Azrieli
College of Engineering
Jerusalem

עזריאלי
מכללה אקדמית להנדסה
ירושלים

פרויקט קבוצתי

פרוטוקול דחיסה להעברת מידע לענן עבור IOT

Fig. 1. ZTE's end-to-end IoT infrastructure solution

תיאור האתגר

- בפרויקט זה, ניצר סביבת עבודה המדמה שדה חקלאי "חכם".
- נרצה שהשדה הנ"ל יתפקד בצורה הטובה ביותר כדי להפיק לחקלאי את היבול הטוב ביותר, ולשם כך נשתמש בחיישנים (התקני IOT בעלי יכולת של חישה, עיבוד ושידור מידע) שמחוברים לאורך ולרוחב השדה. כל חיישן דוגם מספר מדדים, כגון: טמפרטורה, לחות האדמה, קרינה, תזוזה וכ"ו.
- החיישנים (התקני ה-IOT) משדרים את הנתונים ל-getWay המוגדר כלקוח.
- ה-getWay מעבד את הנתונים ועל סמך פרמטרים מסוימים, יחליט איזה מהנתונים שנקלטו ע"י החיישנים הינם רלוונטיים לשליחה לענן ולטיפול המתאים (כגון – השקיה נוספת, קירור וכ"ו..).

הצגת הבעיה\רעיון המחקר

- בפרויקט קבוצתי זה, נפתח מערכת עם צד שרת וצד לקוח אשר מזרימה בכל עת נתונים המשתנים בהתאם לסביבה.
- נרצה להימנע משליחת מידע כפול או לא רלוונטי, ולכן ברוב הזמן נשלח את הדלתאות (שינויים).
- מטרתנו הינה לצמצם עד כמה שניתן את השליחה מצד לקוח לשרת.
- מימוש המטרה: בניית פרוטוקול של דחיסת נתונים.

מטרת הפרויקט

- המערכת שנפתח תציע פרוטוקול לדחיסת נתונים עבור IOT.
- המערכת תהיה בנויה בצורה כזאת שניתן יהיה להתאימה לסביבות עבודה שונות בהתאם לדרישת הלקוח, והשימוש בפרוטוקול יהיה זהה בכולם.
- המערכת תכלול בתוכה צד שרת וצד לקוח, אשר מזרים בכל עת נתונים המשתנים בהתאם לסביבה.
- נרצה להימנע משליחת מידע כפול או לא רלוונטי ולכן מטרתנו לשלוח מידע תוך צמצום הנתונים וחיסכון במקום בזיכרון (אחסון), בסוללה חשמלית של ההתקן, ובזמן הריצה של התוכנית עד כמה שניתן.
- המערכת תהיה מורכבת משני חלקים עיקריים: עיבוד הנתונים וחלוקתם ל-frame חלקי, כלומר- דלתאות, או ל- frame מלא ושליחתן מצד לקוח לצד שרת (ענן).

מילון מונחים

- **Frame מלא:** מסגרת המכילה את כל המידע בשלמותו.
- **Frame חלקי/ דלתא:** מסגרת המכילה מידע חלקי, לאחר הורדת המידע כפול או מידע שאינו רלוונטי, כלומר שהשינויים הינם זניחים.
- **getWay:** מוגדר כלקוח. מאגד את כל הנתונים אשר מתקבלים מהחיישנים, מעבד אותם ושולח אותם לשרת.
- **פרוטוקול תקשורת:** הוא נוהל לתקשורת. כלומר, אוסף של כללים המגדירים את אופן בקשת וקבלת נתונים מהלקוח לשרת.
- **Stream:** רצץ של בתים המייצג מידע שנשלח או מידע שמתקבל.
- **Socket:** השם שניתן לכל אחת משתי נקודות הקצה בקשר שנוצר. כל נקודת קצה משמשת תהליך מסוים.
- **Socket programming:** מהווה את ההתקשרות בין התוכניות אחת לשנייה, כאשר האחת פועלת בתור שרת והשנייה בתור הלקוח.

דרישות ואפיון הבעיה (1)

- נסתכל על שדה חקלאי בגודל 10 דונם למשל.
- החיישנים הממוקמים בשדה דוגמים בכל זמן את הנתונים בהתאם למדדים (כגון טמפרטורה, לחות האדמה, קרינה, תזוזה) ומשדרים אותם לgetWay.
- ה-getWay שולח לענן בכל פרק זמן מוגדר את הנתונים וכך החקלאי יראה אם יש שינוי ואם השינוי צורך טיפול.

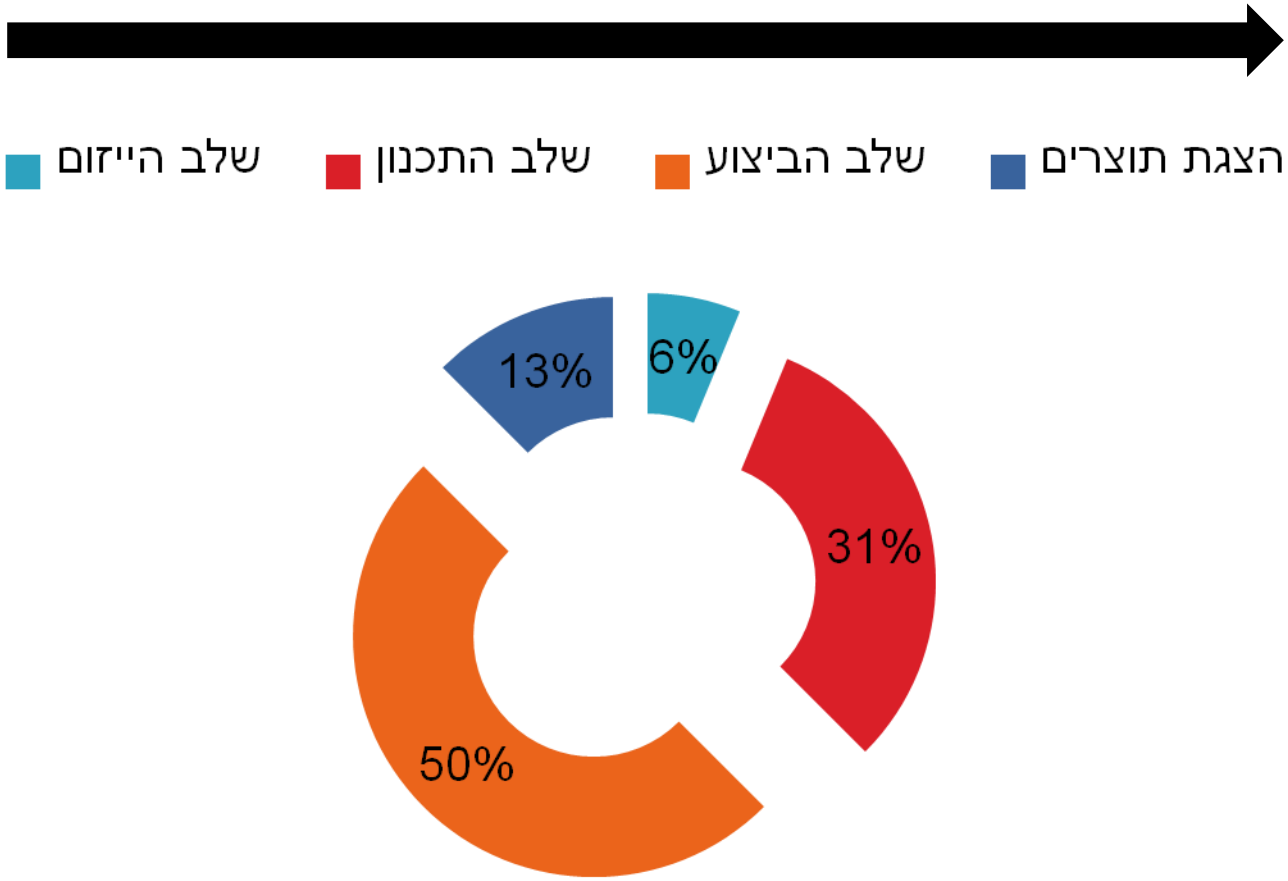
דרישות ואפיון הבעיה (2)

- ישנם מצבים שבחלק מהנתונים שבשדה לא יהיה שינוי כלל, או שהשינויים הינם זניחים מאוד.
- שליחת כל הנתונים הללו בכל עת מהלקוח לשרת, יוצרת מצב של שליחת נתונים כפולים או נתונים לא רלוונטיים לענן, מה שגורם לשליחה "כבדה" יותר.
- שליחה זו יכולה ליצור בעיות כגון: עומס יתר באחסון בענן כאשר נשלחים הרבה נתונים לא רלוונטיים, וכן בעיות בבטרייה של הענן, אשר חייה יגמרו מהר יותר כאשר השליחה לענן תהיה כבדה ברוב הזמן.

דרישות ואפיון הבעיה (3)

- נרצה להימנע משליחת נתונים כפולים או לא רלוונטיים.
- לכן מטרתנו הינה לצמצם עד כמה שניתן את המידע ולשלוח לענן רק את השינויים המשמעותיים שנמדדו, כלומר שליחת הדלתאות. ואחת לכמה זמן, נשלח לענן frame מלא, כלומר את כל המידע בשלמותו.
- הדרישה הינה יישום פרוטוקול דחיסה מקסימלית של הנתונים ושליחת הדלתאות לענן, תוך עמידה במשימה של שליחת כל הנתונים הרלוונטיים.

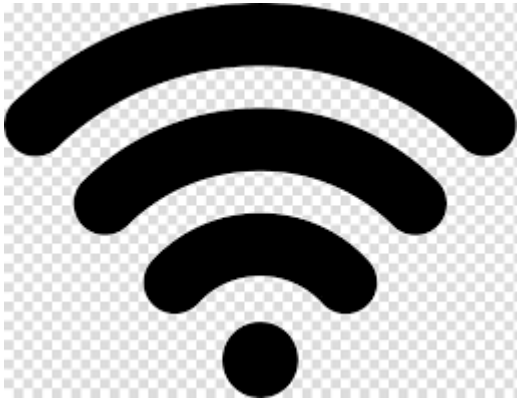
לוח זמנים לבצוע הפרויקט



הפרויקט כתהליך

Smart field

1



פרויקט 1- שדה חכם

Bin_xls=
'0011011...'

2

צד לקוח getaway

GETAWAY
HOMES FOR RENT

פרויקט 2 – צד לקוח

3

צד שרת: ענן

פרויקט 3 – צד שרת

פרויקט מס' 1 – בניית מע' הדמיה

- הבעיה הראשונה שנדרש לה הינה "בעיית סביבת העבודה".
- אנו ניצור סביבת עבודה אשר תדמה ככל הניתן סביבת עבודה אמיתית.
- סביבת העבודה שיצרנו הינה סביבת עבודה המדמה שדה חקלאי חכם.
- בשלב הראשון של עליית התוכנית, המשתמש יכניס למערכת את מספר החיישנים אשר 'יפוזרו' בשדה החקלאי, וכן את מספר המדדים שכל חיישן דוגם.
- בשלב השני, המשתמש יכניס למערכת עבור כל מדד, את הפרמטרים הבאים:

פרויקט מס' 1 - בניית מע' הדמיה

- שם המדד.
- ערכי סף עליון ותחתון קבועים לכל מדד.
- ערך סטייה עליון, וערך סטייה תחתון לכל מדד.
- ערך קריטי עליון וערך קריטי תחתון לכל מדד.

- לאחר שהמשתמש הכניס את הפרמטרים עבור כל המדדים, את המידע הנ"ל נשלח להתקני ה-IOT אשר ידווחו מידע (בצורה רנדומלית מספרים בטווח המתאים לפי ערכי הסף שהגדיר המשתמש), כלומר המערכת תתחיל לדמות סביבת עבודה אמיתית.
- לאחר מכן שוב יכנסו ערכים בצורה רנדומלית בכל פרק זמן שהוגדר מראש בכדי לדמות סביבה אמיתית ופעילה.

פרויקט מס' 2 – בניית צד לקוח

- את הנתונים אנו נאחסן במטריצה שבה מספר השורות מייצג את מספר החיישנים, ומספר העמודות מייצג את מספר המדדים עבור כל החיישנים.
- במערכת שלנו, בצד לקוח נגדיר שתי מטריצות:
  - הראשונה, מטריצת ***update*** הקולטת מהחיישנים בכל זמן מוגדר מראש את הפרמטרים.
  - השנייה, נקראת מטריצת ה-***reference*** אשר בשלב הראשון תהיה העתק של המטריצה המתעדכנת.
  - באיטרציות הבאות, בהתאם לערכי הסטיות שהמשתמש הכניס, תיעשה השוואה בין הערכים החדשים שבמטריצת update לבין הערכים שבמטריצת הreference והנתונים יעודכנו בהתאם.

פרויקט מס' 3 – בניית צד שרת

- בשלב הראשון, המערכת בצד לקוח שולחת לענן את מספר החיישנים ומספר המדדים שהמשתמש הזין.
- הענן יוצר לעצמו מטריצה אשר בדומה לשתי המטריצות הנמצאות בצד לקוח (update, reference), מספר השורות מייצג את מספר החיישנים, ומספר העמודות מייצג את מספר המדדים עבור כל החיישנים.
- בנוסף, הענן מקבל מהלקוח את מערך המדדים שלו המכיל את המידע על כל המדדים.

פרוטוקול דחיסת נתונים

שלב התחלתי- אתחול בצד שרת ולקוח

במערכת שלנו, בצד **לקוח** מוגדרות שתי מטריצות:

- האחת, מטריצת update אשר תפקידה לקלוט בכל זמן נתונים מהסביבה. מאותחלת להיות עם ערכים רנדומלים על פי ערכי הסף.
- השנייה, מטריצת ה-reference אשר בשלב הראשון תהיה העתק של המטריצה המתעדכנת.
- במטריצות, מספר השורות ייצג את מספר החיישנים, ומספר העמודות מייצג את מספר המדדים עבור כל החיישנים.

בצד **שרת** מוגדרת מטריצה אחת:

- מקבל מהלקוח את מספר החיישנים ומספר המדדים, ובהתאם לכך בונה מטריצה.
- בשלב הראשוני מקבל את מטריצת ה-update הראשונה מהלקוח.

פרוטוקול דחיסת נתונים

<u>שלב שני, 'השוטף' - סינון הנתונים</u>

- בכל פרק זמן מוגדר יש קליטה של נתונים מהסביבה לתוך מטריצת update, וסינון הנתונים הכפולים/לא רלוונטים.
- בכל איטרציה, נשווה את הערך שבמטריצת update לערך שבמטריצת ה- reference, ונסנן את המידע הלא רלוונטי על פי ערכי סטייה, והמידע הנותר זהו הדלתא שנשלח.

פרוטוקול דחיסת נתונים

<u>אופן אחסון הנתונים</u>:

את הנתונים אשר נשלח לענן, נסווג לשני אופנים:

- <u>Frame מלא</u> - כל המידע בשלמותו. frame מלא יאוחסן בתוך מטריצה.
- <u>Frame חלקי</u> - דלתא אשר מכילה חלק מהנתונים, כאשר הנתונים שסוננו, הינם נתונים כפולים או לא רלוונטים שהשינוי בהם היה זניח בהתאם לסטייה שהכניס הלקוח.
- הדלתא תאוחסן בתוך וקטור, כך שהמיקום הראשון יכיל את מיקום הערך (i,j) שיידחסו למשתנה מסוג short, והמיקום השני יכיל את הערך.

פרוטוקול דחיסת נתונים

דחיסת אינדקסים:

- אנו מעוניינים בשליחה מצומצמת עד כמה שניתן במסגרת דחיסת נתונים בצורה מקסימלית. ולכן יש שימוש בדחיסת אינדקסים.
- הדחיסה תיעשה על המיקום של הערך (i,j), כלומר נדחוס את j ו-i לערך אחד ולתא אחד בווקטור.
- ניצור משתנה מסוג short, אשר תופס בזיכרון 2 bytes. כל אינדקס נשלח ב-8 סיביות. כאשר אינדקס i יאוחסן ב-8 סיביות עליונות, ואינדקס j יאוחסן ב-8 סיביות התחתונות של המשתנה.

```
else{
    if(num>arrIndex[j].getDeviation_Sup()){
        referenceMatrix[i][j]=updateMatrix[i][j];
        short temp = (short) (i<<8 | j);
        vectorToSend.add((float)temp);
        vectorToSend.add(updateMatrix[i][j]);
    }
```

- החיסכון שנקבל: חסכון של 16 סיביות בדחיסה לערך יחיד של short.

פרוטוקול דחיסת נתונים

פיענוח דחיסת אינדקסים:

- נחלק את i ב 2^8 בשיטה יעילה. בהנחה שכל אינדקס הוא מקסימום 2^{8-1} .
- נסתכל על הסיביות העליונות- כאשר מזיזים 8 סיביות אחורה נקבל את i. עושים and לוגי עם הסיביות העליונות לאחר שאופסו בשביל לקבל את הערך של j.

```
for(int i=0; i<getVector.size()-1; i+=2){
    int iIndex, jIndex; float value;
    short temp = (short) Math.round(getVector.elementAt(i));
    iIndex = temp >> 8;
    jIndex = temp & 255;

    value=getVector.elementAt(i+1);
    matrixOfValueIndex[iIndex][jIndex]=value;

}
    getVector.clear();
```

פרוטוקול דחיסת נתונים

<u>שלב שלישי- אופן השליחה לענן</u>

– נשלח את כל ה-frame (כל המידע) שיאוחסן בתוך מטריצה במקרים הבאים:

- במידה וכמות הנתונים שבווקטור עולה על 66% מסך כל המידע.
- לאחר ששלחנו מספר מוגדר של frame-ים חלקיים (למשל 30 דלתאות).

- בכל איטרציה נאחסן את הווקטור/מטריצה בbuffer. נשלח את כל הbuffer לשרת במקרים הבאים:
  - כאשר ה buffer יתמלא.
  - כאשר נקבל פרמטר במדד מסויים שעבר את גבול הערך הקריטי.
- השרת יקבל את ה-buffer עם כל האובייקטים שבתוכו.

שימוש בפרוטוקול TCP/IP

- השליחה מצד לקוח לצד שרת תיעשה באמצעות אובייקט מחלקה Socket.
- ה-Socket מהווה נקודת קצה במחשב הלקוח ליצירת תקשורת עם נקודת קצה השנייה- השרת.
- ה- Socket מחזיק stream אשר מהווה רצף של בתים שמייצג מידע שנשלח או מידע שמתקבל.