ומערכת הפעלה שונה ולכן יש צורך להכיר אותו ואת אפשרויות החיבור שלו אל רכיבי ה IOT.

הפרויקט שלנו התחיל בדיונים עם המנחה שלנו על רקע עולם ה IOT ושימושיו בתעשייה, במהלך הדיונים שלנו הרצנו כמה סיטואציות ותסריטים בהם פועלים רכיבי ה IOT בתעשייה במתקנים ומערכות שונות.

אחד התסריטים שבחרנו לשים עליו דגש הוא מתקן אסדת קידוח היושב בלב ים, מנוטר ומפוקח היטב על ידי מרכז בקרה עם שימוש נרחב בחיישנים ורכיבי ה IOT, הניטור מבוצע בדרך כלל בחיישני לחץ וטמפרטורה על צינורות גז ומקומות קריטיים במתקן, אך יש מגבלה מסוימת, רוב האסדות יושבות באזורים מבודדים ללא קליטה סלולרית או של כל תקשורת נתונים כלשהי.

ניטור ופיקוח הנתונים הוא קריטי וחיוני להמשך תפעול שוטף של המתקן, שכן שגיאות בודדות יכולות לגרום לקריסת מערכות והשבתה כוללת של המתקן כולו ואף אפילו לסכנת חיים לכוח אדם הפועל בו.

היעדר קליטה סלולרית בשל המיקום המבודד והרצון לעשות את המערכות האלו כמה שיותר לשימוש מרוחק ואוטומטי (על מנת לחסוך במשאבים ולהתייעל) – יש צורך במערכת תקשורת שתדע להעביר את המידע במהירות ובמלואה אל הגורמים המוסמכים (קרי: מרכזי בקרה),

ולכן, יש צורך בשימוש במערכת תקשורת לוויינית המסתמכת על לוויין תקשורת אשר נמצא באטמוספירה של כדה"א.

שימוש בלוויין תקשורת הוא יקר ומצריך שימוש במשאבים כספיים וטכנולוגיים רבים, מערכות אלו צריכות להסתמך על תקשורת זו ולכן החברות אשר בבעלותן המתקנים מבזבזות כסף רב רק על ניתור והעברת הנתונים.

אתגר נוסף שיש להתמודד אתו על מנת שלא כל המידע יעבור הוא הצורך בשרתי אחסון מידע רבים בנקודת היעד אליה מגיע המידע מהמערכות.

כמו כן, היות ועובר מידע רב יש צורך ברוחב פס ענק במערכת התקשורת הלוויינית, שאין צורך לספר כמה היא מוגבלת היות ולא קיימים מספר רב של לווייני תקשורת לשימוש פרטי.

בפרויקט זה נראה פתרון אפשרי יעיל, חסכוני ומהיר לאותן בעיות אשר סופר מקודם.

על מנת לטפל בבעיות נאלצנו לחקור רבות, להתנסות ולהתמקצע בצורה מקיפה ויסודית כדי להגיע לשורשי הבעיות כדי לקבל פרספקטיבה ודרך לפתרון.

המחשבה שלנו שהובילה אותנו לאורך כל הדרך היא לטפל בבעיה ברמה המקומית משמע

לסנן ולחסום את תעבורת המידע הלא נחוצה ולאפשר העברת מידע במקרים מסוימים

ספציפיים עליהם נרחיב בהמשך, כדי לעשות זאת היה עלינו לנצל את יכולות המחשוב בעלות הפוטנציאל הגבוה של רכיבי ה-IOT, רכיבים אלו מתנהגים ופועלים בדיוק כמו מחשב.

אמנם, רכיבים אלו מוגבלים ביכולות העיבוד ואחסון הנתונים שלהם, אך עדיין ניתן לבצע

באמצעותם חישובים והרצות אלגוריתמים בצורה דיי מהירה ואמינה.

המימוש שלנו לפתרון יהיה פרוטוקול כללי אשר יקלוט מידע מהחיישנים, יסנן מידע לא נחוץ

ויעביר מידע רק במצב שיוגדר על ידנו או שיוגדר על ידי האלגוריתם.

מצב זה ייתפס כשינוי מצב של המערכת, שינוי מצב מבחינתנו ומבחינת האלגוריתם מסמל

בעיה או תקלה אפשרית בפעולה השוטפת של המערכת.

בנוסף, על מנת לצמצם עוד את תעבורת המידע וליצור צוואר בקבוק בתקשורת הנתונים אנחנו

נממש בנוסף מערכת תקשורת בין החיישנים שתהיה מורכבת מרשת של שבבי IOT שיוגדרו

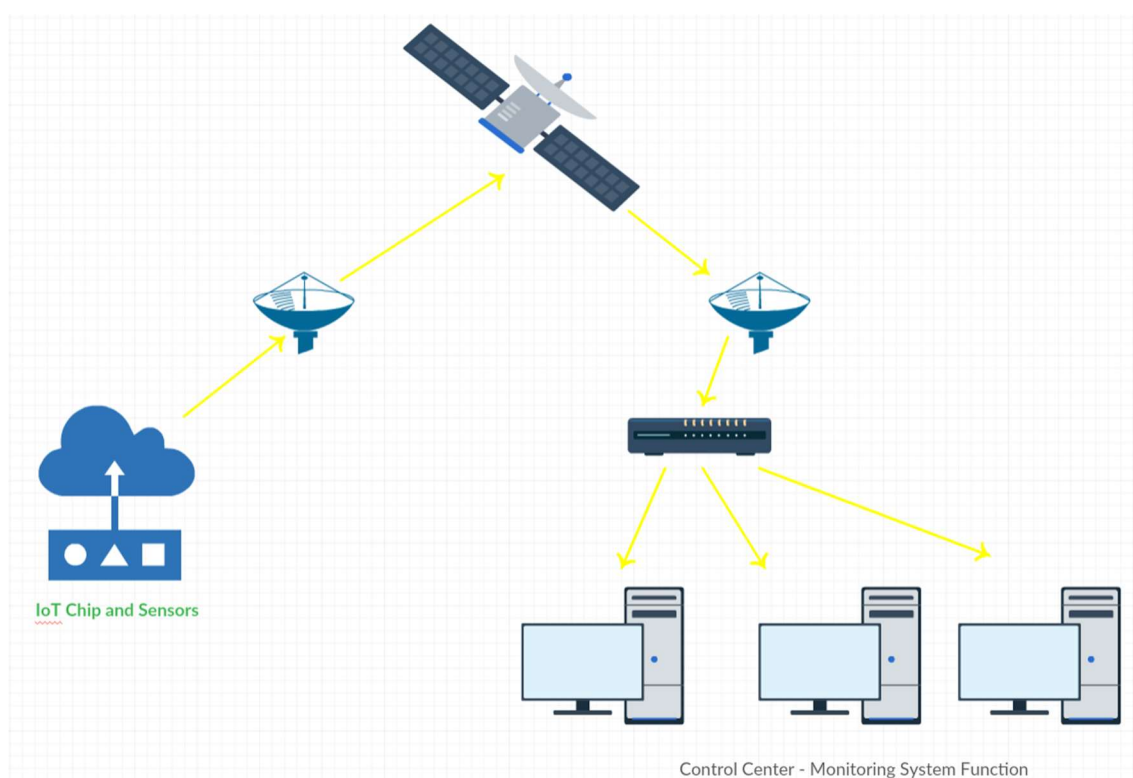
כסלייבים ושלב אחד בודד שייבחר כמאסטר ודרכו יועבר כל המידע אשר משודר ברשת.

בצורה זו של מימוש הפרוטוקול אנו מקווים שנוכל לחסוך את עלויות שימוש במערכת תקשורת

לוויינית והצרת השימוש בשרתי אחסון רבים כדי לאחסן את המידע המתקבל מפעולות

המערכת.

להלן תרשים המדגים את מערכת התקשורת הלוויינית הקיימת כיום במערכות אלו.



ולפני שעוברים לתיאור הבעיה והמשך הדו"ח, אנו חייבים לציין ולהדגיש שהפרויקט שבחרנו לעשות הוא **פרויקט מחקרי**, לכן, האתגר הכי גדול שלנו היה המחקר. המחקר הרב שביצענו במשך כל השנה, הניסויים הרבים עד שמשהו קטן עבד במערכת שלא הכרנו לפניכן, בשבב שלא היה לנו התנסות לפניכן ובממשקים ותוכנות שלא היה לנו נגיעה לפניכן, כל אלה היו לנו כמסלול מכשולים, אך, אט אט הצלחנו לעבור כל משוכה בהצלחה אבל עם הרבה עבודה קשה וכן גם לא מעט טעויות. לסיכום, נצטט משפט חכם שאמר תומאס אדיסון ומסכם בצורה יפה את דרכנו: "אם מצאתי אלף דרכים שלא עובדות, לא נכשלתי. כל ניסיון שכשל הוא עוד צעד קדימה".

תיאור הבעיה

נתאר את הבעיה אותה עתידה לפתור הפרוטוקול שלנו, נתאר בתרחיש מסוים מהעולם האמיתי: ישנו בן-אדם חיוני וקריטי לארגון מסוים, הוא מחויב על ידם להגיע למקום ספציפי, האדם הנ"ל נמצא באזור מבודד גיאוגרפי ועל כן הוא חייב להשתמש בתחבורה מיוחדת, התחבורה הזו עולה כסף רב לאנשים שביקשו מהאדם להגיע לנקודת היעד, כדי שהאדם יגיע ללא דיחוי, התחבורה חייבת להיות זמינה וכמובן אמינה, עכשיו כדי לחזור לעולם שלנו נשנה את האדם לנתונים, ובשפע.

כדי להתמודד עם הנתונים העצומים, אנו צריכים "תחבורה" רבה – משמע אנו צריכים קווי תקשורת היכולים לספק תעבורת מידע עצומה, מהירה ואמינה.

השאלה הנשאלת היא, איך מעבירים מידע רב כדי שהמערכות האלו ימשיכו לרוץ ולעבוד בצורה שוטפת?

ניכרת בעיה בשימוש ברשת סלולרית או בשימוש בכבלי רשת, ייווצר אתגרים הרי חלק מהן נמצאות בלב ים – חוטי חשמל ותקשורת לא יכולים להגיע לשם, תשתיות כבלים תת-ימי מסובכות ומצריכות משאבים רבים כגון כסף, כוח אדם וחומרי גלם. לכן, הפתרון האופטימלי הוא שימוש ברשתות תקשורת יקרות ומורכבות, כגון תקשורת לוויינית. בנוסף, לכל הנתונים האלו שמגיעים מהחיישנים (והמערכות ומגיעים הרבה) לנקודת היעד – "מרכזי הבקרה", היות והנתונים רבים ומסיביים, יש צורך בשרתי אחסון רבים עם יכולות אחסון מרבית - גם הם יקרים מאד מבחינה כספית.

12

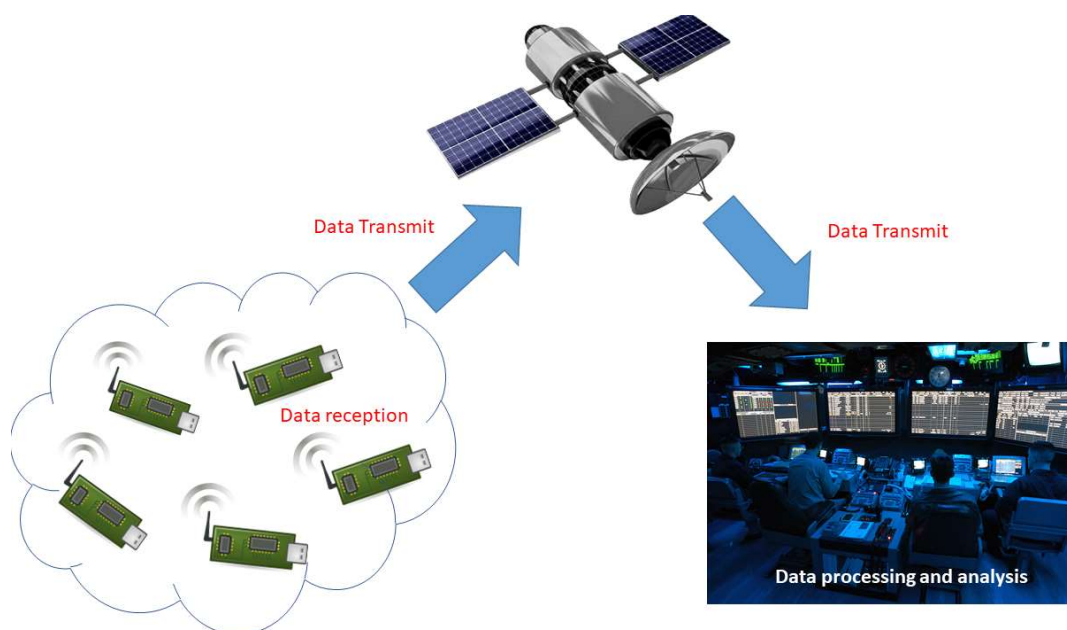
הסברנו רבות שיש צורך לשמור את הנתונים, אבל מה הם הנתונים?

הנתונים הם דגימות, תוצאות ומדדים אשר מתקבלים מהחיישנים המנתרים את המערכות הפיזיות, דוגמא למערכת פיזית: צינור גז והנתונים הם: מדידות טמפרטורה. נתונים אלו חשובים וקריטיים לגורמים האנושיים המוסמכים שידע לנתר ולבצע אנליזות על מנת לאפשר פעולה שוטפת של המערכת וגם לקבל מסקנות והחלטות שיסייעו בשיפור המערכת, בין אם זה לווסת את זרימת הגז או להגביר את הזרימה או כל דבר אחר שגורם מוסמך מכיר.

דרישות ואפיון הבעיה

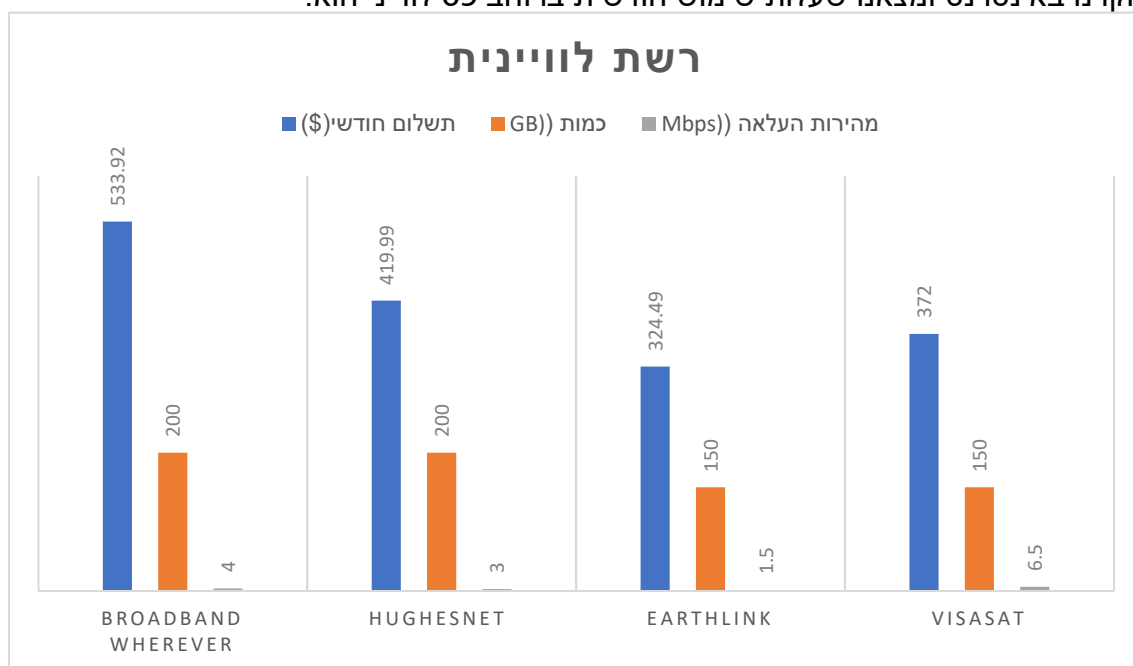
המטרה העיקרית של הפרוטוקול הינה לטפל בבעיות ברמה המקומית, כלומר הפרוטוקול יותקן על השבבים ויבצע את כל הקליטה, עיבוד, סינון ושידור בתוך השבב ובכך מונעים ממידע רב לעבור סתם.

מצורף התרשים הבא כדי להמחיש את התהליך שקורה במצב הנוכחי מהרגע שנקלטים הנתונים ועד הרגע שהם מגיעים למרכזי הבקרה שם הם מעובדים ומנותחים.



רצינו לבדוק לעומק את הנושא עלויות של רשתות לווייניות כדי לקבל מסקנות ומידע נוסף.

חקרנו באינטרנט ומצאנו שעלויות שימוש חודשית ברוחב פס לווייני הוא:



הסבר על התרשים: לאחר חקירה מקיפה באינטרנט מצאנו את 4 הספקיות רשת לוויינית הכי פופולריות שיש.
מצאנו שהמחירים נעים בין 325-532 דולר לרוחב פס חודשי של GB 150-200, כמו כן, מהירות ההעלאה נעה בין 1.5-4 מגה-בייט לשנייה. עד כאן נראה שזה לא כזה משמעותי. (המחיר הנ"ל מדבר על תשלום חודשי נטו, אך יש הוצאות נוספות כגון התקנה ואחזקה של נתבים ואנטנות ועוד..)

אך כמה רוחב פס בכלל צריכים המערכות המבודדות עליהן דיברנו?

בכתבה שקראנו (להלן [הלינק](#)), נמצא גם בנספחים) ודובר שם על הצורך ב IOT באסדות קידוח גז ונפט, על מנת לשפר את היעילות ולמזער את זמן ההשבתה הבלתי מתוכנן כאשר קרתה תקלה בקדיחה.
מפרסם הכתבה ציין שחברת סיסקו המפורסמת הסיקה שאסדת גז טיפוסית מבודדת מניבה 1-2TB של מידע כל יום וכ- מיליון דגימות בשנייה, כך שהכמות מגיעה לכ- 30-60TB של נתונים בחודש.
בהנחה שאנו חושבים ביעילות ולוקחים את התשלום הנמוך ביותר לרשת לוויינית שזה 325 דולר לחודש, במחיר זה אנו מקבלים GB 150 רוחב פס (Broadband).
חישוב מהיר וגס יעריך שמדובר ב 1,560,000 דולר בשנה על העברת המידע דרך צינורות הרשת הלוויינית עד ליעד, וזה אפילו ללא חישוב של הוצאות: התקנה, אחזקה ועוד.. של הנתבים, אנטנות, שרתי אחסון, כבלים וקווי תקשורת מקומיים.
יש לציין שמתגלה לנו בעיה נוספת, בהנחה ואפשר להעלות רק 1.5 מגה-בייט בשנייה ואנו מחויבים להעביר 2 טרה בייט ביום, התוצאה היא שאנו נעביר את כל המידע ביום וחצי מאשר יום אחד ולכן חלק מהמידע יעבור באיחור רב.
רוב המידע שעובר הוא מידע שגרתי ותקין. אז למה אנו צריכים להעביר אותו הלאה?

האתגרים הטכנולוגיים

- יכולת העיבוד המוגבלת של השבב Linkit.
מבדיקות והרצות ראשוניות לבדיקת מאמץ למערכת שמנו לב לאיטיות וזמן ריצה ארוך, במקרים מסוימים התוכנית קרסה או נתקעה.
- עבודה עם חיישן חדש וזר לנו, ממחקר רחב באינטרנט לא מצאנו ניסויים ופרויקטים שבוצעו בשילוב רכיב ה IOT שלנו - Linkit וחיישן המרחק HC-sr04, לכן אנו נדרשים לבנות אלגוריתמים מאפס שיצליחו לחבר בין שניהם על מנת שנוכל להמשיך בבניית הפרוטוקול שלנו.
- בניית רשת מרובת חיישנים על בסיס מתודולוגיית שליט/עבדים (Master/Slaves).
על מנת לבנות רשת זו נצטרך לחקור רבות על יישומים למתודולוגיה בין רכיבים ומארחים ברשת המקומית.

הבעיה מבחינת הנדסת תוכנה

זמינות ואמינות - יש צורך בזמינות ואמינות תמידית של המערכת על מנת להימנע ממצבים שמידע קריטי פשוט ילך לאיבוד. אנו רוצים שהמערכת תרוץ באופן תמידי ללא נפילות, כמו כן, נרצה שהמערכת תעבוד בצורה אמינה ללא תוצאות וחישובים לא צפויים.

אחסון המידע - בעיה נוספת שיכולה להתעורר מבחינת הנדסת תוכנה היא בנוגע לאחסון המידע עצמו, בפרט, הכוונה היא למידע המתקבל מהחיישנים המחוברים לרכיבי ה-IOT. רכיב ה-IOT מוגבל בכמות המידע אשר ניתן לאחסן בו, ורוב הדגימות שנרצה לדגום לא "כבדות" מבחינת זיכרון אך שיש מלא דגימות זה כבר סיפור אחר.

הגדרת תקשורת מונחית שליט/עבד (Master/slave) – בנוסף לאלו, אנו נרצה לייעל את תעבורת המידע וליצור צוואר בקבוק על מנת לצמצם את קווי התקשורת בין מרכז הבקרה (נקודת היעד של הנתונים) לרשת הרכיבים. את הייעול הזה נבצע באמצעות מתודולוגית מאסטר/עבד (slave), הרעיון הוא, שיהיה רכיב אחד מסוים שייחשב החזק ביותר והוא יבחר להיות המאסטר ודרכו תבוצע כל תעבורת המידע, שאר הרכיבים ייבחרו כעבדים והם יבצעו את עבודתם בהתחברות לחיישנים, קבלת נתונים, עיבוד הנתונים, ואם נתפס על ידם מצב של חריגה במדדים הם ישדרו את הנתונים אל המאסטר, כמובן שהוא יקבל את הנתונים בצורה מסודרת ויעביר אותם הלאה. הרעיון הוא פשוט להבנה אך הביצוע והמימוש הוא קשה מאד, נצטרך להסתמך על מחקרים קודמים וננסה לממש את הגרסה שלנו בפרוטוקול, תוך כדי מתן תשומת לב לסנכרון תהליכים, שליחת סיגנלים והודעות במקרה ותקלה קרתה בעת עבודת הפרוטוקול כגון מאסטר שנפל או עליית מאסטר חדש חזק יותר. אין ספק שזה החלק המאתגר ביותר בשלב שידור הנתונים והקמת התקשורת בין שאר הרכיבים.

תיכון יעיל של כלל מרכיבי הפרוטוקול באלגוריתם – לפרוטוקול שלנו 3 מערכות בעלי אופי פונקציונאלי (קליטה, עיבוד, שידור), כל מערכת תעבוד ותרוץ באופן אוטונומי ללא תלות בשאר המערכות האחרות, אך הדרישה היא על מנת שאלגוריתם המלא יעבוד בצורה חלקה ללא שגיאות או פעולות לא צפויות. כדי שהפרוטוקול ירוץ בצורה שלא משתמעת לשתי פנים אנו נדרשים לתכנן בקפידה תיכון ותרשימים של האלגוריתם שלנו.

תיאור הפתרון

כדי לענות על הבעיות בהן נקבנו בפרקים הקודמים - חקרנו רבות במחקרים ומאמרים באינטרנט ב-איך לגשת לבעיה ואיך לבצע תוכן מפורט ומקצועי לפתרון שרצינו להגיע אליו בסיום הפרויקט. כשמגיע מלא מידע מהחיישנים, מלא מידע יכול להתפסס וגם מלא מידע יכול להיות מיותר, לכן, בפרויקט זה אנו מציעים לאותם אנשים וארגונים המשתמשים בחיישנים ורכיבי IOT דרך להתמודד עם הכמות מידע המסיבית הזו. הפרוטוקול שיצרנו "מרצה" את תעבורת הנתונים, מייעל את התקשורת וחוסך במשאבים יקרים כמו: שרתים, רוחב פס של תקשורת יקרה ומורכבת ואף כוח אדם מיומן. ניגשנו לפרויקט הזה וחילקנו אותו ל-3 חלקים עיקריים:

- קליטת נתונים.
- עיבוד נתונים.
- שידור נתונים.

1. **שלב קליטת הנתונים** - בשלב זה השתמשנו בחיישן מרחק וחיברנו אותו לשבב דרך חיבורי General-purpose input/output עם כבלי חיווט ומטריצת חיבורים אלקטרונית. כדי להביא את הנתונים הנקלטים מהחיישן בצורה מידית ואמינה ביצענו מספר רב של ניסויים ובדיקות עד שהגענו לתוצאה הרצויה והצלחנו לשמור את הנתונים כדי שהאלגוריתם יוכל בהמשך להשתמש בהם. הדגימות שלקחנו הם של מרחקים אשר נקלטו בחיישן בזמנים שהגדרנו ולפי תסריטים שביצענו ושנפרט עליהם בהמשך במסגרת תכנית הבדיקות.

2. **שלב עיבוד הנתונים** - בשלב זה לקחנו את הנתונים ששמרנו, ביצענו חישובים לעבד את הנתונים ולאבחן אם נמצא שינוי במצב הקיים. המצב הקיים מבחינתנו הוא כמות מסוימת של דגימות שמסמלת לנו תבנית התחלתית ומהווה לנו מדד למצב שגרה תקין. בשלב הראשוני, האלגוריתם מחפש אם קיימת תבנית התחלתית, אם האלגוריתם לא מצא תבנית כלשהי שקיימת, הוא לוקח מדדים בזמן אמת לפי סדר זמנים שהוגדר לו על ידי המשתמש ובכך בונה תבנית התחלתית שתהווה אבן דרך למצב תקין. בשלב הבא המערכת לוקחת כל פרק זמן מסוים ומוגדר מראש דגימה, ולפי חוקים שהוגדרו מראש (שבין היתר הגדירו גם איך לשלוף את התבנית ההתחלתית) מגדירה תבנית עדכנית של הנתונים והדגימות שהתקבלו. לאחר מכן, מבוצע שלב ההשוואה – השוואת התבניות ומציאת הסטייה מהמקור. כאשר באחת הדגימות העדכניות נמצאה סטייה במדידה לעומת התבנית ההתחלתית, האלגוריתם משדר את הנתונים למאסטר באמצעות תקשורת שרת/לקוח על ידי Sockets ותהליכונים (Threads) על מנת שהאלגוריתם יוכל להמשיך לקלוט ולהשוות נתונים.

3. **שלב שידור הנתונים** – בשלב זה האלגוריתם משדר את המדידות אל מרכז הבקרה.

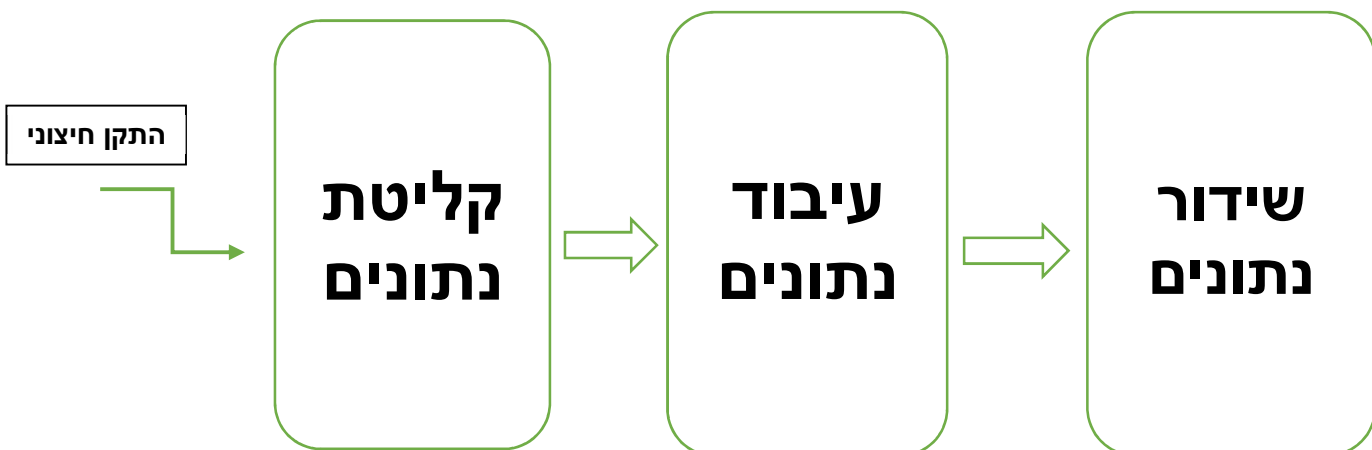
כשהגענו לשלב הזה, חשבנו על כך שאם יש לנו רשת רחבה של מלא חיישנים – לשם הדוגמה במאמר שהבאנו על אסדת קידוח, ביצעו שם רישות וחילוץ של 600 רכיבי IOT שיוכלו לנתר ולנתח כל פרט קטן במתקן הענק הזה, בכל פעם שיש שינוי מצב ברכיב כלשהו אז הוא ישדר הלאה למרכז הבקרה, כך שיוכל להיווצר מצב ש- 600 רכיבים ישדרו באותו זמן למרכז אחד, זהו מצב לא אידיאלי ולא פותר לנו שום בעיה, בנוסף, לכל רכיב נצטרך להגדיר כתובת IP חיצונית, פרוטוקולי תקשורת חיצוניים להעברת נתונים כמו TCP ו-FTP כך שזה פשוט נראה לא יעיל.

החלטנו לבסוף כדי לייעל את כל עניין שידור הנתונים הוא ללכת על מתודולוגית שליט/עבדים (Master/Slaves).

המטרה היא שאחד מהרכיבים ישמש כשליט וכמעין "צוואר בקבוק" ודרכו תעבור כל ההתקשרות לרשת החיצונית אל היעד – מרכז הבקרה (במקרה שלנו, המשתמש האנושי).

האלגוריתם שמימשנו הגדיר באופן מקומי ואוטונומי אצל כל רכיב מי הוא המאסטר, וכמובן אם הרכיב הנ"ל הוא לא מאסטר אז האלגוריתם הגדיר אותו כעבד (Slave). הבחירה של המאסטר נעשית באופן מסוים שעליה נפרט בהמשך הדו"ח, כמו כן נפרט על אופן פעולת העבד בהקשר לפרוטוקול שלנו.

להלן תרשים זרימה של הפרוטוקול:



הערה: התקן חיצוני – חיישן (מרחק, קול, אור...)

הסבר כללי על הפרוטוקול

הפרוטוקול ירוץ באופן תמיד ויבצע כל הזמן מספר פעולות אשר יבטיחו את המשכו. הפעולות שהאלגוריתם עושה באופן תמידי:

- קליטת נתונים ולקיחת מדידות.
 - סריקה ומיפוי כל הרכיבים המחוברים ברשת.
 - הגדרת רכיב שישמש כמאסטר.
 - בדיקה האם המאסטר עדיין פעיל.
 - הגדרת רכיב חזק יותר שישמש כמאסטר החדש (כמובן, שהמאסטר הישן עובר להיות עבד).
- בנוסף, ישנם עוד כמה פעולות שהאלגוריתם מבצע לכל ישות בנפרד:

שליט (Master) - יוגדר באופן התחלתי בכל תחילת הרצת האלגוריתם, וישתנה רק כאשר הוא נפל – משמע, הרכיב התנתק מהרשת או לא פעיל יותר, בנוסף, המאסטר יתחלף כאשר הגיח לרשת רכיב שהוא חזק יותר.

השאיפה שלנו הייתה לחפש דרך מתוחכמת לבחירת השליט, הרעיון שעלה במוחנו הוא שהשליט ייבחר על פי מידת קרבתו אל נתב הרשת האלחוטית (הרכיבים מחוברים דרך Wi-Fi לנתב) וזאת כי על פי המחשבה שלנו הרכיב הקרוב ביותר לנתב הוא בעל קליטת האינטרנט החזקה ביותר ולכן דרכו יועבר המידע בצורה המהירה ביותר, ככל שחקרנו ברשת ומימשנו גרסת קוד שתוכל לבדוק עבור כל אחד מהרכיבים מה קרבתו הגענו למסקנה שזה יעמיס על הרכיב ועל האלגוריתם וכך כתוצאה מכך הפונקציונאליות תמומש חלקית או בכלל לא.

בנוסף, כאשר ראינו את הפלט לגבי קרבת הרכיבים אל הנתב המרחקים היו ד"י שווים או שונים בצורה מינורית ולכן הבנו שבסיטואציה של מלא רכיבים במיקום סגור ותחום, האלגוריתם לא יעבוד בצורה טובה, לבסוף החלטנו שכדי שהפרוטוקול יעבוד עם ביצועים טובים אנו לא נוסף את האלגוריתם לבחירת מאסטר ונסתמך על כתובות ה IP של הרכיבים, בדקנו את הבחירה בכמה נתבים שונים המגדירים כתובות ה IP שונות לכל רכיב ולכן כל רכיב נבחר להיות שליט. אלגוריתם השליט יבצע פעולות אלו בנוסף:

- האזנה תמידית ברשת לנתונים שיתקבלו משאר הרכיבים שהוגדרו כעבדים.
- שידור הנתונים שהתקבלו מהרכיבים אל מרכז הבקרה.
- שליחת הודעת מאסטר חדש הגיח לכל הרכיבים ברשת, כולל המאסטר הישן.

עבד (Slave) - יוגדר אף הוא בתחילת הרצת האלגוריתם, תפקידו של העבד לא שונה בהרבה מהמאסטר אלא שבמקום לשדר את הנתונים לאחר שנמצאה חריגה הוא משדר את הנתונים אל המאסטר.

כפי שהסברנו מקודם, מטרתנו לצמצם את קווי התקשורת ותעבורת המידע וליצור היררכיה אלגוריתם העבד יבצע פעולות אלו בנוסף:

- שידור חבילת הנתונים אל המאסטר המורכבת משם הרכיב, המדידה החריגה וזמן תפיסת המדידה.
- בדיקה האם המאסטר נפל/לא זמין.
- האזנה – הבאנו את האפשרות שגם העבד יאזין וזה למקרה והמאסטר נפל ורכיב אחר תפס את זה בזמן ולכן הוא משדר לשאר הרכיבים שיעדכנו את סטטוס המאסטר.



כמו כן, שיטת ההאזנה תהיה רלוונטית גם למקרה והגיה רכיב חדש שהחליט שהוא המאסטר החדש ולכן, העבד צריך גם פה לעדכן ברשומות שלו את המאסטר החדש בשביל המשך תקין של האלגוריתם.

מספר הבהרות:

- במקרה של רכיב אחד הזמין ברשת המקומית ומריץ את האלגוריתם - הוא יוכרז כמאסטר כברירת מחדל.
- אם מחובר לרכיב המאסטר גם חיישן אז בוודאי שגם הוא הוא סורק וקולט נתונים ומחפש חריגות.
- עבד לא יוכל להעביר את שאר המידע לשאר העבדים, אך ורק למאסטר.
- האלגוריתם שלנו בהקשר של מאסטר/עבד דומה במהות לתבנית תיווך Mediator שלמדנו במסגרת התואר, המאסטר משחק תפקיד של מתווך ברשת של מנויים (subscribers), בסופו של דבר, הוא לא מנחית פקודות לביצוע לעבדים אלא מאזין ומחכה לנתונים מהעבדים.

תיאור האלגוריתם

במחצית הראשונה של השנה כאשר עסקנו במחקר ובניית הפרוטוקול, הגענו למסקנה שכדי לעבוד בצורה הטובה ביותר אנחנו צריכים לתכנן את האלגוריתם של הפרוטוקול ולהסתמך עליו כאשר נתחיל לפתח אותו בקוד.

חשוב לציין שבהתחלה בנינו את האלגוריתם בצורה בסיסית, לא לקחנו כמה מקרי קצה ואפשרויות מסוימות שיכולות לקרות במהלך ריצת הפרוטוקול, לכן שינינו אותו והוספנו דברים כדי שנוכל להגיע למשהו מוגמר שנוכל להסתמך עליו. בתרשים הבא מוצג האלגוריתם שבנינו לפרוטוקול שלנו. הפרוטוקול מוצג בפסאודו קוד והוא מוצג ככה כדי להעביר את המסר לכלל האנשים המעוניינים לדעת איך בנינו את הפרוטוקול וללא תלות בשפת תכנות מסוימת.

מהלך הפרוטוקול:

- בשלב הראשוני הפרוטוקול מוודא שיש תבנית דגימות התחלתית, אם לא, האלגוריתם מבצע באופן מידי סדרת דגימות כדי להגדיר תבנית ראשונית.
 - לאחר מכן האלגוריתם נכנס ללולאה תמידי (הגדרת פרוטוקול – רץ כל הזמן ומבצע אלגוריתם).
- בתוך האלגוריתם מבוצעות הפעולות הבאות:
1. בדיקה אם הרכיב הוא מאסטר - אם כן האזן למידע שיגיע מהרכיבים האחרים, בנוסף.
 2. לקיחת מדידות ובדיקה בזמן אמת אם המדידות שונות מהתבנית
אם כן אז האלגוריתם מבצע:
 - א. בדיקה אם השינוי הוא חריג בהרבה מהתבנית, אם לא, ממשיך בהרצה כי לא נתפס שינוי.
 - ב. אם כן, משדר את הנתונים למאסטר (אם הוא המאסטר אז כמובן שלא משדר הלאה)

3. ממשיך בריצה מחדש ומעדכן תבנית כל כמה זמן.

The Protocol Algorithm

```
if initalSample is not define /* initial pattern to compare with real time data */  
    initalSample = start measurements()  
  
loop(forever)  
{  
    If you are Master  
    {  
        Listen for data from slaves /* always listen to slaves */  
        Transmit changes catches from Slaves /* transmit data to control center */  
    } /* transmit data will be via external network */  
    if collect real-time data is different from initalSample  
    {  
        try check if data changed in expected way  
        {  
            Every period update InitialSample  
  
            transmit OK message /* ensure system correctness */  
        }  
        catch /*change in data is more different and require further handling*/  
        {  
            transmit data to Master/* The next chain in the communication*/  
        }  
    }  
    else  
    {  
        transmit OK message /* ensure system correctness */  
    }  
} /* End of loop forever */
```

*הערה: ישנו תיעוד במסומן בצבע ירוק לגבי כל שלב בקוד האלגוריתם.

תהליכים ונתוני המערכת

במצב ההתחלתי של המערכת יבוצעו מספר פעולות:

- הפרוטוקול יבדוק האם קיימת תבנית התחלתית ראשונית בקבצי המערכת, אם לא קיימת, הפרוטוקול יבצע מספר דגימות והן יעוצבו לדמות תבנית התחלתית.
- הסבר נוסף ומפורט על אופי בחירת התבנית יינתן בהמשך הדוח בתכנית בדיקות.
- הפרוטוקול יבצע סריקה ומיפוי של כל הרכיבים ברשת המקומית, לאחר מציאת כתובת IP פעילה, הפרוטוקול יוודא האם אותה כתובת היא רכיב IOT ואם כן הוא יוסיף אותה לרשימת רכיבים פעילים שקיימת אצלו ובאמצעותה הוא ישתמש יותר מאוחר בפרוטוקול.
- בדיקת הפרוטוקול על כתובת IP בוצעה באמצעות פונקציות ושיטות Sub-Process, בנוסף בשימוש בפקודות SSH ו-PING.
- PING – חבילת נתונים הנשלחת ממקור מסוים ליעד מסוים, מטרה העיקרית לה היא משמשת היא בחינת תקינות התקשורת בין נקודת המקור לנקודת היעד. בפקודה זו בדקנו איזה מארח פעיל.
- SSH – (Secure Shell) פרוטוקול של התחברות למכשיר מרוחק והרצת תהליך בתוך המחשב המרוחק, כמובן שהתהליך מאובטח ויש צורך בהליך התחברות והזנת סיסמא שידועה רק ליודעי דבר. מה שרצינו מהמחשב המרוחק הוא לשלוף את הנתונים שלו ובכך לזהות שהוא רכיב IOT. (ולא לשם הדוגמה: מחשב נייד), וזה מה שאפשר לנו עם הפרוטוקול הזה.
- למזלנו את הפיתוח ביצענו בשפת פיתון ולכן יכולנו לבצע את כל הדברים האלו בהצלחה רבה לאחר מחקר ופיתוח.

לאחר המצב ההתחלתי האלגוריתם יבצע מדידות כדי להשוות עם התבנית ההתחלתית על מנת למצוא חריגות במדידות החדשות, כמו כן, האלגוריתם יגדיר מי הרכיב מאסטר.

בשלב הבא ותוך כדי שהפרוטוקול מבצע מדידות מהחיישן, יוגדרו שני מצבים אפשריים:

מצב עבד (Slave) – מצב זה אומר שכאשר התגלתה חריגה בדגימות, הדגימה לא תשודר ישירות למרכז הבקרה, אלא תשודר את רכיב המאסטר.

מצב שליט (Master) – במצב זה המאסטר מאזין לחבילות המידע שיגיעו משאר הרכיבים, כאשר קיבל את הנתונים במלואם, המאסטר מחזיר תשובה לרכיב השולח שהנתונים התקבלו.

חשוב להדגיש בשני המצבים במידה ויש חיבור לחיישן אז יש איסוף נתונים ודגימות.

כמו כן, לשני המצבים יש אופציות שידור והאזנה כדי להקשיב או לשדר שינויים במצב המערכת.

שינויים אפשריים:

- מאסטר נפל.
- מאסטר חדש נבחר.

תיכון המערכת – תרשים מחלקות

- התרשים הבא מתאר את מחלקות הפרוטוקול כפי שבנינו במהלך הפיתוח:
- מחלקת Main – סורקת אחר רכיבים ומגדירה מי הרכיב השליט.
- מחלקת HC – הגדרת מחלקה לחיישן ה HC-sr04 על מנת לקלוט את הנתונים במלואם ולעבד את הנתונים. (יורשת ממחלקה אבסטרקטית – Sensors).
- מחלקת Listen – מחלקת האזנה לרכיבים על מנת לקבל נתונים והודעות (משתמשת במחלקת Message על מנת לקבל מבנה של הודעה).
- בנוסף ישנם עוד מחלקות עזר שתפקידם לסייע במהלך תקין של האלגוריתם. המחלקות הם: Auxiliary functions, IOT Device, Constant variable, Sample.

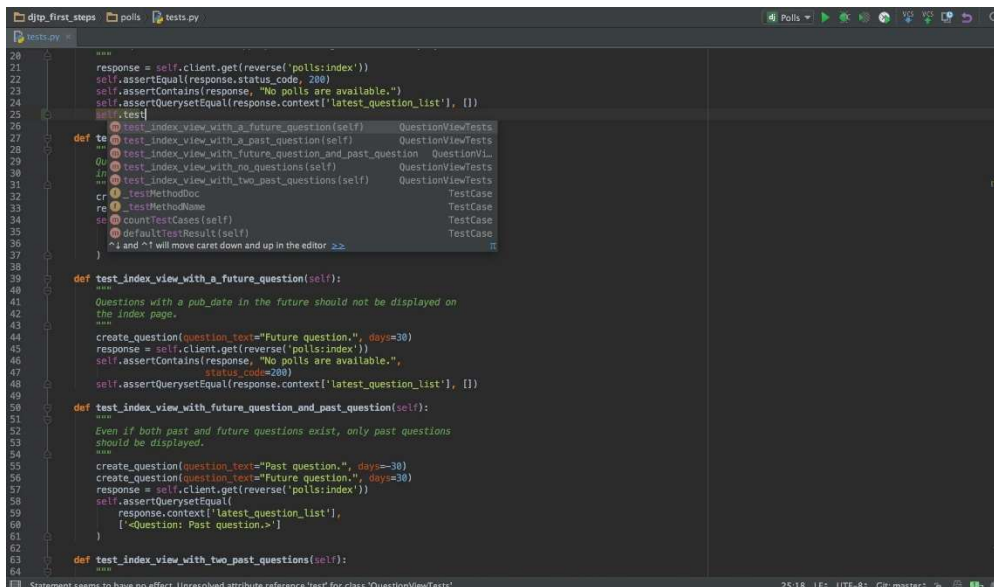


תיאור הכלים המשמשים לפתרון

PyCharm IDE – סביבת פיתוח משולבת נוחה בשפת Python, סביבה זו מאפשרת כתיבה ועריכת קוד מקור, עוזרת בתיקון שגיאות, הידור ו/או מנפה (דיבאגר). כמו כן, הסביבה כוללת כלים לבקרת תצורה וניהול גרסאות וכן כלים המקלים על בניית יישומים בעלי ממשק משתמש גרפי. בסביבה זו כתבנו את כל האלגוריתם שלנו ואנו ממליצים לכל מפתח בשפת פיתון להשתמש בכלי עבודה היעיל הזה.



להלן צילום של התוכנה:



23

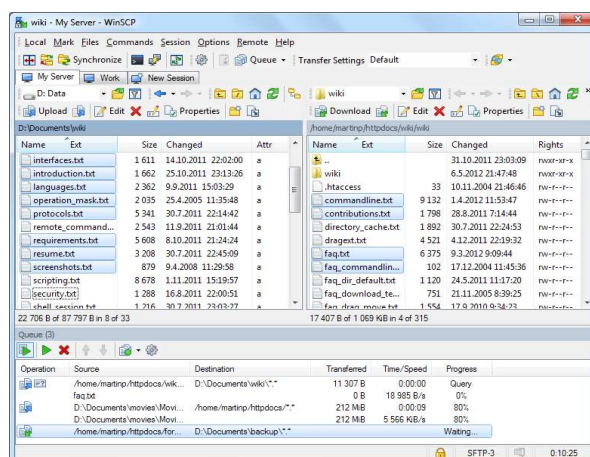
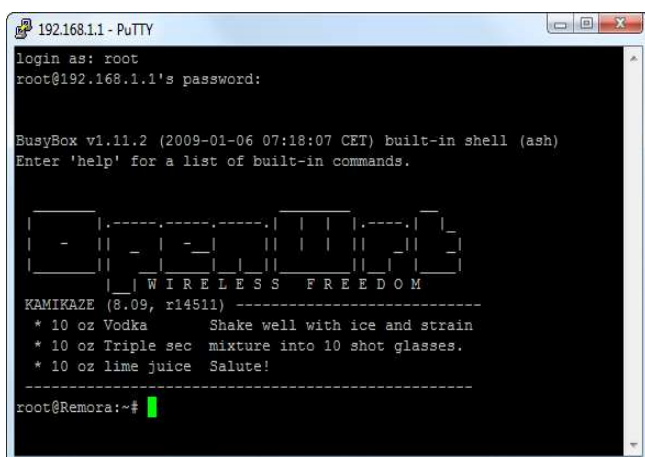
WinSCP – כלי פיתוח המאפשר חיבור מהיר ברשת לרכיב ה IOT, כלי זה מאפשר העברת קבצים מהירה ברשת מהמחשב המקומי אל שטח אחסון היושב בשבב (מחשב המרוחק), כמו כן ניתן לגשת לכל קבצי המערכת בשבב, הכלי מאפשר שינוי גישה לקבצים (כתיבה, קריאה, מנהל)



התוכנה היא קוד פתוח ללא תשלום ומאפשרת אפשרויות חיבור S3, WebDAV, SFTP, FTP ו-SCP עבור Windows. הפונקציה העיקרית שלו היא העברת קבצים בין המחשב המקומי לבין המחשב המרוחק. מעבר לכך, WinSCP מציע יכולת סקריפטנג ומנהל קבצים פונקציונאלי. בתחילת השנה היינו מוגבלים מאד בעבודה על הרכיבים כי לא הכרנו את התוכנה הזו, היינו צריכים להעלות קוד דרך תוכנת Putty שעבדה בצורה לא יעילה וחכמה, כמו כן היינו צריכים כל פעם להתחבר למנהלי ההפעלה של השבבים כדי לגלות את כתובות ה IP שלהם ואז לגשת

ישירות אליהם דרך putty, רק לאחר שגילינו את התוכנה הזו "החיים שלנו נהיו הרבה יותר קלים" והצלחנו להתחבר בצורה מהירה אל מנהל קבצים ולהריץ את התוכניות במערכת.

ניתן לראות כאן צילום של התוכנה כדי לקפל פרספקטיבה על אופן העבודה שלנו אתה. התוכנה הזו משמשת כמנהל קבצים מותאם לרכיבי IOT, בצדו הימני ניתן לראות את הקבצים המופיעים בתוך הרכיב, ניתן לגלול ולעבור לתיקיות אחרות. בצדו השמאלי מופיעים הקבצים בתיקייה מסוימת במחשב המקומי. ניתן להעביר ישירות קבצים מצד אחד לשני. כמו כן, ניתן להפעיל ממשק שורת פקודה בשם Putty אשר יאפשר לנו להריץ את הפרוטוקול בשבבים ולראות את פעולותיו על המסך. ניתן לראות פה צילום מסך של ממשק ה Putty בעת התחברות לכתובת IP שמצינת בראש התמונה.



הפרויקט נוהל לחלוטין על ידי מערכת ה GitHub אשר סיפקה לנו :

- מאגר הקוד.
- מערכת משימות ומטלות.
- מצב הפרויקט.
- מסמכים ומידע נוסף שרלוונטי לפרויקט.

כמו כן אנו ניהלנו יומן אירועים של מהלך הפרויקט מתחילתו ועד סופו באתר [Trello](https://trello.com/). בעמוד הראשי של היומן ניתן לראות את ההתקדמות שלנו במשך השנה בחלוקה לדברים שונים שביצענו כגון: פגישות עם המנחה, מחקר ולמידה, פיתוח ועבודה על הפרוטוקול, ועבודה על מסמכים ודוחות. ניתן לראות את העבודה שביצענו גם בחלוקת תאריכים במשך השנה לפי ימים וחודשים – צד ימין למעלה.

תיאור הפרוטוקול שמומש

התוצר הסופי שלנו בסופו של דבר הוא פרוטוקול כללי להעברת נתונים מינימלית. הפרוטוקול יקלוט נתונים מהחיישנים, יחשב את המדידות ויבדוק האם הם חריגים מנקודת הסף של התבנית ההתחלתית שהגדרנו ותעודכן במשך הריצה של האלגוריתם. כמו כן, הפרוטוקול ישדר את הנתונים אל רכיב המאסטר. למרות אופיו המחקרי של הפרויקט, מומשה פונקציונאליות רבה והרבה שיטות שיוועות להתמודד עם תקלות צפויות כמו נפילת מאסטר ובחירת מאסטר חדש. במהלך הפיתוח הבנו שהשבבים עליהם אנו עובדים מוגבלים מאד ביכולת אחסון ועיבוד ולכן עבדנו קשה על מנת לשפר יעילות של האלגוריתם ובעצם "להרזות" אותו. בסופו של דבר, הראנו דרך לצמצם את תעבורת המידע ברכיבי ה-IOT, על מנת לחסוך בחומרה ומשאבים פיננסיים שלטענתנו מיותרים.

25

נפרט את הפונקציונאליות שהענקנו לאלגוריתם:

- סריקה ומיפוי של מארחים (Hosts) ברשת המקומית – הסריקה מוצאת את כתובות האי-פי ושם של רכיבי ה-IOT.
 - קליטה ומדידת נתונים מחיישנים מחוברים המחוברים אליהם פיזית.
 - הגדרת רכיב כשליט או עבד.
 - פונקציות השוואת מדדים ותבניות.
 - יכולות שידור והאזנה לכל רכיב
 - יכולת גילוי לאפשרות שרכיב המאסטר נפל (התנתק ברשת או עמוס/לא זמין לקבלת נתונים).
 - יכולת החלפת מאסטר בעת הגעת רכיב חדש חזק יותר.
- חשוב לציין שהפרוטוקול ירוץ לנצח, היות וכל הזמן מתקבלים נתונים חדשים מהחיישנים.

הערה : בתכנית הבדיקות תוכלו להיחשף יותר למרכיבי הקוד שלנו .

תכנית בדיקות

בפרק זה נציג Test Plan שלם לפרוטוקול שלנו. מטרת הבדיקות היא לוודא, עד כמה שאפשר, שהאלגוריתם עושה את מה שהוא צריך לעשות ולא עושה מה שאינו צריך לעשות. אופי הבדיקות שלנו יהיה בדיקות אוטומטיות על ריצת האלגוריתם, ונציג כעת תכנית בדיקה מסודרת, שהצלחה בכל שלביה תאשר במידה רבה את תפקודו התקין של הפרוטוקול עבור העבודה שנעשתה בו עד כה. רוב הבדיקות יהיו בדיקות של פונקציונליות האלגוריתם לאימות פעילות המערכת בהתבסס על הדרישות ממנה. בפרק זה נציע תכנית בדיקה עבור המרכיבים הנוכחיים שמומשו עד לשלב ההגשה, נחלק את התכנית שלנו לשני שלבים: שלב בדיקות תקינות הפרוטוקול בשגרה ושלב בדיקות תקינות הפרוטוקול כאשר קרתה תקלה. נרצה להראות שגם כאשר הפרוטוקול שלנו מתמודד עם תקלה, הוא מסוגל לתפעל אותה, ובכך נוכל אולי לקבוע שהוא Fault-Tolerance.

עבדנו עם 5 רכיבים, כל רכיב מראש הוגדר עם שם שונה ממין מסדר עולה: D1, D2, D3, D4, D5.

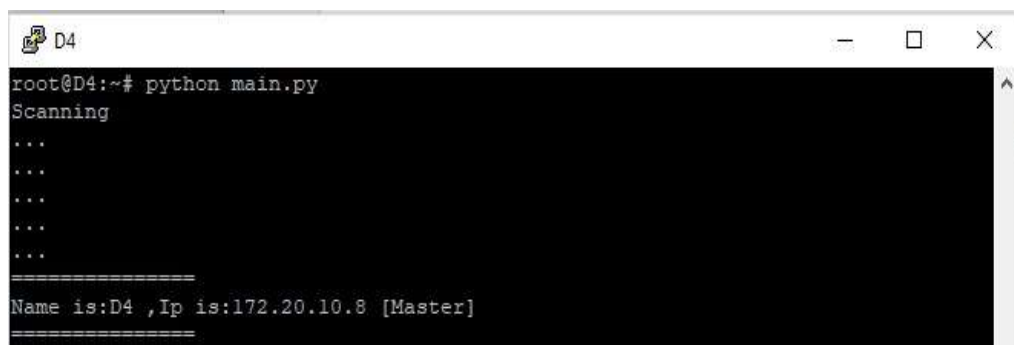
בדיקות הפרוטוקול במצב שגרה

בדיקה 1 – סריקת רשת ומיפוי רכיבים:

על מנת שהאלגוריתם יתחיל בריצה שלו ויבין מי הרכיבים ברשת שמולם הוא ירצה לעבוד, האלגוריתם צריך לסרוק את הרשת ולמפות את הרכיבים הפעילים. (במקרה הספציפי, מי הוא הרכיב השולט שאליו ישלח את הנתונים במקרה וימצא חריגה במדדים). האלגוריתם משתמש כדי לסרוק את הרשת ולהגיע לכל רכיב בספריית Sub-process, ספרייה זו מאפשרת לבצע תת-תהליכים ולהריץ פקודות Shell בתוך הסקריפט שלנו. נאלצנו לבדוק כל IP ברשת המקומית שלנו וזאת כי כשהרצנו את האלגוריתם שלנו בכמה נתבים שונים, כל רכיב קיבל כתובת IP שונה ורנדומלית ולכן האלגוריתם סורק 255 כתובות IP. לכל כתובת IP השתמשנו בפקודת Ping כפי שתיארנו מקודם, ולבסוף כדי לקבל שהכתובת שהוא עובד מולה היא בעצם מחשב מארח בדמות רכיב IOT, האלגוריתם משתמש בפרוטוקול SSH שהוא פרוטוקול התחברות למחשב מרוחק, כיוון שהגדרנו מבעוד מועד את כל הרכיבים שיהיו עם אותו Login ו-Passwd, האלגוריתם הצליח להתחבר אל הרכיב המרוחק ולקבל ממנו את שמו. כדי לבדוק זאת ביצענו 2 בדיקות עיקריות והרצנו אותן מספר פעמים מכל רכיב שהיה ברשותנו (ברשותנו עמדו 5 רכיבי IOT).

בדיקה 1.1 : סריקת רשת ומיפוי רכיבים – רכיב אחד מחובר ברשת:

ביצענו בדיקה מכל רכיב שהיה ברשותנו, היות והרצנו את זה על רכיב אחד בלבד כשהוא רק מחובר ברשת קיבלנו בסופו של הסריקה – 0 תוצאות לגבי רכיבים אחרים, ולכן הוא גם הוגדר כברירת מחדל כרכיב שליט (Master).
הרצנו את זה בכל רכיב שהיה ברשותנו, קיבלנו תוצאה נחרצת בכל פעם שהרצנו את הבדיקה.
כדי להמחיש פה נראה לכם פה צילום של פעולת הסריקה והתוצאה שהתקבלה על ידי אחד מהרכיבים שלנו (במקרה זה: D4).



```

root@D4:~# python main.py
Scanning
...
...
...
...
=====
Name is:D4 ,Ip is:172.20.10.8 [Master]
=====

```

תוצאה: הבדיקה התבצעה בהצלחה !

בדיקה 1.2 : סריקת רשת ומיפוי רכיבים – 5 רכיבים ברשת:

ביצענו בדיקה מכל רכיב שהיה ברשותנו, רק שהפעם חיברנו והפעלנו את כל הרכיבים שקיימים ברשותנו, חשוב להדגיש שהאלגוריתם מוצא את הרכיבים אפילו מבלי שהם יריצו את האלגוריתם בעצמם, מספיק שיהיו מחוברים לרשת והוא מוצא אותם.
גם פה הרצנו מספר פעמים כדי לקבל תוצאות נחרצות ואכן קיבלנו אותם.
הסיבות לכך הם אמינות הפרוטוקולים והפקודות שהשתמשנו בהם, פקודת Ping משתמשת בפרוטוקול ICMP שמאד אמין, כמו כן, פרוטוקול SSH מבצע התחברות מרחוק למחשבים מארחים רק לאחר שעבר את כל מבדקי האבטחה ולכן גם פה מובטחת לנו אמינות.
כדי להמחיש פה נראה לכם פה צילום של פעולת הסריקה והתוצאה שהתקבלה על ידי אחד מהרכיבים שלנו (במקרה זה: D1).



```

root@D1:~# python main.py
Scanning
...
...
...
...
=====
Name is:D5 ,Ip is:10.0.0.1 [Master]
Name is:D1 ,Ip is:10.0.0.29
Name is:D2 ,Ip is:10.0.0.30
Name is:D3 ,Ip is:10.0.0.31
Name is:D4 ,Ip is:10.0.0.32
=====

```

תוצאה: הבדיקה התבצעה בהצלחה !

בדיקה 2 – בדיקת תבנית קיימת/הגדרת תבנית:

בבדיקה הבאה רצינו לבדוק האם האלגוריתם טוען את התבנית ההתחלתית עליה הוא יכול להסתמך במשך היום, האלגוריתם שלנו בשלב הראשוני בודק אם קיימת תבנית על ידי חיפוש קובץ דגימות התחלתיות בתיקייה בה האלגוריתם מותקן. אם מצא קובץ שכזה הוא טוען ומסתמך עליו כתבנית התחלתית, אם לא מצא, הוא יוצר תבנית התחלתית.

השיטה שמימשנו למימוש התבנית שלנו הוא: **בכללי ניתן להגדיר כל דגימה איך תילקח**, אנחנו בחרנו לעשות 24 מדידות כדי לדמות דגימה כל שעה ביום, כמובן שלא יכולנו למדוד כל שעה אז בחרנו שהמדידות יתבצעו כל כמה שניות (על מנת לחסוך זמן ולראות את התוצאות בצורה די מהירה ומידית).

השיטה מבצעת מדידה ועושה השוואה למול המדידה הקודמת באמצעות חישוב דלתא: $(x_1 - x_0)$, אם התוצאה לא גדולה מהסטייה (גם פה אנו בחרנו, כמובן שניתן לשנות) המדידה החדשה נכנסת למערך של ההשוואות הקודמות, לבסוף כאשר יש מדידה שסוטה מההכריזה שקבענו, כל שאר הדגימות שהיו במערך הקודם מוציאות לנו פלט סופי של ממוצע.

לדוגמה:

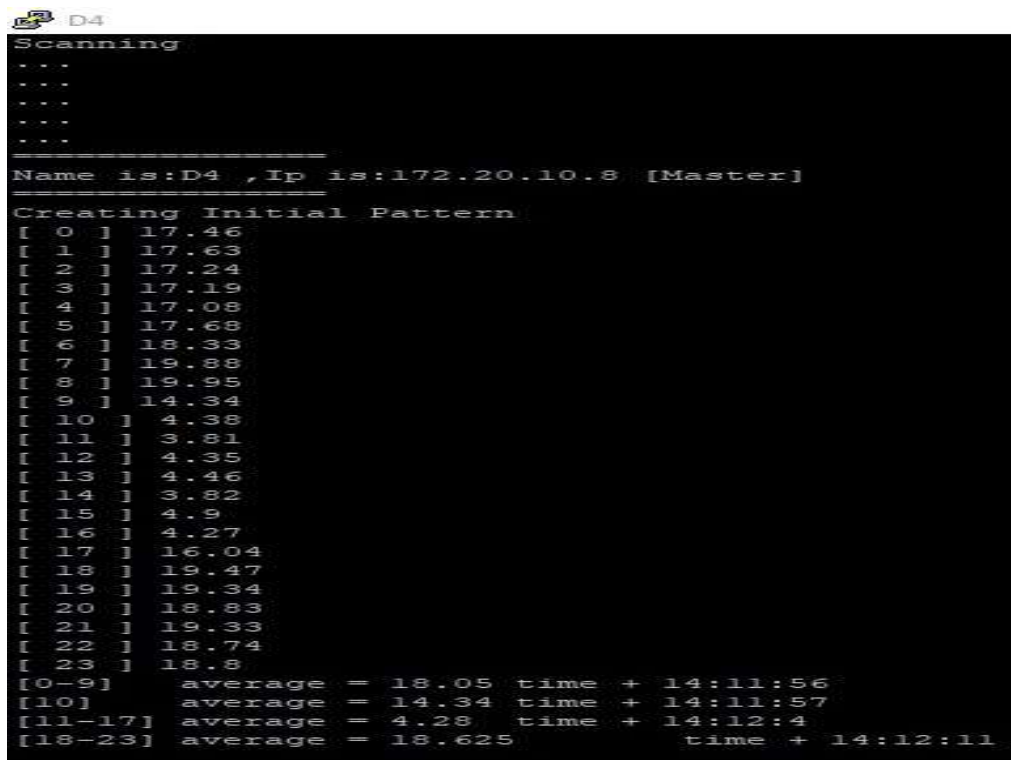
Measurements: {[index: 1-13, avg:8.6 time:13:56-02:56][index: 14, avg:25.3,time:03:56][index: 15-24 , avg:64.32 , time: 04:56-12:56]}

בדקנו את 2 המקרים :

בדיקה 2.1 - תבנית לא קיימת, יצירה חדשה

מחקנו את קובץ התבנית שלנו והרצנו כדי לבדוק האם האלגוריתם תופס את זה שאין תבנית ויוצר בעצמו חדשה בזמן אמת.

28



```

D4
Scanning
...
...
...
Name is:D4 ,Ip is:172.20.10.8 [Master]
Creating Initial Pattern
[ 0 ] 17.46
[ 1 ] 17.63
[ 2 ] 17.24
[ 3 ] 17.19
[ 4 ] 17.08
[ 5 ] 17.63
[ 6 ] 18.33
[ 7 ] 19.88
[ 8 ] 19.95
[ 9 ] 14.34
[10] 4.38
[11] 3.81
[12] 4.35
[13] 4.46
[14] 3.82
[15] 4.9
[16] 4.27
[17] 16.04
[18] 19.47
[19] 19.34
[20] 18.83
[21] 19.33
[22] 18.74
[23] 18.8
[0-9] average = 18.05 time + 14:11:56
[10] average = 14.34 time + 14:11:57
[11-17] average = 4.28 time + 14:12:4
[18-23] average = 18.625 time + 14:12:11
  
```

תוצאה: הבדיקה התבצעה בהצלחה !

בדיקה 2.2 - תבנית קיימת

כדי לבדוק את זה כמו שצריך, השארנו את הדגימה שביצענו מקודם והרצנו שוב.

```

root@D4:~# python main.py
Scanning

...
...
...
...
...

=====
Name is:D4 ,Ip is:172.20.10.8 [Master]
=====

Initial Pattern exist
[0-9]   average = 18.05 time + 14:11:56
[10]    average = 14.34 time + 14:11:57
[11-17] average = 4.28  time + 14:12:4
[18-23] average = 18.625 time + 14:12:11

```

תוצאה: הבדיקה התבצעה בהצלחה !

29

בדיקה 3 – תפיסת חריגה ושליחה לרכיב השליט (Master):

בבדיקה זו האלגוריתם רץ כרגיל על המדידות שמתקבלות אצלו בכל רגע, הוא מבצע השוואה למול התבנית כדי למצוא חריגה במדדים, אם הוא מוצא הוא משדר את הנתונים אל המאסטר. אנו נבדוק את תקינות האלגוריתם על גבי 3 רכיבים שיעבדו במקביל.

לשם כך בחרנו את רכיבים

D3,D1,D4, מי שנבחר בתחילת

הריצה כמאסטר הוא רכיב D1,

נצפה שילכו אליו המדידות החריגות

ובמידות.

```

root@D4:~# python main.py
Name is:D1 ,Ip is:10.0.0.29 [Master]
Name is:D3 ,Ip is:10.0.0.31
Name is:D4 ,Ip is:10.0.0.32
=====
Creating Initial Pattern
[ 0 ] 13.91
[ 1 ] 13.05
[ 2 ] 13.61
[ 3 ] 13.31
[ 4 ] 13.85
[ 5 ] 13.58
[ 6 ] 13.69
[ 7 ] 13.77
[ 8 ] 13.78
[ 9 ] 13.81
[10] 14.37
[11] 13.43
[12] 13.72
[13] 13.9
[14] 13.82
[15] 13.26
[16] 13.82
[17] 13.9
[18] 13.74
[19] 13.77
[20] 13.7
[21] 13.79
[22] 13.24
[23] 13.26
[0-23] average = 13.6634762409 time + 15:7:14

Sample: 13.31
O.K
Sample: 13.91
O.K
Sample: 4.68
Trying to send 4089 Message
Message arrived 4089
Send the exception measurement
Sample: 10.6
O.K

```

כפי שרואים ב D4, נתפס שינוי והוא נשלח למאסטר. למטה רואים בהגדלה.

```

Sample: 13.91
O.K
Sample: 4.68
Trying to send 4089 Message
Message arrived 4089
Send the exception measurement
Sample: 10.6
O.K

```



באותה מידה נמצא גם חריגה ב D3, עכשיו נראה אם המאסטר שלנו קיבל את 2 החריגות מהרכיבים המודדים.
מימין רואים בהגדלה.

D3

```
Sample: 11.99
O.K
Sample: 3.57
Trying to send 7189 Message
Message arrived 7189
Send the exception measurement
Sample: 11.34
O.K
```

```

root@kali:~# python main.py
Scanning
...
...
...
...
...
=====
Name is: D1 , Ip is: 10.0.0.29 (Master)
Name is: D3 , Ip is: 10.0.0.31
Name is: D4 , Ip is: 10.0.0.32
=====
Creating Initial Pattern
No sensor is connected
No sensor is connected
No sensor is connected
No sensor is connected
No sensor is connected
No sensor is connected
NoSample 10.0.0.31
No sensor is connected
No sensor is connected
No sensor is connected
No sensor is connected
No sensor is connected
No sensor is connected
No sensor is connected
No sensor is connected
No sensor is connected
NoSample 10.0.0.32
No sensor is connected
Got an exception from D3: the measurement is:4.68 Routine measure
ment: 15.643782469
No sensor is connected
No sensor is connected
Got an exception from D3: the measurement is:2.76 Routine measure
ment: 10.357933043
No sensor is connected

```

רכיב המאסטר (D1) שמקבל משני מהרכיבים הודעות על כך שנמצאו חריגה מהנתונים. למטה רואים בהגדלה.

30

```
No sensor is connected
  Got an exception from D4: the measurement is:4.68 Routine measure
ment: 13.6634782609
No sensor is connected
No sensor is connected
  Got an exception from D3: the measurement is:2.76 Routine measure
ment: 10.5573913043
No sensor is connected
```

תוצאה: הבדיקה התבצעה בהצלחה!

בדיקות הפרוטוקול במצב תקלה

בדיקה 4 – עליית רכיב שליט (Master) חדש

כדי שהפרוטוקול יעבוד כמו שמצופה, אנו צריכים להיערך לאפשרות שיעלה רכיב שעולה על הדרישות של רכיב שליט והוא בעצם בעמדה עליונה יותר מהשליט הנוכחי, אז על פי האלגוריתם הרכיב החדש הופך להיות השליט, הדבר הטריקי ושאתגר אותנו הוא, איך מעדכנים את שאר הרכיבים על השליט החדש? הרי מבחינתם יש להם שליט חדש?

הפתרון שמימשנו את זה הוא שברגע שהרכיב החדש עולה לרשת הוא שולח הודעה לכל הרכיבים הפעילים ברשת שהוא השליט החדש, הם כמובן בודקים שהוא עונה על הדרישות ובהתאם מגדירים אצלם שהוא השליט החדש ובכך ממשיכים את ריצת האלגוריתם אתו כהשליט החדש.

הרכיב הקודם שהיה שליט מורד בדרגה והופך להיות רכיב עבד (Slave). כל זה מתאפשר כי האלגוריתם שרץ בכל אחד מהרכיבים מסתמך על סט של חוקים שקובעים איך יבחר השליט.

- בשלב הראשון חיברנו 2 רכיבים לרשת – D3, D4, ולכן על פי האלגוריתם מופיע לנו רשימה מצומצמת של 2 רכיבים פעילים, D3 נבחר להיות המאסטר על פי זה שהכתובת שלו הכי נמוכה.

D4	D3
<pre>Name is:D3 ,Ip is:10.0.0.31 [Master] Name is:D4 ,Ip is:10.0.0.32</pre>	<pre>Name is:D3 ,Ip is:10.0.0.31 [Master] Name is:D4 ,Ip is:10.0.0.32</pre>

31

- בשלב הבא חיברנו עוד רכיב לרשת – D1, האלגוריתם שלו מתחיל בסריקה והוא מוצא 3 רכיבים פעילים ברשת (כולל הוא עצמו), הוא מגדיר את עצמו כמאסטר.
- בשלב הבא הוא שולח הודעות לשאר הרכיבים ברשת (להם מימשנו כמובן פונקציות האזנה שרצות באופן תמידי). ההודעה שנשלחת: אני המאסטר החדש.

```
root@D1:~# python main.py
Scanning
...
...
...
...
...
=====
Name is:D1 ,Ip is:10.0.0.29 [Master]
Name is:D3 ,Ip is:10.0.0.31
Name is:D4 ,Ip is:10.0.0.32
```

סריקת רכיב D1 ומיפוי הרכיבים, מציאתם והגדרתו כמאסטר

- הרכיבים מקבלים את ההודעה ובודקים מקומית אם באמת הרכיב החדש הוא המאסטר החדש, לאחר שהסיקו שהוא החדש, הם מגדירים אצלם שרכיב D1 הוא המאסטר החדש.

D4	D3
<pre>Update master, new master: D1 Initial Pattern exist [0-23] average = 10.5573913043 time + 15:5:33</pre>	<pre>Update master, new master: D1 Initial Pattern exist [0-23] average = 10.5573913043 time + 15:5:33</pre>

תוצאה: הבדיקה התבצעה בהצלחה!

בדיקה 5 – נפילת רכיב שולט (Master)

רצינו להתמודד עם האפשרות שאלגוריתם יקבל שגיאה ורכיב אחד התנתק מהרשת (למשל: נפילת מתח חשמלי).

האלגוריתם בשאר הרכיבים רץ כרגיל, עד למקרה ובו נתפסה חריגה באחד הרכיבים והוא מנסה לשלוח את המידע אל הרכיב השליט.

הרכיב מנסה לשלוח N מסוים של פעמים ($N=3$ אצלנו, ניתן להגדיר את זה בקלות למספר אחר) לאחר שהוא לא מצליח לשלוח בכל הפעמים, האלגוריתם מניח שרכיב השליט נפל ולא פעיל יותר. בשלב הבא הרכיב שזיהה את הנפילה מודיע לכולם באמצעות הודעות מערכת (שהגדרנו שהיא מסמלת נפילת מאסטר), כל רכיב אחר שנמצא ברשת מקבל את ההודעה, מפענח אותה כהודעת מערכת מסוג זו. לאחר קבלת ושליחות ההודעות, כל רכיב סורק מחדש את הרשת וממפה את הרכיבים הפעילים, לאחר סיום הסריקה כל רכיב מגדיר רכיב שליט חדש והפרוטוקול ממשיך בריצתו כרגיל.

בדיקת נפילת הרכיב השולט –

כדי לבדוק את זה כמו שצריך בדקנו מספר פעמים, על 2 סוגים של תרחישים.

1. בשלב הראשוני בדקנו רק עם שני רכיבים - בהתחלה הוגדר שליט כמו שצריך ולאחר מכן ניתקנו את הרכיב השליט מהחשמל ורצינו לראות איך יגיב הרכיב השני, בדקנו את זה עם מספר רכיבים. **תוצאה:** הבדיקה התבצעה בהצלחה, הרכיב השני הצליח לתפוס את נפילת השליט כאשר ניסה לשלוח לו נתונים, סרק מחדש והגדיר את עצמו כמאסטר.

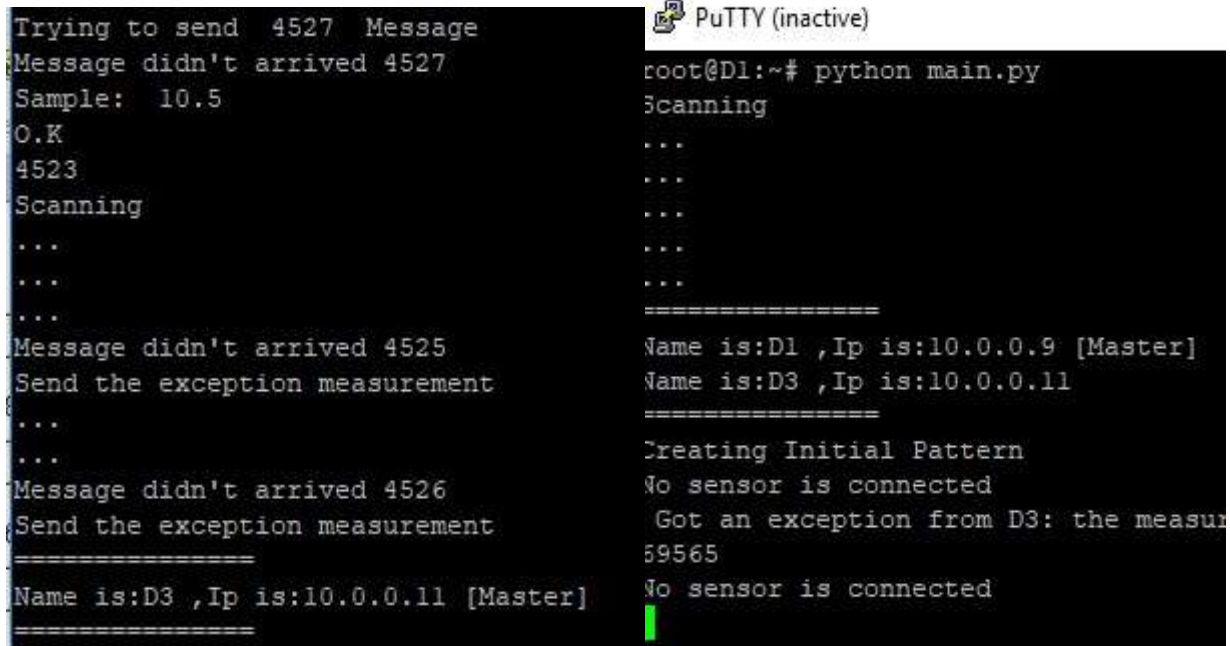
D3

זיהה את הנפילה סרק ומצא רק את עצמו ולכן הגדיר את עצמו כמאסטר. (כברירת מחדל)

D1

הוגדר תחילה כמאסטר ולאחר מכן ניתקנו אותו מהחשמל

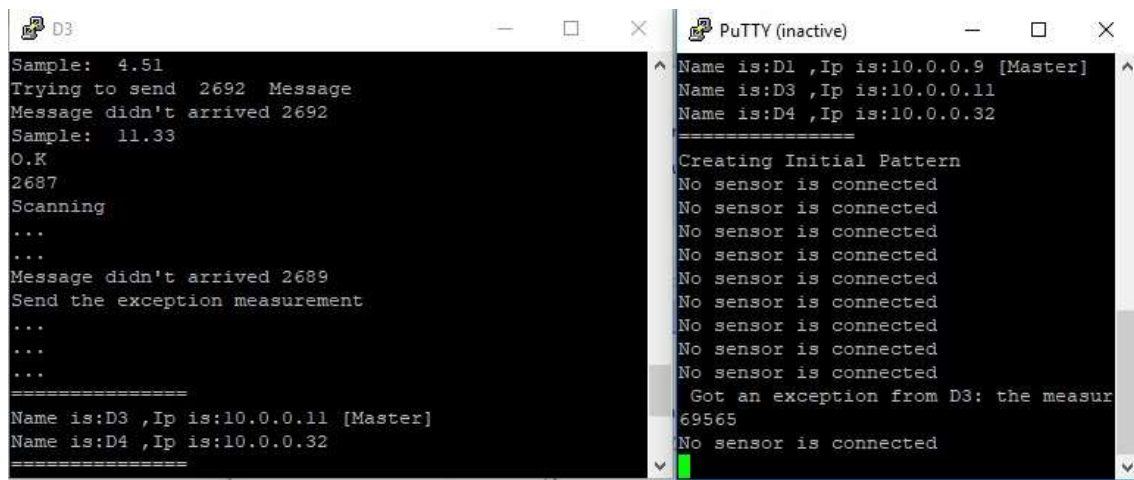
32



```
Trying to send 4527 Message
Message didn't arrived 4527
Sample: 10.5
O.K
4523
Scanning
...
...
...
Message didn't arrived 4525
Send the exception measurement
...
...
Message didn't arrived 4526
Send the exception measurement
=====
Name is:D3 ,Ip is:10.0.0.11 [Master]
=====
```

```
PuTTY (inactive)
root@D1:~# python main.py
Scanning
...
...
...
=====
Name is:D1 ,Ip is:10.0.0.9 [Master]
Name is:D3 ,Ip is:10.0.0.11
=====
Creating Initial Pattern
No sensor is connected
Got an exception from D3: the measur
59565
No sensor is connected
```


2. בשלב המשני בדקנו עם מספר רכיבים - רצינו לבדוק עם מספר רכיבים על מנת להראות את האפשרות של השליחה לכל הרכיבים הודעה וזאת על מנת שהם ידעו על הנפילה ויסקרו גם. גם פה בדקנו כמה פעמים. **תוצאה:** הבדיקה התבצעה בהצלחה, אחד הרכיבים הצליח לתפוס את נפילת השליט כאשר ניסה לשלוח לו נתונים, שלח הודעות לשאר הרכיבים והם סרקו מחדש ובחרו את D3 כמאסטר לאחר שבמקודם היה זה D1 השליט. מצורף תמונה הממחישה את הנפילה והבחירה המחודשת.



תוצאה: הבדיקה התבצעה בהצלחה !

תוצאות תכנית הבדיקות:

ערכנו 5 בדיקות משמעותיות שהדגישו את הפונקציונאליות העיקרית של הפרוטוקול שלנו, יש לציין שהיה לנו כמה באגים קטנים במערכת אך בגלל שהרצנו כמה פעמים לא מעטות ידענו בסופו של דבר להגיע לשורש הבעיה ולתקנה.

- סריקת רשת ומיפוי רכיבים – **התבצעה בהצלחה !**
- מציאת תבנית קיימת, הגדרת תבנית – **התבצעה בהצלחה !**
- מציאת חריגה במדדים ושליחה למאסטר – **התבצעה בהצלחה !**
- הגדרת רכיב שליט חדש חזק יותר – **התבצעה בהצלחה !**
- נפילת רכיב שליט – **התבצעה בהצלחה !**

אנחנו מגדירים את תכנית הבדיקות שלנו כהצלחה – 100% !

סקירת עבודות הקשורות והשוואה לספרות

המחקר והפיתוח שלנו התבסס על ההנחה שלא נמצא פרוטוקול שבאמת מבצע עיבוד וסינון מידע שיושב על רכיבי IOT, אך לאחר מחקר רב מצאנו מספר רב של פרוטוקולי תקשורת והעברת נתונים הקשורים לרכיבים אלו.

פרוטוקולים אלו פותחו על ידי ארגונים גדולים כגון חברת IBM ובוצעו תוך מתן דגש מיוחד לפונקציונאליות ולשינויים בקווי התקשורת בעת מעבר חבילות המידע בצורת התקשורת. בחירתנו הייתה לא להמציא את הגלגל מחדש, אלא להסתמך על פרוטוקולים אלו בכל הנוגע לשידור נתונים ומבנה ארכיטקטורה מורכב המכסה את כל הנדבכים הנדרשים למימוש הפרוטוקול.

להלן 3 מהפרוטוקולים הנפוצים בעולם ויש להם שימוש רחב ברכיבי IOT ומערכות אחרות.

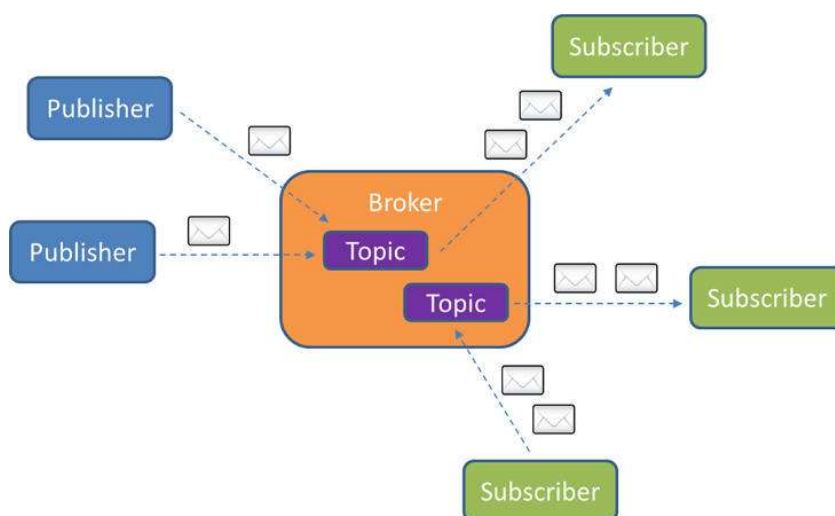
1. **MQTT** - פרוטוקול חיבוריות ממכונה למכונה (M2M) כמו כן הוא פועל גם ברכיבי IOT, פרוטוקול זה קל מבחינת זיכרון ויכולת עיבוד כך שאפשר להפעיל אותו על רכיבים וחיישנים קטנים, בנוסף הוא אידיאלי לשימוש ברשת חיישנים בחיבור מרחוק ע"י לווין.

הפרוטוקול פותח ב-1999 על ידי אנדי סטנפורד-קלארק וארלן ניפר.

ב-2013 חברת IBM הגישה ל OASIS (ארגון הקובע סטנדרטים בתחום התוכנה) את MQTT v3.1 ומאז היא הפכה לסטנדרט עולמי, לפרוטוקול הזה מגוון שימושים בתעשייה וביניהם פייסבוק מסנג'ר אשר משתמש בפרוטוקול זה.

המאפיין העיקרי שלו הוא ללא איבוד נתונים ולכן הוא יושב מעל שכבת TCP ובכך מאפשר פשטות ושליחת נתונים אמינה.

דפוס העבודה ב MQTT הוא של publish/subscribe. כל מכשיר יכול לפרסם מידע בנושא מסוים לכל שאר המכשירים שעשו מנוי לנושא זה, ולהפך, הוא יכול לעשות מנוי לנושא מסוים. כל המידע הזה עובר דרך המתווך שמקושר לכל המכשירים, וזוהי כל העבודה שלו.



יתרונות:

- קל מבחינת זיכרון ויכולת עיבוד
- מאפשר הפעלה מרחוק על רשת חיישנים ע"י חיבור לווין
- העברת מידע מהירה מעל TCP

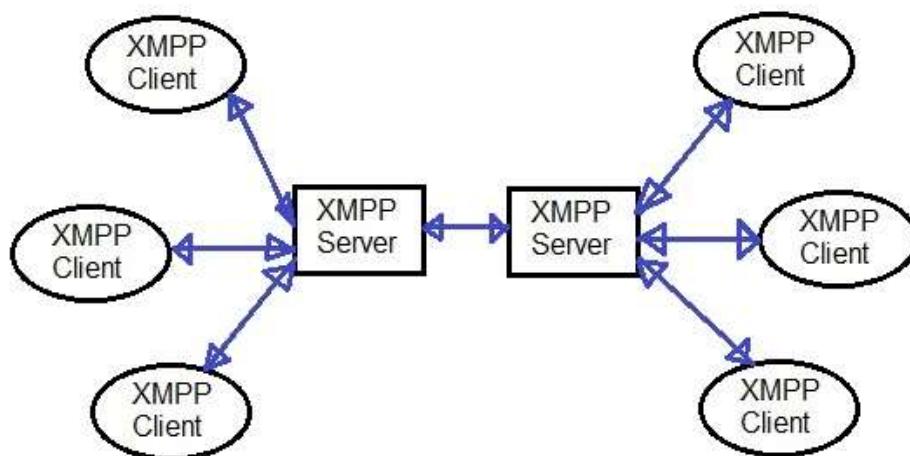
חסרונות:

- לא עובד בזמן אמת, ולרוב זמן אמת נחשב אצלו כמה שניות.
- מסובך להפעלה, קשה להתקנה.

הפרוטוקול הזה דומה במהותו למה שאנו רצינו להשיג מהפרוטוקול שלנו, קל זיכרון, מאפשר העברת מידע בצורה פשוטה וניתן להפעיל אותו מרחוק, לקחנו ממנו את הרעיון למימוש תקשורת הנתונים שלנו בכל הנוגע ל Publisher ו-Subscribers רק שצמצמנו את מרכיב המתווך והחלטנו לשנות קצת את הארכיטקטורה ולבצע תקשורת שתהיה מבוססת על רכיב אחד שמקבל נתונים מכולם ורכיבים רבים ששולחים לו.

2. **XMPP** – פרוטוקול תקשורת והעברת נתונים, מאפשר החלפת מסרים כמעט בזמן אמת, במקור נקרא "Jabber", והוא פותח על ידי קהילת קוד-פתוח ששמה ג'אבר בשנת 1999. הוא תוכנן להיות בר הרחבה, והשתמש בו בין היתר להעברת קבצים, וידאו, משחקים וכמובן IOT ביישומים כגון רשת חכמה. בניגוד לפרוטוקולים אחרים בתחום, XMPP תוכנן כפרוטוקול פתוח בעל הפצה חופשית ולכן כל אחד יכול להשתמש בקוד ולבנות לו XMPP מותאם אישית.

מנגנון הפרוטוקול עובד בצורה יותר יעילה בעיקר בזה שהוא עובד על ידי דחיפת ההודעות בניגוד לפרוטוקולים דומים כדוגמת HTTP שמשתמש בבקשות ותגובות. בהקשר של IOT, הוא מאפשר דרך קלה להגיע להתקן וליצור לו כתובת ברשת, זה מאד שימושי אם הנתונים עוברים מרחוק, בדרך כלל בין 2 נקודות לא קשורות. הוא לא נועד להיות מהיר ולרוב השימוש שלו הוא במשיכות (pull) מידע במקרה הצורך, במונחים אנושיים זמן אמת שלו יהיה כמה שניות, הוא מספק דרך מצוינת לחבר מכשירי חשמל ביתיים (כגון: תרמוסטט) לשרת אינטרנט כך שתוכל לגשת אליו מהפלאפון שלך.



יתרונות:

- פרוטוקול פתוח, בר הרחבה.
- יכולת חיבור מכשירי חשמל IOT – לשרת אינטרנט.
- מחבר בין 2 נקודות בצורה לא ישירה, 2 הנקודות לא חייבות להכיר באופן ישיר אחד את השני.

חסרונות:

- איטי ולא עובד בזמן אמת.
- לא עובד ללא שרת XMPP.

מפרוטוקול זה לקחנו את הרעיון גישה קלה אל הרכיבים הממוקמים ברחבי הרשת, אנו רוצה לגשת אליהם במהירות וכמובן לקבל מהם מידע, מימשנו את זה קצת דומה בכך שאנו דוחפים את הנתונים וההודעות שלנו קדימה אל הרכיב השולט ולא מסתמכים על בקשות ותגובות.

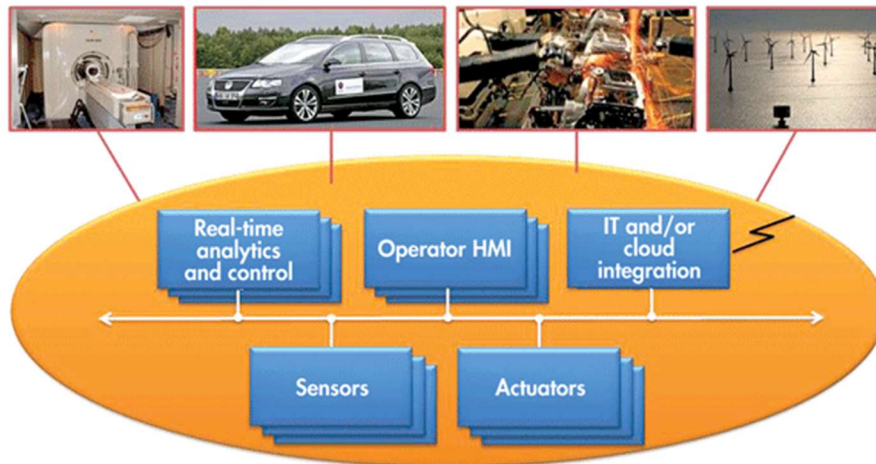
השיפור שלנו הוא שבניגוד לפרוטוקול הזה כאשר נופל שרת XMPP אין המשך עבודה אצלנו האלגוריתם מוצא מאסטר חדש והוא משמש כנקודת יעד חדשה להעברת הנתונים.

3. **DDS** - בניגוד ל MQTT ול- XMPP, שירות הפצת הנתונים (DDS) מתמקד במכשירים שמשמשים באופן ישיר בזיכרון המכשיר, המטרה העיקרית של הפרוטוקול הזה הוא לחבר מכשירים מסוימים לאחרים, בניגוד לאחרים DDS יכול להעביר מיליוני הודעות לשנייה בצורה יעילה ואמינה לכמה מכשירים בו זמנית.

מנגנון הפרוטוקול עובד בצורת מפרסם/מנוי, היות והוא מציע שירות למלא מכשירים עם הרבה הודעות בו זמנית, חיבור TCP לא ישים פה כי הוא מאפשר בדרך כלל חיבור מוגבל בין 2 נקודות ולכן ב DDS יש שימוש ב quality of service (בקרת איכות מפורטת), אשר דוגלת בשידור בו זמנית לכמה מכשירים (multicast), ניתנת להגדרה ותמיכה יתירה.

פרוטוקול זה מציע דרך עוצמתית לסנן איזה מידע ילך ולאן, וה- "לאן" יכול להיות אלפי מכשירים בו זמנית, הוא מאד דומה למעין אוטובוס-מידע, אוטובוס המידע שולט על כניסת המידע ויש לו דימוי של מסד נתונים וזאת כי במסד נתונים יש שליטה על הרשאות הגישה אל המידע (data) המאוחסן, חלק מהמכשירים קטנים ולכן יש גרסאות קלות של DDS שרצות עליהן בצורה מוגבלת.

מערכות משולבות בעלות ביצועים גבוהים משתמשות ב- DDS, זוהי הטכנולוגיה היחידה שמספקת את הגמישות, האמינות והמהירות הדרושים לבניית יישומים מורכבים בזמן אמת. דוגמה לכאלו: מערכות צבאיות, חוות רוח, מערכות אינטגרציה של בתי חולים, מכשירי הדמיה רפואית ועוד.



יתרונות:

- מהיר ומשדר בזמן אמת (מיקרו-שניה).
- יכול לתקשר עם אלפי מכשירים בו זמנית.
- מסוגל לשדר מיליוני הודעות בשנייה.

חסרונות:

- לא קל לשימוש, דורש ידע מעמיק ברשתות אינטרנט ומודל השכבות.
- פחות אמין כי הוא לא משתמש בחיבור TCP.

מה שאהבנו בעיקר מהפרוטוקול הזה הוא היכולת לתקשר עם אלפי מכשירים בו זמנית, האפשרות הזו יכולה לשמש אותנו בשימוש יעיל עם מספר רב של רכיבים שיכולים לתקשר בו זמנית ובמקביל להמשיך באלגוריתם בהקשר של קליטה ועיבוד נתונים.

יכולת השידור המהירה של פרוטוקול זה עזרה לנו להבין שבכדי להגיע לתוצאות הטובות ביותר ובנוסף שאלגוריתם יבצע את הפעולות שלו בסנכרון מושלם עם תהליכים אחרים אנו צריכים שאלגוריתם שלנו ישדר בצורה דומה.

אך למרות היתרונות המשמעותיות של פרוטוקול זה, אנו לא רוצים לסבך את האלגוריתם שלנו ורוצים שהפרוטוקול יהיה ניתן להרחבה ובצורה פשוטה, לכן, עבדנו עם חיבורי TCP שקצת האטו את מהירות השידור (למרות שבעין אנושי לא שמים לב) אך כמובן הוסיפה לנו אמינות ומקצועיות לגבי שידור הנתונים וקליטה מחיישנים.

סיכום\מסקנות

לסיכום הפרויקט עד כה, בנימה קצת אישית, אפשר לומר שקצב ההתקדמות לא היה קבוע במיוחד, והושפע מהצורך ללמוד ארכיטקטורה חדשה שהייתה זרה לנו לחלוטין, אך בעיקר מהעומס שהיו לנו בעבודה ובלימודים- בתקופות אלו ההתקדמות בפרויקט הייתה איטית בהרבה. בכל זאת, בהינתן מסגרת זמן סבירה ומיקוד בעבודה על הפרויקט, אפשר לומר שהתפוקה היא הרבה מעבר למספקת – הצלחנו לעמוד כמעט בכל המשימות שהקצתנו לעצמנו והקצה לנו מנחה הפרויקט.

- חקרנו רבות על רכיב ה IOT ברשותנו, למדנו רבות על המאפיינים שלו, חומרה, דרכי גישה של הנתונים אל המכשיר וממנו, חיישנים שנוכל לחבר אליו ונצליח לקלוט מהם מידע וכמובן איך לחבר אותו למחשב, להריץ ולפעול עליו בזמן אמת.
- היות והמכשיר מריץ קבצי סקריפט בשפת Python, למדנו את השפה, גילינו שהיא שפת סקריפטים מונחית עצמים חזקה, ידידותית ואלגנטית וטומנת בה ספריות רבות שבאו לשפר ולעזור לממשק המשתמש.

לאחר המחקר המקיף התחלנו בעבודה על הפרוטוקול ועבדנו לפי היעדים שהצבנו לעצמנו על מנת להעמיד את הפרויקט שלנו לכדי מוצר מוגמר ומוכן לשימוש.

מסקנות נוספות מעניינות שעלו מביצוע הפרויקט :

- פעם המי יודע כמה, התגלתה החשיבות האדירה של שימוש ב GIT . במהלך הפרויקט הזה בלבד, מעבר לניהול הקוד בצורה נוחה, היה לנו נוח לראות את הקומיטים של כל אחד ואת התקדמות הפרויקט בהדרגה.
- בפרויקט זה למדנו שאין גבול למה שאפשר להשיג עם מספיק כוח רצון ומוטיבציה. בחודשים הראשונים של העבודה על הפרויקט לא חשבנו שנוכל לצבור כל כך הרבה ידע ולממש את הפרויקט לגמרי ברמת שליטה כל כך גבוהה בארכיטקטורה.
- מבחינת סיכון האי עמידה בזמנים, סיכון זה בא לידי ביטוי בכך שלא הספקנו לממש דרך מתוכננת יותר לבחירת מאסטר מבלי לגרוע בביצועי המערכת ולכן אנחנו משאירים את שני האפשרויות האלה לפיתוח עתידי שייעשה בשבועות הקרובים. מעבר לכך, הייתה עמידה בכל היעדים ועל כך אנחנו גאים מאוד !

חשוב להדגיש שמערכת שלנו בוצעה לצד הטוב ביותר שלנו אך מכיוון וזה פרויקט מחקרי המטרה שלנו הייתה יותר להראות שיש דרך להגיע לפתרון הבעיות שצינו ולא למצוא את הדרך הטובה לפתור אותה, כי בסופו של דבר אנחנו עדיין סטודנטים ולא עומדים לרשותנו הרכיבים והתשתיות החזקות ביותר.

נספחים

רשימת ספרות \ ביבליוגרפיה

1. **Internet of Things (IoT): A vision, architectural elements, and future directions**
<https://arxiv.org/ftp/arxiv/papers/1207/1207.0203.pdf>
2. **IoT gateway: Bridging wireless sensor networks into internet of things**
<http://yzhang.org/courses/20172018tcpip/papers2read/0301.pdf>
3. **The Internet of Things (IoT): Applications, investments, and challenges for enterprises**
https://s3.amazonaws.com/academia.edu.documents/41516575/The_internet_of_things.pdf?AWSAccessKeyId=AKIAIWOWYYGZ2Y53UL3A&Expires=1510592709&Signature=wdPAewiQ%2FCUyKUI8MLk8gcmo7tA%3D&response-content-disposition=inline%3B%20filename%3DThe_Internet_of_Things_IoT_Applications.pdf
4. **IoT Standards & Protocols Guide**
<https://www.postscapes.com/internet-of-things-protocols>
5. **IoT solution helps contractors effectively manage rig deployment**
<https://www.offshore-mag.com/articles/print/volume-77/issue-8/equipment-engineering/iot-solution-helps-contractors-effectively-manage-rig-deployment.html>
6. **How IoT is helping this offshore driller gain efficiencies**
<https://www.zdnet.com/article/iot-helping-offshore-driller-gain-efficiencies>
7. **Oil and Gas at the Edge**
<https://www.automationworld.com/oil-and-gas-edge>
8. **Master-slave synchronization**
<https://patents.google.com/patent/US6134234A/en>

קישורים למערכות ניהול הפרויקט ובקרת תצורה וסרטון

#	מערכת	מיקום
1.	מאגר הקוד	https://github.com/arik-le/Chips-Bits
2.	יומן	https://trello.com/b/3eGIbgko/project-schedule
3.	ניהול פרויקט	https://github.com/arik-le/Chips-Bits/issues
4.	הפצה	https://github.com/arik-le/Chips-Bits/releases
5.	סרטון גמר	https://drive.google.com/open?id=1YMerdPIFeyGWavxKKDiKXG5HIA-PPD5c

סרטון הפרויקט – אנחנו יודעים שקצת חרגנו מ-4 דקות ובכל זאת בחרנו להתחיל את הסרטון עם פתיח של דקה וחצי שאנו מקווים שתוכל לתת לכם פרספקטיבה והבנה יותר טובה של מה שאנו ניסינו לעשות כל השנה במחקר ופיתוח הפרוטוקול, אפשר לדלג ישר לתיאור שלנו אבל אנו ממליצים לא לעשות זאת, לדעתנו שווה לכם להשקיע את הדקה הנוספת וכן לצפות בו. איכות הסרטון תפצה על כך בהבטחה!

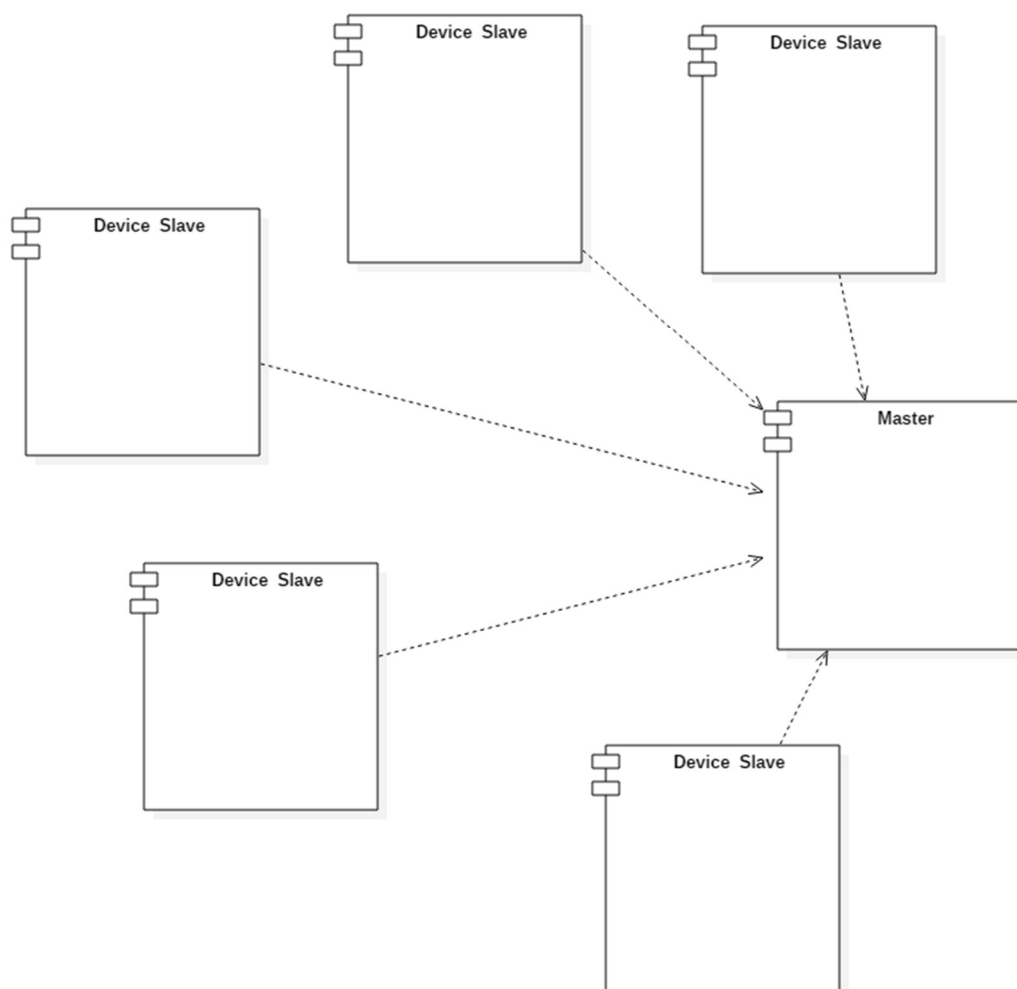
לאחר מכן, יש תיאור שלנו על הפרויקט כולל הרצת הפרוטוקול.

חשוב לציין שצפייה בסרטון לא מציגה את הפרויקט שלנו במלואו ובשל מגבלת זמן לא יכולנו להציג יותר כדי שתקבלו את התמונה המלאה.

תרשימים וטבלאות

תרשים רכיבים – components

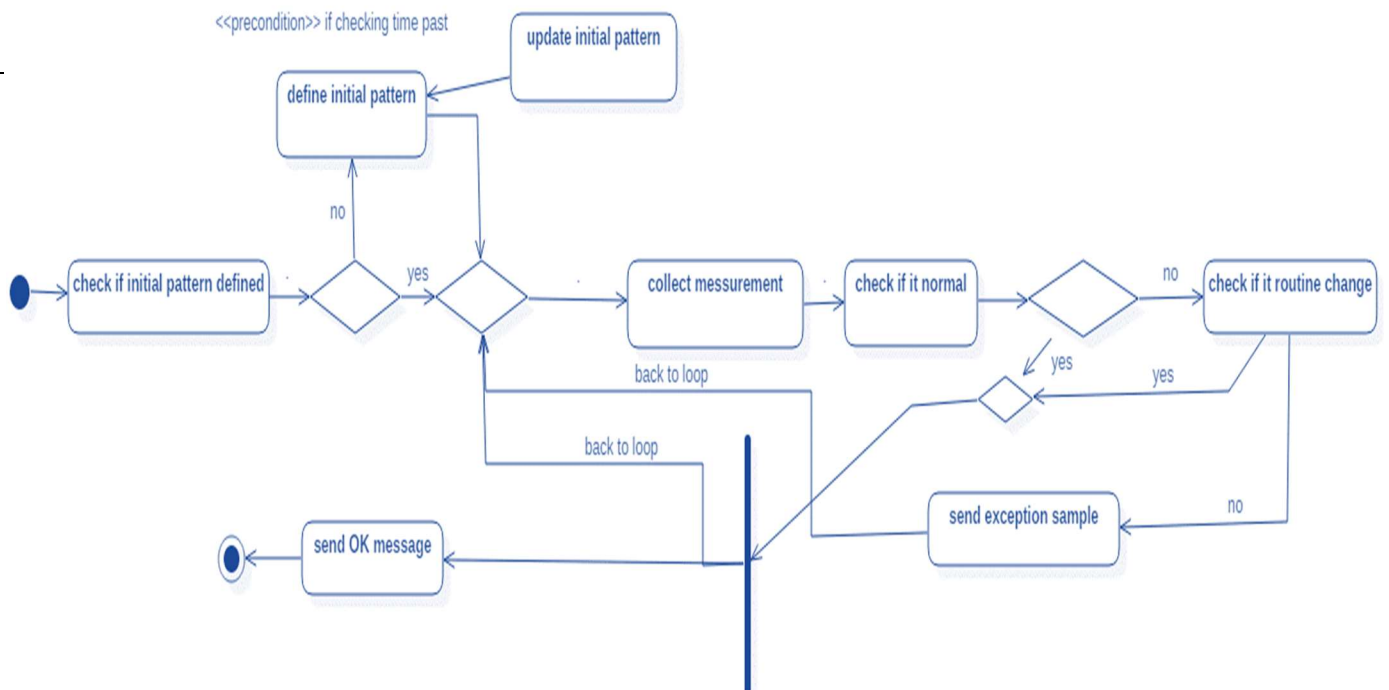
המערכת הכללית תורכב מרשת של רכיבי IOT, מתוך כל הרכיבים יוגדר אחד מהם כרכיב שליט, כל תעבורת המידע ברשת תועבר דרכו.



תרשים פעילות – Activity Diagram

בתרשים הבא ניתן לראות את התנהלות הפרוטוקול בעת פעילות המערכת. באופן התחלתי הפרוטוקול בודק אם קיימת דגימה ראשונית (כמעין ברירת מחדל) עליה הפרוטוקול יסתמך שזה מצב שגרתי ותקין, אם לא הוגדר דגימה שכזו, הפרוטוקול מתחיל במדידות ומגדיר. לאחר מכן, הפרוטוקול נכנס ללולאה תמידית בה הוא מבצע את הפעולות הבאות כל כמה זמן מוגדר מראש: האלגוריתם לוקח דגימות בזמן אמת ומשווה עם הדגימה הראשונית אם בוצע שינוי כלשהו בנתונים. אם לא בוצע שינוי האלגוריתם ימשיך בריצה. אם כן נתפס שינוי בנתונים, הפרוטוקול יבדוק האם השינוי הוא שגרתי (כמו למשל בדוגמה עם חיישן האור והשמש שמתחלשת בהדרגה), במידה וכן, ימשיך לרוץ האלגוריתם ללא שידור למאסטר. אם השינוי הוא חריגה, האלגוריתם ישלח את הנתונים שנקלטו בצורה מסודרת וברורה עם פרטים רלוונטיים למקרה שקרה. באופן דיפולטיבי, כל כמה זמן האלגוריתם יעדכן את הדגימה הראשונית על מנת למזער שגיאות נתונים ופערי סטיות לאורך זמן.

42



תכנון הפרויקט

תאריך:	התכנון:
9.10	פגישת הכרות עם המנחה
10.10	בחירת נושא והגשה לרכז
22.10	הגשת נושא הפרויקט וטופס ההתנעה
30.10	פגישת עבודה עם המנחה על שלב ההצעה
19.11	הגשת דו"ח שלב ההצעה: סקר שוק, דרישות, מהות הפרויקט, סיכונים ותרשימים. – גיא ואריק
1.2	הגשת מוצר אלפא – מימוש אב טיפוס, מסמכים נוספים וארכיטקטורת המערכת. – גיא ואריק
14.3	ניסויים בקליטת נתונים - אריק וגיא
15.3	חיבור ה-GitHub לכלי הפיתוח PyCharm-אריק
8.4	שידור נתונים בין שני רכיבים - גיא
7.5	סריקת רשת ומיפוי רכיבים - אריק
10.5	הגשת מוצר ביטא – תיכון ומימוש האלגוריתם, גרסת ביטא: ניסוי של שינוי מצב - גיא ואריק
15.5	שיפור האלגוריתם לאחר הפקת לקחים ודגשים מהמנחה - אריק וגיא
20.5	פונקציית מאסטר – האזנה -אריק
20.5	פונקציית עבד – קליטה, שידור, עיבוד - גיא
10.6	מימוש סופי פונקציות נפילת מאסטר, בחירת מאסטר חדש, שידור נתונים במקביל לחישה - גיא
11-14.6	דוח סופי, תיקון באגים והרצה סופית של תכנית בדיקות – גיא ואריק
14.6	הגשת הפרויקט
תאריך שיקבע עם המנחה	פגישת הפקת לקחים וסיכום הפרויקט

כמו כן היה לנו פגישות רבות עם המנחה במהלך השנה על מנת לתכנן תכניות, משימות ועוד.

טבלת סיכונים

#	הסיכון	חומרה	מענה אפשרי
1	לא נגיע לתוצאה שרצינו בסיום הפרויקט	גבוהה	ייעוץ נרחב עם המנחה , מעקב צמוד אחר ההתקדמות של הפרויקט בהתאם לציפיות מכל שלב , במקרה הגרוע תיאום ציפיות חדש.
2	סביבת העבודה וכלי פיתוח לא יהוו מענה ראוי למטרת הפרויקט	נמוכה	מחקר מעמיק על כלי הפיתוח , התאמת השבב לסביבת העבודה , במקרה הגרוע החלפת שבב.
3	חילוקי דעות עם המנחה לגבי מטרות ותוצאות רצויות	בינונית	עבודה משותפת ופגישות מרובות להכוונה ותיאום ציפיות עם המנחה.
4	האלגוריתם יביא לתוצאות לא צפויות	בינונית	מתן זמן רב לתיקון שגיאות וריג'קטים , בדיקות מרובות במגוון מקרי קצה ועוד.
5	האלגוריתם לא יזזה אירוע חריג שיעיד על שינוי מצב	גבוהה	תיקון שגיאות , כתיבת האלגוריתם בצורה מודולרית ומסודרת שתאפשר חזרה לאחור מבלי לפגוע ביעילותו
6	מוגבלות זיכרון תפגע ביעילות האלגוריתם	בינונית	צמצום האלגוריתם , צמצום ספריות שפחות חשובות לצורכי המערכת , עבודה שוטפת על הקוד בדגש על יעילות מקום וזמן.

טבלת דרישות

O = Optional, D = Desirable, M = Mandatory

מס' דרישה	תיאור	עדיפות	סוג
1	סביבת העבודה תזהה את השבב ותצליח לכתוב את האלגוריתם על השבב	M	פונקציונאלי
2	השבב יצליח להפעיל את האלגוריתם ביעילות מקום מינימלי	D	פונקציונאלי
3	השבב ידע לעבוד במצב תקין ולזהות שינוי מצב.	M	פונקציונאלי
4	האלגוריתם יהיה מודולרי כך שניתן יהיה להוסיף אלמנטים נוספים בעתיד שישדרגו את האלגוריתם	D	תמיכה לעתיד
5	האלגוריתם יהיה יציב לבדיקות מרובות וימנע קריסות מערכת	M	יציבות
6	האלגוריתם לעבוד באופן כללי על מגוון חיישנים בעלי פונקציונאליות שונה	O	פונקציונאלי
7	האלגוריתם ידע לשדר נתונים למאסטר	M	פונקציונאלי
8	סריקת הרשת ומיפוי הרכיבים תעבוד – לא ייווצר מצב שרכיב פעיל לא נמצא	M	פונקציונאלי
9	רכיבים נחותים יצליחו להאזין ולקלוט הודעות מערכת	M	פונקציונאלי
10	האלגוריתם ידע לזהות בזמן נפילת מאסטר	M	פונקציונאלי

Abstract

The following product is being submitted as a final project for JCE - College of Engineering - Jerusalem.

The project is a research project in the field of data processing and IOT components.

The project was designed for the research and development of a general protocol for the analysis, filtering and transmission of data in IOT components that collect data from physical sensors.

It aims to assist companies and organizations that implement IOT technology in their systems and want to reduce the use of their resources and costs when operating them.

The following document will present the introduction to the IOT world and the technological revolutions that will be caused by using this field, will present the existing problems that led to the research and development of this project, and then present the proposed ways of solving the problems through our protocol.

Finally, we will present our progress during the year to the final research and development on our chosen topic.

Software Engineering Department

Final Project - June 2018

**R&D - general protocol for the transmission of
information in IOT components**

By: Arik Levy and Guy Maimoni

Supervisor: Guy Leshem

Approved by the supervisor: [Guy Leshem](#)

Date: 13.6.18

Approved by the projects coordinator:

Date:



Software Engineering Department

Chips & Bits

48



Tamuz, 5778

June 2018