**מאפייני הפרוטוקול - שלב קליטת הנתונים + עיבוד הנתונים**

תאריך עדכון : 20.01.18

שם הפרויקט: פיתוח פרוטוקול העברת מידע מינימלית ברכיבי IOT.

מחברים: אריק לוי וגיא מימוני

כפי שתיארנו במסמכים קודמים, הפרוטוקול שלנו יורכב משלושה שלבים שונים : קליטת הנתונים (מרכיב הפיזי לתוך התקן ה IOT ) , שלב עיבוד הנתונים ( שלב עיבוד וניתוח המידע המתקבל, קבלת החלטה אם היה שינוי מצב והאם לשלוח את המידע למרכז הבקרה ) , שלב שליחת הנתונים ( למרכז הבקרה ).

אנו נתעסק עכשיו בשלב קליטת הנתונים שהוא שלב מאד קריטי ולשם עיבוד המידע וקבלת החלטות, אנו צריכים שהמידע ייקלט ויעבור להתקן IOT במלואו וכמובן שיעבור בצורה אותנטית.

כפי שרשמנו בעבר אנו נשתמש בכלי הפיתוח להתקני IOT – Arduino על מנת לחבר בין כל הרכיבים , כלי הפיתוח הנ"ל משמש כמעין מערכת הפעלה אשר מקשרת בין כל הרכיבים, ממפה את כל הכניסות והיציאות של השבב.

בנוסף נשתמש ב WinScp על מנת לצרוב קבצי Python אשר יכילו את האלגוריתם לפרוטוקול ובהם נקלוט את המידע ונעבד אותו .

לשם קבלת נתונים מהתקן הפיזי נדרש חיבור פיזי להתקן IOT .

קריאת המידע מההתקן החיצוני לתוך התקן ה IOT תבוצע על ידי מספר פקודות ספציפיות שרלוונטיות לכל התקן חיצוני (למשל: מצלמה , חיישן משקל , חיישן קול ועוד..)

האלגוריתם שיותקן בתוך השבב יאזין למידע שיכנס דרך כניסות מסוימות (input) כמו למשל כניסת micro - USB למצלמה או מידע שיועבר דרך הפינים המחוברים לשבב ואפשר לחבר להם חיישנים דרך כבלים או מטריצה .

מצורף למטה דוגמה הנותנת לנו המחשה לקישור בין כלי הפיתוח Arduino לבין תכנית הכתובה בפיתון אשר תשב על השבב עצמו .

*להלן דוגמה לתכנית ב Arduino הקוראת מפין 7 ערך וכותבת אותו בפין 13.*

int outPin = 13; // LED connected to digital pin 13

int inPin = 7; // pushbutton connected to digital pin 7

int val = 0; // variable to store the read value

void setup()

{

pinMode(outPin, OUTPUT); // sets the digital pin 13 as output

pinMode(inPin, INPUT); // sets the digital pin 7 as input

}

void loop()

{

val = digitalRead(inPin); // read the input pin

digitalWrite(outPin, val); // sets the LED to the button's value

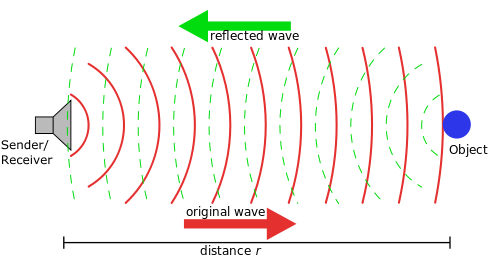
}

ועכשיו לקטע קוד בפיתון :

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | import pyfirmata    board = pyfirmata.Arduino('COM3')  it = pyfirmata.util.Iterator(board)  it.start()    sw = board.get\_pin('d:7:i')  led = board.get\_pin('d:13:o')    while True:      value = sw.read()          led.write(value)  board.exit() |

**בהמשך אנו נראה ונדגים קליטת נתונים מחיישנים אשר חוברו להתקן ה-IOT שלנו** .

**עבודה עם חיישן מרחק HC-SR04**

לשימושנו עמד חיישן מרחק של חברת Grove בשם HC-SR04 אשר חיברנו אל שבב ה – IOT שברשותנו (Linkit smart 7688 duo) .

החיבור נעשה באמצעות לוח ניסוי (מטריצה) לחיבור רכיבים אלקטרונים באמצעות כבלי מתח ופינים, כבל אחד שהוא Input חובר לפין מספר 12 , וכבל אחר שהוא Output חובר לפין מספר 13.

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 ).

חשוב לציין שהחיישן קולט מרחק בטווחים 2-400 ס"מ.

המטרה שלנו הייתה כמובן לתפוס את המרחק עד לאובייקט בשטח הפתוח במדדים תכופים ולאורך זמן, ולכן מדדנו בהפרשי זמן קבועים מרחק על אובייקט דומם.

לצורך יצירת סטנדרט התחלתי שממנו נוכל לתפוס את השינוי ולהשוות , מדדנו מספר מסוים של מדידות לאורך זמן וחישבנו ממוצע על מנת שנוכל להשוות בהמשך .

הממוצע הנ"ל שימש כתבנית התחלתית שידמה מצב שגרה בעת הרצת הפרוטוקול הכללי שנרצה לפתח ולממש בסיום הפרויקט .

לאחר שקיבלנו את מצב השגרה שלנו , התחלנו להריץ מדידות מרחק בזמנים קבועים וקבענו סף מסוים כך שאם המדידה תחרוג ממנו ייתכן שינוי מצב, את הסף קבענו באופן זמני ורלוונטי לניסוי הנ"ל – ממוצע התבנית ועוד X ס"מ או פחות X ס"מ , הכוונה היא שאם המדידה חרגה ביותר או פחות מ X ס"מ למצב התקין שקבענו , המערכת זיהתה שינוי מצב , את ה X שינינו כל ניסוי שביצענו .

בכל ניסוי שביצענו עם X שונה הצלחנו להשוות ולתפוס את שינוי המצב עם המצב הרגיל, כמובן שכל שינוי מצב נתפס בזמן שונה .

את הקוד עצמו הצלחנו לכתוב ולהתאים לשבב שלנו עם החיבורים הספציפיים המתאימים לו לאחר מחקר מעמיק באינטרנט והרבה ניסוי וטעייה, כמובן שזה היה לא קל היות וכל התחום הזה זר לנו ולא נלמד בשום קורס שלמדנו במסגרת התואר, הרבה מן החומר נוגע יותר לתחום האלקטרוניקה, דבר שלא קשור לתחום הנדסת התוכנה שאנו לומדים במסגרת הלימודים בהשכלה הגבוה.

ניסוי 2 – בדיקת השינויים לאורך זמן – מציאת הדלתא בין השינוי הקודם לעדכני

לניסוי הבא רצינו לבדוק את השינויים לאורך זמן עם השינויים שקורים בדרך כלל בשגרה כמו שמש שזורחת או שוקעת לעיטה ולכן אם נמדוד בחיישן אור אז ככל שהזמן עובר מדידת האור תשתנה וזה כמובן מצב שגרתי ולא משהו חריג שמצריך בדיקה ועיבוד נתונים, דבר שהפרוטוקול שאנו מפתחים מבצע.

השתמשנו בחיישן המרחק שברשותנו ומדדנו מרחקים שונים לאורך זמן ( כמספר דקות) , בין כל מדידה למדידה ביצענו חישוב דלתא: Delta= Mi – Mi-1 וכמובן שמרנו את זה במערך כדי שיהיה לנו עם להשוות בהמשך .

בסימולציה שביצענו רצינו לראות מצבים שונים של מצב שגרה כך שבכל פעם שמודדים משתנה משהו מסוים במדידות בצורה שגרתית ולא חריגה כדי לדמות מציאות כלשהי כמו למשל הדוגמה עם השמש שהסברנו מקודם שבמשך היום הראות וחוזקת האור עולה ויורדת בהתחשב למצב ושעות היום.

היות וחיישן מרחק קצת מגביל אותנו לאפשר מציאות כלשהי ,התערבנו בתהליך המדידות שלו ושינינו את פני השטח אותם החיישן מודד, הזזנו אובייקטים בזמנים שונים ובמרחקים שונים על מנת שנצליח לקבל אינדיקציה מסוימת נחרצת שתדמה מציאות שגרתית .

לאחר שביצענו את כל המדידות ההתחלתיות וחישובי הדלתא, הגענו לשלב השני של הניסוי שזה מדידה וחישוב השינוי לאורך זמן על מנת לתפוס בזמן אמת **שינוי מצב** .

המדידות בוצעו באותם זמנים שביצענו את המדידות בשלב א' על מנת שיהיה תאימות בין החישובים הקודמים והעדכניים , כמובן שאת שלב ב' ביצענו כמה פעמים לפחות על מנת לקבוע בכל פעם ולבדוק אם האלגוריתם קולט בזמן אמת שינוי מצב כפי שאמור להיות או ממשיך כרגיל כאשר אין באמת שינוי מצב כפי שאנו מצפים שיהיה.

**סיכום:**

קיבלנו תובנות רבות מניסוי זה , שימוש בחיישן מרחק הוא לא קל היות וישנם חישובים רבים על מנת לקבל את המרחק המדויק בזמן אמת ולא בדילאי כלשהו הנותן לנו את המסקנה שהנתונים לא בהכרח נכונים.

את חישובי הדלתא ביצענו על מספר סימולציות בזמן אמת כדי לבדוק אם האלגוריתם תופס את השינוי – כשהוא צריך , ולא תופס את השינוי - כשהוא לא צריך, כמובן הרצנו את זה כמה פעמים כדי לשלול כל אפשרות של פלט לא תקין ונתונים שקריים שיעידו על אי תקינות האלגוריתם .

לאחר כל הסימולציות שביצענו הגענו לתובנה שהאלגוריתם עובד כמו שצריך בשלב זה וכן הצלחנו למצוא ולתפוס את שינוי המצב במערכת בזמן אמת ומתי שרצינו שיתפוס ( מתי שאמור להיות שינוי מצב לפי הסימולציה הנכונה שהרצנו), כמובן הרצנו עוד מספר סימולציות וציפינו שהאלגוריתם לא יתפוס שינוי מצב כי כמובן לפי הנכונות שלנו בסימולציות האלו לא בוצע באמת שינוי מצב בפני השטח .

יצאנו מאד מרוצים מהתוצאות ומסיכום השלב הזה , כמובן שנרצה לעשות בעתיד עוד ניסויים על מנת להכריע בצורה נחרצת את נכונות ותקינות האלגוריתם בחיישנים אחרים ובשלבים אחרים , כגון שלב שידור הנתונים במקרה שיש שינוי מצב חריג שמצריך העברת מידע.