Porting MQTT-SN support to cc1350

מגישים:

שלמה ווקנין , ת.ז. 311130678

דוד ליפשיץ , ת.ז. 206107740

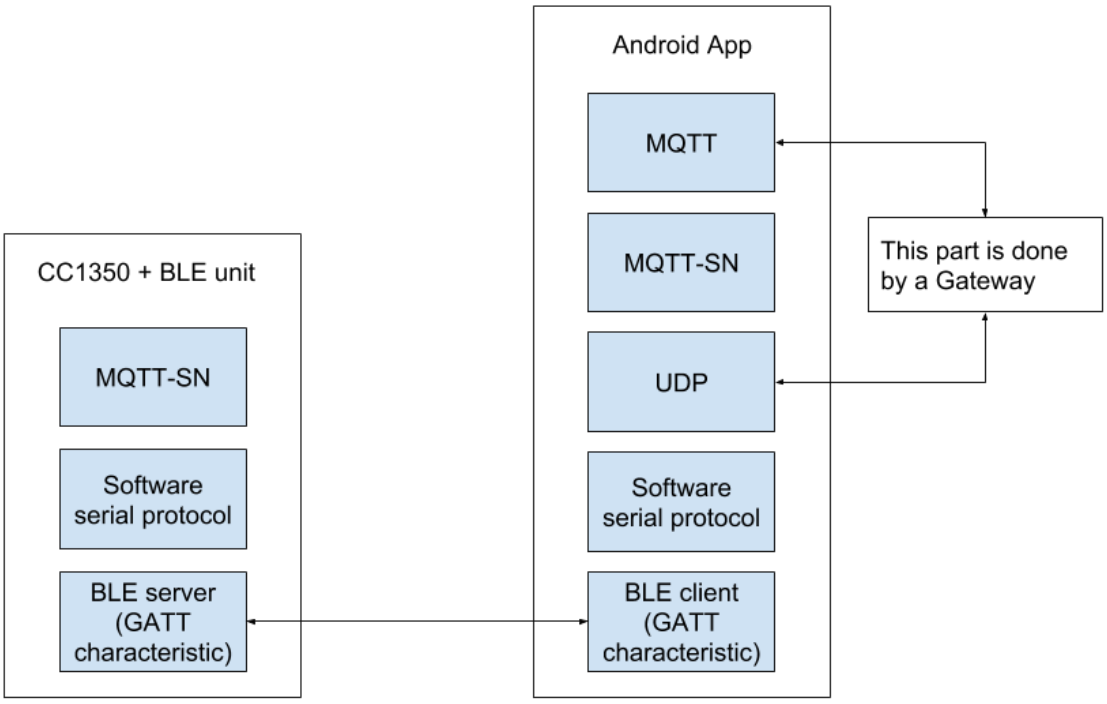
הפרויקט בנוי משני חלקים:

1- ה-cc1350, עליו יושב קוד המממש את MQTT-SN ושולח את הודעותיו על גבי BLE (Bluetooth low energy).
2- קוד היושב על Android, מקבל את ההודעות היוצאות מהבקר, מתרגם אותם להודעות MQTT ושלוח אותם אל broker (על גבי TCP).

ניתן למצוא את שני החלקים בrepository הבא:

https://github.com/davidLif/MQTT_over_BLE

המבנה הלוגי של התקשורת הוא כדלקמן:



MQTT-SN on the board:

על מנת לבצע porting של mqttsn בבקר, השתמשנו בקוד פתוח שמופיע בקישור:

https://github.com/eclipse/paho.mqtt-sn.embedded-c

הספרייה מכילה פונקציות שמאפשרות connect, disconnect, publish, subscribe שנשלחות אל mqttsn-gateway, שמעביר אותן אל הbroker. על מנת להטמיע את הספרייה נדרשנו לממש מספר פונקציות של שליחה וקבלה. המימוש של פונקציות אלה נעשה באמצעות שליחה וקבלה של BLE.

על מנת להפריד את שילוב הBLE והקוד עבור MQTTSN , תחילה שילוב mqttsn נעשה מעל גבי פרוטוקול uart. זה אפשר גמישות בפיתוח ושימוש במחשב בלבד ללא צורך לכתוב אפליקציה לאנדרואיד.

Software serial protocol over BLE:

interface – Software serial protocol עבור שליחה וקבלה של מסרים בצורה בסיסית. לכל צד ישנה פונקציית כתיבה מסוג אחד וקריאה מסוג אחד. נשלח raw byte array ואורך ההודעה.

הכתיבה מהEmulation board אל הAndroid נעשית בעזרת notifications מcharacteristic יחיד:

הboard הוא הBLE server בקשר שבין השניים, ולכן כל פעם שהוא רוצה לשלוח הודעה ל Android, הוא פשוט שולח notification דרך הcharacteristic בשם Tx_channel.

הכתיבה מכיוון הAndroid לEmulation board מורכבת מ2 characteristic :

1- Rx_val - המכיל את תוכן ההודעה, את אורכה ואת הserial number שלה.
2- Rx_ack - משתנה זה מכיל את ערך הack של הserver, לאחר שהוא קרא את ההודעה של הclient.

כיוון שהandroid הינו במקרה הזה הBLE client, הפעולה היחידה שהוא יכול לעשות והserver יוכל לראות הינה כתיבה לתוך GATT characteristic המוגדר כwriteable.

כאשר הandroid רוצה לכתוב לboard, הוא כותב את הערך הרצוי לתוך הRx_val, כאשר הbyte האחרון מחזיק את הserial number של הפקטה ואת אורך הmessage שהAndroid רוצה לשלוח לboard (הפעלה של MASK עם 0x1F על הbyte האחרון של תיתן את האורך. שאר הביטים של הbyte האחרון מכילים את הpacket serial num).

על הboard להאזין לכתבות לתוך המשתנה Rx_val, ולאחר הקריאה לכתוב את ערך הbyte האחרון לתוך הRx_ack כסימן מוסכם לכך שההודעה נכתבה וניתן לדרוס את Rx_val עם ערך חדש.

הpacket serial num נועד בכדי לזהות "כתיבות חסרות" שלא הגיעו אל הboard ולהוות אינדיקציה לכתיבה חדשה, גם אם נכתב אותו התוכן כמו בהודעה הקודמת.

הודעות 2 הצדדים מוגבלות בגודלן ל19 בתים.

עבור התקשרות הAndroid מול הBLE נעזרת בקוד ורעיונות מ
https://github.com/googlesamples/android-BluetoothLeGatt

Software serial to UDP:

התרגום מsoftware serial לUDP מתבצע באופן טריוויאלי – הודעת יחידה מתרגמת לפקטת UDP יחידה.

השליטה בפורטים של השליחה והקבלה נעשית בעזרת פרמטרים הנמצאים ב mqttgatewayparam.xml , קובץ המשמש עבור קונפיגורציית הUDP והGateway.

MQTT-SN gateway:

הקוד עבור הgateway נלקח (בכמה שינויים קטנים, כמו כתיבת לוגים וקונפיגורצייה) מהפרויקט:

https://github.com/jsaak/mqtt-sn-gateway/tree/master/apps/MQTTSN-Gateway

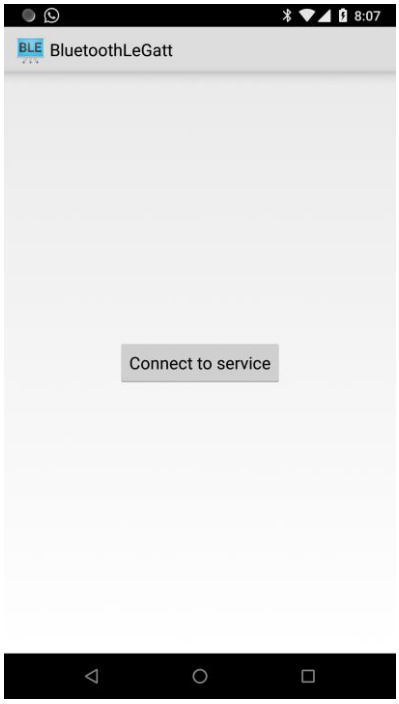
בערוץ אחד, הgateway מאזין לפקטות UDP המכילות את הודעות הMQTT-SN, מחלץ מתוכן את התוכן, מתרגם אותן לפקטות TCP, ושולח אל הbroker. בערוץ השני, מתבצעות כל אותן הפעולות אבל הפוך (בערוץ זה הודעות הbroker מתורגמות להודעות MQTT-SN על גבי פקטות UDP.

קונפיגורציית הgateway נשמרת בקובץ mqttgatewayparam.xml .

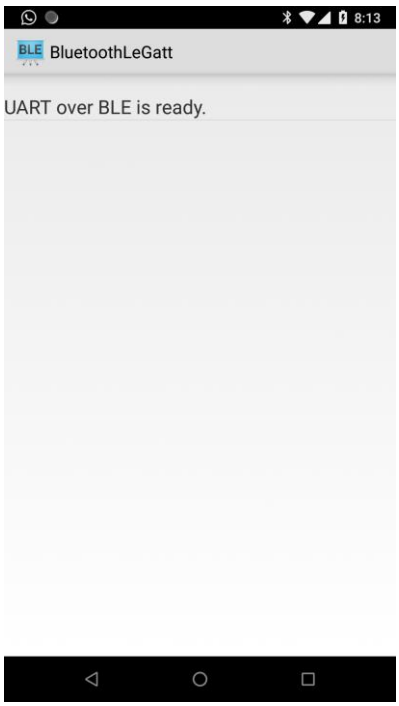
מעטפת אפליקציית הAndroid:

*) לפני הפעלת האפליקציה, חשוב לתת לה את ההרשאות שהיא דורשת (Bluetooth ו-location). יש להדליק את הlocation והbluetooth לפני ניסיון ההתחברות אל הboard. (הlocation זוהי דרישה של Android, שכן ניתן להשיג את מיקום משוער עליך בטווח מסיום מחיבור הbluetooth).

מסך ראשי – ממסך זה ניתן לנסות להתחבר אל הboard. במידה והניסיון יכשל, תקפוץ הודעת שגיאה.



מסך לוגים ופעילות האפליקציה - במסך זה האפליקציה נמצאת בזמן שכל הNet stack שבנינו פועל.
לוגים בסיסיים נכתבים אל הUI. לוגים מפורטים יותר ניתן לראות דרך הLogcat.



<u>הרצה לדוגמא:</u>

לאחר חיבור בין הפלאפון והבקר בפרוטוקול BLE, הבקר מתחבר אל הברוקר של mqtt, ומבצע publish כל שניה.
כל ההודעות עוברות דרך הgateway.

הברוקר שבו השתמשנו הוא השרת iot.eclipse.org. זהו שרת עבור מפתחים. בנוסף, eclipse מספקת כלי שבו ניתן "לרגל" על התעבורה של mqtt שמגיעה אל השרת, ומאפשרת לבצע Publish לנושאים שמעוניינים. הכלי נקרא mqtt-spy (בהמשך תהיה תמונה).

הסנפנו את התקשורת מול השרת באמצעות אפליקציה שנקראת packet capture, והעברנו אל המחשב ופתחנו בwireshark. זו התמונה שקיבלנו:

	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.0.10	198.41.30.241	TCP	54	32763 → 1883 [SYN] Seq=0 Win=4096 Len=0
2	0.000100	198.41.30.241	192.168.0.10	TCP	54	1883 → 32763 [SYN, ACK] Seq=0 Ack=1 Win=4096 Len=0
3	0.000200	192.168.0.10	198.41.30.241	TCP	54	32763 → 1883 [ACK] Seq=1 Ack=1 Win=4096 Len=0
4	1.686000	192.168.0.10	198.41.30.241	MQTT	80	Connect Command
5	1.870000	198.41.30.241	192.168.0.10	MQTT	58	Connect Ack
6	6.422000	192.168.0.10	198.41.30.241	MQTT	69	Publish Message
7	11.415000	198.41.30.241	192.168.0.10	TCP	54	1883 → 32763 [ACK] Seq=5 Ack=42 Win=4096 Len=0
8	11.415000	192.168.0.10	198.41.30.241	MQTT	69	Publish Message
9	16.406000	198.41.30.241	192.168.0.10	TCP	54	1883 → 32763 [ACK] Seq=5 Ack=57 Win=4096 Len=0
10	16.406000	192.168.0.10	198.41.30.241	MQTT	69	Publish Message
11	21.382000	198.41.30.241	192.168.0.10	TCP	54	1883 → 32763 [ACK] Seq=5 Ack=72 Win=4096 Len=0
12	21.382000	192.168.0.10	198.41.30.241	MQTT	69	Publish Message
13	26.302000	198.41.30.241	192.168.0.10	TCP	54	1883 → 32763 [ACK] Seq=5 Ack=87 Win=4096 Len=0
14	26.302000	192.168.0.10	198.41.30.241	MQTT	69	Publish Message
15	31.360000	198.41.30.241	192.168.0.10	TCP	54	1883 → 32763 [ACK] Seq=5 Ack=102 Win=4096 Len=0
16	31.360000	192.168.0.10	198.41.30.241	MQTT	69	Publish Message
17	36.340000	198.41.30.241	192.168.0.10	TCP	54	1883 → 32763 [ACK] Seq=5 Ack=117 Win=4096 Len=0
18	36.340000	192.168.0.10	198.41.30.241	MQTT	69	Publish Message

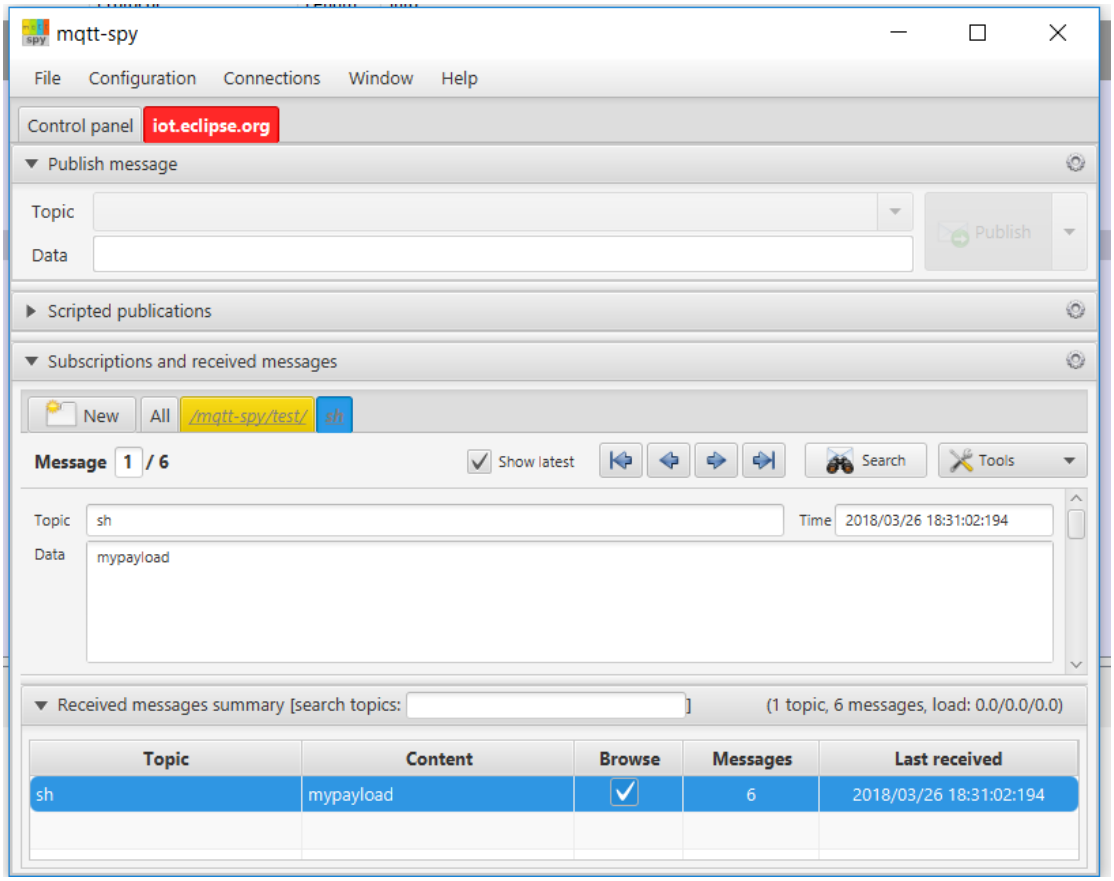
ניתן לראות כי השרת קיבל בקשת התחברות בפרוטוקול mqtt ובוצע publish. לא התקבל ACK משום שעבדנו בqos0.

הסנפה של חבילת הpublish:

```
> Transmission Control Protocol, Src Port: 32763 (32763), Dst Port: 1883 (1883), Seq: 42, Ack: 5, Len: 15
v MQ Telemetry Transport Protocol
  v Publish Message
    > 0011 0000 = Header Flags: 0x30 (Publish Message)
      Msg Len: 13
      Topic: sh
      Message: mypayload
```

ניתן לראות כי ביצענו publish לנושא sh עם תוכן mypayload.

על מנת לוודא שאכן ההודעות הגיעו אל השרת כמו שצריך פתחנו במחשב אחר mqtt-spy וביצענו subscribe לנושא sh. זה מה שהתקבל:



רואים כי אכן התקבלו 6 הודעות בנושא sh.

לקחים שנלמדו על תכנות embedded מכתיבת האפליקציה:

1- לעיתים debug בצורתו ה"קלאסית" הוא בעייתי (הפרעות בזמנים לפרוטוקולי תקשורת, נדרש קמפול עם פרמטרים שונים וכדומה). לעיתים משתמשים ב"עזרים" חיצוניים בכדי לגעת מה קרה בקוד, כמו לדים או כתיבה למשתני BLE אותם ניתן לראות מבחוץ.
2- פעולות "מקביליות" – במידה וקיימת "מקבליות", תצורת העבודה אתה שונה במידה מה אשר באפליקציות רגילות. דבר זה לעיתים מסבך פעולות רגילות כמו האזנה ושליחה בו זמנית.
3- הקפדה על גודל הstack – הstack אינו "אינסופי" ויש להקצות משאבים בהתאם לפונקציות וללוגיקה שכל task מבצע.

נקודות לשיפור:

1- כתיבת קוד כך שתפקידי הBLE יתחלפו בין הAndroid לboard (הserver על הandroid, והclient על הboard) בכדי לאפשר לכמה לוחות לתקשר בו זמנית מול android יחיד. לא הצלחנו לממש לבד BLE client על הboard.
2- Error handling – משני הצדדים הerror handling ברמת הSoftware serial protocol-BLE היא די נמוכה. צריך להרחיב את המעטפת כך שתתמוך בצורה טובה יותר בנפילות תקשורת (כמו שמירה על הpacket serial num).