PillBOX – תהליך כתיבת המערכת

נעה יפה, אמיר ג׳ונפור, פלג נויפלד

במסמך זה נסקור את תהליך יצירת המערכת לפי מרכיביה. עבור כל רכיב נציין כיצד כתבנו את הקוד (פלטפורמה, עקרונות מיוחדים), ונרחיב על קשיים שנתקלנו בהם במהלך הכתיבה במידה והיו כאלה.

I. Texas Instruments CC1350 LaunchPad – מריץ את אפליקציית ה-BLE ואת לוגיקת המערכת והחיישנים:

לוגיקת המערכת והחיישנים:

בכניסה למערכת הלוח משתמש במידע שקיבל מהאפליקציה כדי לאתחל את לוגיקת התזמון – מחשב את הזמן לזמן נטילת הכדורים הראשונה ואת המרווחים בין זמני הכדורים בבוקר, צהריים וערב. כשמגיע זמן נטילת כדורים נדלק לד שמחובר לתא הרלוונטי (לצרכי הפרויקט – תא יחיד), במשך זמן ההמתנה הלוח דוגם כל שנייה את החיישן HALL של התא. אם הלוח דוגם את התא ורואה שהתא פתוח הוא מסמן שהתא נפתח ומכבה את הלד שעל התא. אם זמן ההמתנה עובר והחיישן הרלוונטי לא מזהה פתיחה של התא – הוא מדליק פין GPIO שמחובר ללוח ה-ESP8266 והוא בתורו מפעיל את הלוגיקה של שליחת ההתרעות.

בתהליך הפיתוח נעזרנו בכלי SCS (Sensor Controller Studio) שמתממשק עם ה-CCS ונועד לעבודה עם חיישנים. נתקלנו בקשיים כיוון שהפרוייקטים שייצאנו לא תמכו ולא אפשרו ייבוא של ספריות הנחוצות ללוגיקה של המערכת ולכן לא השתמשנו בקבצים שייצר ה-SCS, אבל העבודה עם הממשק עזרה לנו להבין את העבודה עם החיישנים.

תחילה הלוגיקה של ניהול הזמן הייתה ב-thread נפרד, אבל השיטה הזאת הייתה פחות אמינה – ת׳רד לא תמיד התעורר מיד ולכן המעקב אחר הזמנים השתבש. עברנו לשיטה של דגימת החיישנים כל שנייה (בהנחה שלא ניתן לפתוח תא, להוציא את הכדורים ולסגור את התא תוך פחות משנייה).

II. Adafruit feather HUZZAH ESP8266 – מחובר ללוח ה-CC1350 בשביל יכולת Wi-Fi:

ברכיב זה השתמשנו עבור תקשורת Wi-Fi עם השרת שלנו באמזון כדי לממש את שליחת התראות ה-SMS. הקוד של רכיב זה נכתב ב- Arduino IDE בסביבת Linux. בהתחלה רצינו לממש תקשורת מלאה בינו לבין ה-CC1350 (הרחבה בהמשך), ובהמשך עברנו למצב בו הרכיב פותח hotspot כדי לאפשר למשתמש להגדיר פרטי Wi-Fi ולאחר מכן הוא מאזין על פין 14 שמחובר בצד השני לCC. הערך הדיפולטי שהוא קורא הוא 1, וברגע שה- CC מעוניין לשלוח התראת SMS, הוא מוריד את הערך בחיבור זה ל- 0. ה- ESP קולט את ירידת הערך, ושולח בקשת POST ל- API שלנו בשרת אמזון. פרטי הזיהוי של הלוח והשרת צרובים בקוד של ה- ESP ולא ניתן לשנותם (בדומה לפרטי הזיהוי של הלוח שצרובים ב- CC1350).

III. אפליקציית האנדרואיד PillBOX – אפליקציה המאפשרת לרוקח להתחבר ללוח ה-CC1350 ולהגדיר את הזמן הנוכחי ואת שעות נטילת התרופות:

האפליקציה נכתבה ב-Java ופותחה ב-Android Studio IDE, והשלד שלה מבוסס על דוגמה לאפליקציית BLE מתוך מאגר דוגמאות של גוגל (קישור לדוגמה).

כתיבת הקוד לאפליקציה הייתה פשוטה יחסית, בין היתר בזכות הפופולריות של אפליקציות כאלה שיצרה קהילת מפתחים ענפה באינטרנט, עם הרבה דוגמאות קוד והדרכות.
העיכוב היחיד בתהליך פיתוח האפליקציה היה שלא ניתן להשתמש באמולטור של אנדרואיד (תוכנת מחשב שמותקנת בה מערכת ההפעלה של אנדרואיד, ומאפשרת לטעון אליה את האפליקציה ולבדוק אותה בסביבת debug), כיוון שלא ניתן להשתמש בחיבור Bluetooth באמולטור. לכן בכל פעם היה צריך לטעון את הגרסה העדכנית של האפליקציה למכשיר אנדרואיד (פיזי) ולאפשר מהמכשיר מצב debugging.
כמו כן, בדיעבד אנו מבינים שמבחינת מהירות הכתיבה והנוחות היה עדיף להשתמש ב-SDK של אפליקציות BLE לאנדרואיד, שנותן תבנית נוחה של פונקציות נפוצות, ומאפשר לדלג על החלק התכנותי הבסיסי ולהתמקד בפונקציונאליות שנדרשה לאפליקציה שלנו. מצד שני, ההתנסות בכתיבה הייתה חשובה ומלמדת גם היא.

IV. PillBOX Website – מאפשר ללקוח להגדיר רשימת אנשי קשר ולאדמין לפקח על כלל הלקוחות. האתר נכתב בשפת Python עם חבילת Flask (סביבת פיתוח אתרי WEB) ו- boto3 (חבילת ה- API של אמזון לPython). ממלא תפקיד כפול – אתר עבור לקוחות ואדמין, וAPI עבור התראות שמגיעות מה- ESP, והופעל על גבי פלטפורמת EC2 של Amazon AWS (שרת ubuntu 16.04). הבחירה באתר אינטרנט מאפשרת לשנות את ההגדרות מעל כל פלטפורמה שמסוגלת לגלוש (מחשב, סמארטפון, וכו') בניגוד לאפליקציה שמגבילה את המשתמש לסמארטפון.

V. Amazon SNS – שירות נוטיפיקציות שמאפשר שליחה לרשימת אנשי קשר שהוגדרה מראש. השירות עובד בתצורת publish-subscribe, כאשר כל לוח הוא topic בפני עצמו, ואנשי הקשר הם ה-subscribers שלו. השתמשנו בשליחת הודעות SMS.
השירות הזה עובד יחד עם האתר. כשלוח ESP רוצה לשלוח התראה, הוא שולח בקשת POST ל- API באתר שלנו, והאתר משתמש בחבילת boto3 כדי לפנות לשירות SNS ולשלוח התראה לכל מי שמנוי ללוח הרלוונטי. השימוש ב- Amazon SNS ולא בהתראות SMS מהטלפון עצמו מאפשר למערכת שלנו לפעול בתצורת standalone כך שלא יצטרך להיות חיבור BLE קבוע בין הלוח לבין איזשהו סמארטפון שיבצע שליחות SMS בעצמו.

אתגרים נוספים:

- במהלך הפרויקט ניסינו לאפשר מעבר נתונים בין לוח ה-CC1350 לבין לוח ה-feather באמצעות פרוטוקול I2C. אחרי בדיקות וניסויים רבים ברמה החומרתית והתוכנתית, הבנו שלוח ה-feather לא מסוגל להיות בתפקיד slave במסגרת הפרוטוקול. היות שגם ה-CC1350 לא מסוגל להיות slave, ניסינו דרך נוספת – להשתמש בלוח נוסף (M4 מתוצרת Adafruit) שיהווה slave שמתווך בין שני הלוחות האחרים כמאסטרים. הפתרון עבד אבל יצר בעיות תזמון בשליחת ההודעות (לא ניתן לשלוט מי מהמאסטרים מקבל את ההודעות ששולח ה-slave). לכן היינו צריכים לנטוש את פרוטוקול ה-I2C, והיות שפרוטוקול UART דורש יצירת פרוטוקול מעליו להעברת הודעות, החלטנו לעבור לתצורה הנוכחית של הפרויקט, שבה התקשורת מתבצעת בנפרד מול שני הלוחות, מה שמתיישב עם הרציונל של הפעלת המערכת ע"י הרוקח והמשתמש בנפרד.
- במהלך הפרויקט השתמשנו במספר רב של סביבות פיתוח ועבודה: Python, CCS עם C, Arduino IDE, TI Sensor Controller Studio , Android Studio עם Java, שירותי אמזון (הקמת והרצת שרת

לינוקס על גבי EC2, שימוש בשירות אמזון SNS). כל אחת מילאה תפקיד אחר בפרויקט, והיה מאתגר ללמוד את כולן בצורה שתאפשר להתקדם במהירות ובצורה יעילה.