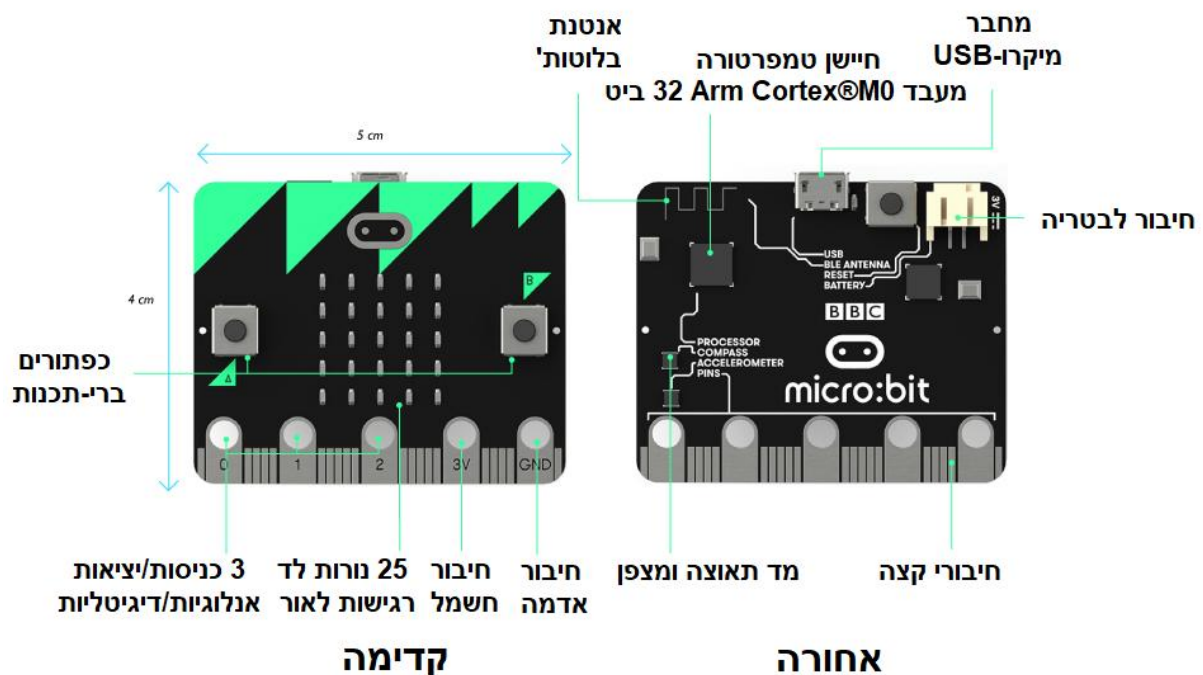


מיקרו:ביט וערכת ההרחבה CROWTAIL 2.0

מיקרו:ביט הוא כרטיס פיתוח, כלומר מחשב זעיר בר-לתכנות שנועד לאפשר הוראה טכנולוגית של פיתוח קוד בצורה קלה ומהנה!



להפעלת הכרטיס אין צורך בידע מוקדם בחשמל או בתכנות. הכרטיס מציע מגוון יכולות בסיסיות אך כדי ליצור פרוייקטים מורכבים יותר ניעזר בערכת ההרחבה הכוללת תוספים שונים כמו מנועים, חיישנים ונורות.

* במסמך זה תמצאו שיעורים ברמות קושי משתנות שמאפשרות לימוד תכנות בצורה מהנה תוך כדי עבודה עם מגוון האפשרויות הגלומות ברכיבי ערכת ההרחבה.

שיעור 1 – היכרות עם כרטיס המיקרוביט

כרטיס המיקרו:ביט הוא בעצם מחשב זעיר. הוא אולי לא נראה כמו מחשבים רגילים שאנו מכירים אך הוא עונה על ההגדרה של מחשב: מכונה שממלאת פקודות של תוכנית. כאשר אנו מדליקים את המחשב האישי שלנו למשל, הוא מריץ סדרה של פקודות המפעילות את מערכת ההפעלה (למשל windows 10). כשנפעיל את כרטיס המיקרו:ביט הוא יריץ את הפקודות שנקבע עבורו. כלומר אנחנו נחליט מה המחשב הזעיר הזה יבצע.

את כרטיס המיקרו:ביט ניתן לתכנת במגוון של סביבות פיתוח. כך למשל ניתן לפתח בסקראץ' ו-MakeCode, סביבות ויזואליות מבוססות בלוקים המאפשרות תכנות פשוט וצבעוני ומתאימות במיוחד למתחילים. כמו כן ניתן לפתח בסביבות מבוססות טקסט דוגמת פייתון וג'אווהסקריפט, המיועדות למתכנתים מנוסים יותר. מתכנתים מתקדמים יכולים לתכנת בשפות עילית דוגמת C/C++.

תהליך העבודה עם המיקרו:ביט כולל את השלבים הבאים:

שלב 1: חיבור

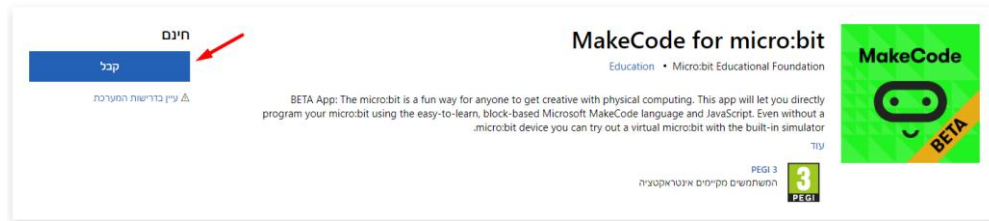
חבר את המיקרוביט למחשב באמצעות כבל מיקרו USB המצורף לו. שימו לב, המיקרוביט יופיע במחשב ככונן חיצוני הנקרא 'MICROBIT':

▼ Devices and drives (2)



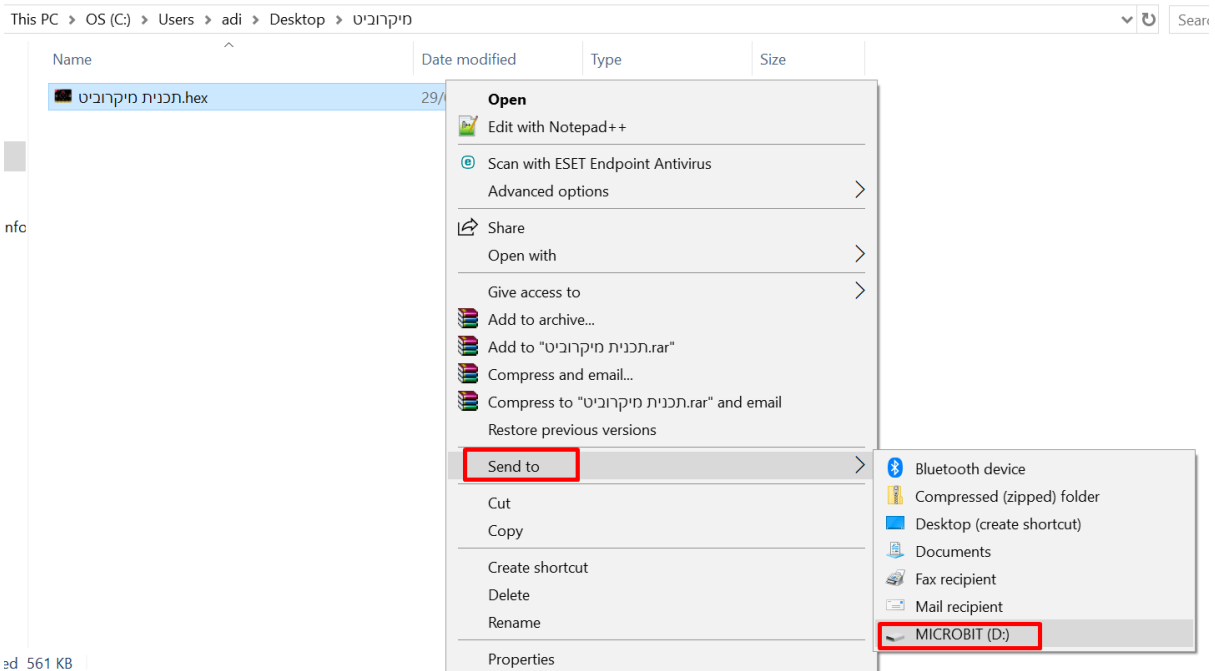
שלב 2: תכנות

לתכנות כרטיס המיקרו:ביט יש לגלוש לסביבת הפיתוח המקוונת MakeCode של מיקרוסופט ב- <https://makecode.microbit.org>. ניתן גם להתקין את הסביבה במחשב (מתאימה לחלונות 10), ולתכנת בה גם ללא חיבור לאינטרנט. [להורדת התקנה לחץ:](#)



שלב 3: הורדה

לחץ על לחצן "הורד" בחלון העורך. בחר שם לתכנית לשמירתה במחשב המקומי שלנו כקובץ בעל סיומת 'hex', שהוא מבנה קומפקטי של התוכנית המיועד לריצה על המיקרו:ביט. ברגע שקובץ ה hex ירד, יש להעתיק אותו אל המיקרו:ביט שלך בדיוק כמו העתקת כל קובץ בשיטת "העתק-הדבק". טיפ: במערכת חלונות ניתן ללחוץ בקליק ימני של העכבר על הקובץ, ולבחור Send to MICROBIT (שלח אל MICROBIT):

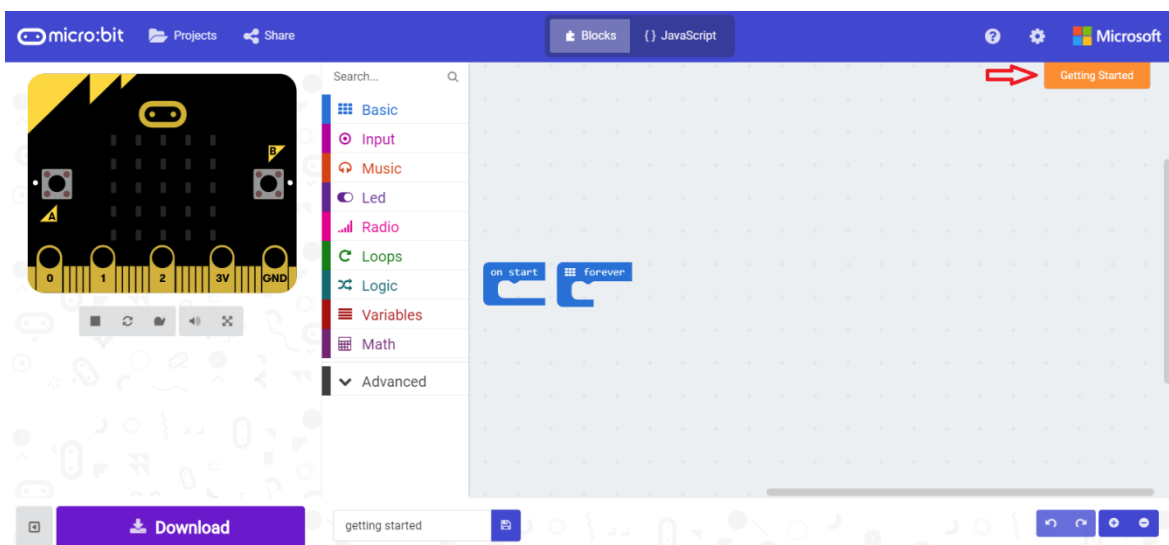


שלב 4: הפעלה

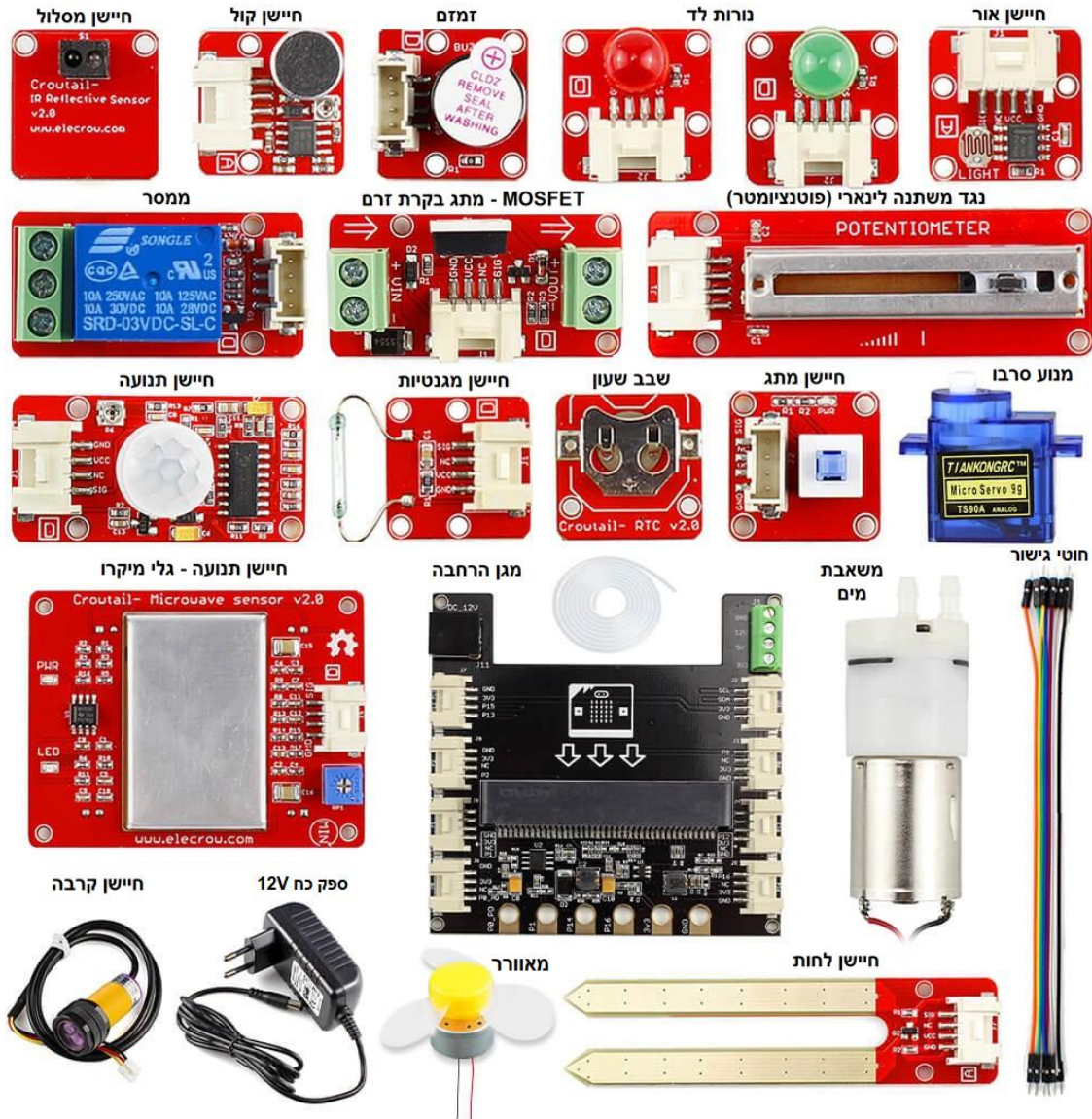
בזמן ההעתקה המיקרוביט יושהה, והנורית הצהובה בצידו האחורי תהבהב עד לסיום ההעתקה של התכנית עם הקוד אל הכרטיס. בסיום ההעתקה הקוד יופעל באופן אוטומטי על הכרטיס והתכנית תתחיל לרוץ.

- * כונן ה- MICROBIT יעלם עם ניתוקו, ויחזור באופן אוטומטי בכל פעם שנחבר אותו, אך התכנית שלנו (קובץ ה hex) לא נשמרת עליו.
- * המיקרו:ביט יכול לקבל רק קבצי hex ולא שום דבר אחר.

תירגול: הפעילו את ההדרכה המובנית הראשונית שבסביבת MakeCode והריצו את המטלות הבסיסיות המוצגות בה:



רכיבי הערכה



מפעילים:

מפעיל הינו רכיב במערכת שאחראי על הזזה ושליטה במנגנון או בתת-מערכת, למשל מנוע האחראי על סיבוב גלגל במערכת. מפעיל דורש אות בקרה ומקור אנרגיה כדי לעבוד. מקור האנרגיה יכול להיות חשמלי, פניאומטי (לחץ אוויר), הידראולי (לחץ נוזל), או כל מקור אנרגיה אחר. כאשר אות הבקרה ("סיגנל") המבוקש מתקבל, המפעיל מגיב בהמרת האנרגיה לאנרגיה מכנית לשם ביצוע תפקידו.

מפעילים בערכה: מנוע סרבו, מנוע DC, נורות לד, זמזם, משאבת מים

חיישנים:

חיישן הוא רכיב המשמש למדידה, בקרה והעברה של מידע למערכות אחרות. החיישן חש תכונה בעולם הפיזי-מוחשי וממיר אותה לאות חשמלי או אלקטרוני שניתן לקריאה על ידי מערכות שונות, המסוגלות לקבל החלטות בהתאם לערכים המתקבלים מן החיישן. חיישנים משמשים במערכות שונות ממגוון תחומים, ונמצאים בכל מקום סביבנו, כך למשל החיישן במעלית מונע סגירה על אדם הנכנס אליה, חיישן בדלתות אוטומטיות מאפשר פתיחתן כשמתקרבים אליהן, חיישן הרברס המותקן באוטו יאפשר התראה בצפצוף כאשר הרכב מתקדם אחורנית אל מכשול, וכן הלאה. ישומים נפוצים של חיישנים נמצאים בכלי רכב, במנועים, בלוחים מלאכותיים, במכשירים רפואיים, בתעשייה וברובוטים.

חיישנים מתחלקים לשתי קבוצות עיקריות בהתאם לטכנולוגיה בה הם פועלים - חיישנים פסיביים (אשר מודדים אנרגיה קיימת, כגון חיישן טמפרטורה), וחיישנים אקטיביים, אשר פולטים אנרגיה ואז קולטים את ההחזר שלה (כגון חיישן מרחק או מכ"ם).

עוד חלוקה נפוצה של חיישנים היא לחיישנים דיגיטליים המסוגלים לזהות שני מצבים בלבד ולהחזיר ערך בינארי של 0 או 1 המעיד על המצב הנוכחי, וחיישנים אנלוגיים המסוגלים לתאר כמות של הערך הנמדד בצורה מספרית.

חיישנים דיגיטליים בערכה: חיישן מתג, חיישן מגנטיות, חיישן תנועה-גלי מיקרו, חיישן תנועה, חיישן מסלול

חיישנים אנלוגיים בערכה: חיישן קול, חיישן לחות, פוטנציומטר, חיישן אור, חיישן קרבה

רכיבי אלקטרוניקה

רכיב אלקטרוני הוא עצם אלקטרוני בסיסי, בעל פונקציה מוגדרת, המשמש ליצירתם של מכשירים אלקטרוניים. רכיב אלקטרוני בנוי בדרך כלל במארז משל עצמו, ולו הדקים מתכתיים (שניים או יותר) המשמשים לחיבורו לרכיבים אחרים. מספר רכיבים אלקטרוניים המחוברים ומתפקדים יחד נקראים מעגל אלקטרוני.

רכיבי אלקטרוניקה נוספים בערכה: ממסר, מתג בקרת זרם MOSFET, שבב שעון

שיעור 3 – עבודה עם מפעילים

חומרים נדרשים:

- כרטיס מיקרו:ביט
- מגן הרחבה CROWTAIL
- מנורת LED עם כבל חיבור

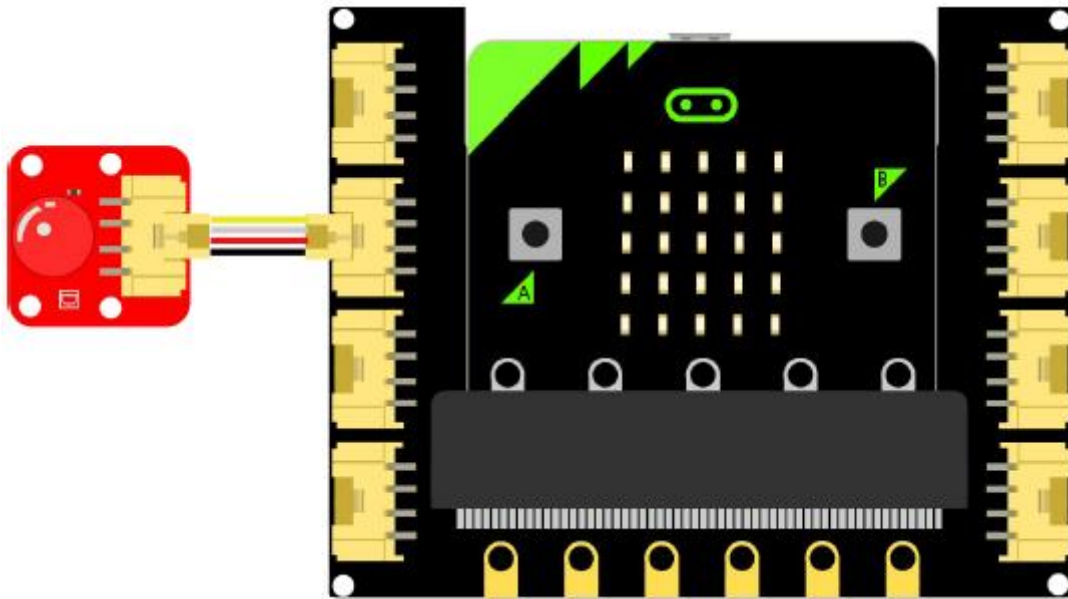
כאמור מפעיל הינו רכיב במערכת שאחראי על הזזה ושליטה במנגנון או בתת-מערכת, למשל מנורה האחראית להדליק נורת לד. מפעיל דורש אות בקרה ומקור אנרגיה כדי לעבוד, כך שהמנורה דורשת אות בקרה כדי לדעת מתי עליה להדליק את הנורה באמצעות סגירת מעגל חשמלי, וחשמל כדי שתוכל לספק אנרגיה חשמלית לנורה באמצעות המעגל שנסגר.

בשיעור זה נלמד כיצד לחבר מנורת לד חיצונית לכרטיס אותה נפעיל בלחיצת כפתור. חשוב להבין שכל תהליך הוספת רכיב חיצוני בפרוייקט מיקרו:ביט כולל את השלבים הבאים:

- 1- הכנסת כרטיס המיקרו:ביט אל מגן ההרחבה
- 2- חיבור פיזי של הרכיב החיצוני אל מגן ההרחבה (באמצעות הכבל הייעודי של הרכיב)
- 3- כתיבת התכנית העושה שימוש ברכיב החיצוני (במסמך זה נדגים את סביבת ה-MakeCode הויזואלית של מייקרו:סופט)
- 4- העתקת התכנית אל הכרטיס המחובר בכבל USB אל המחשב (ר' פרק 1) שתוביל להפעלתה על גבי הכרטיס

נחבר את מנורת הלד למגן ההרחבה באמצעות הכבל שלה, שהינו כבל בעל מחברי JST משני קצותיו. מחברי ה-JST שבקצות הכבל כוללים כניסות ל-4 פינים (מחבר מסוג "נקבה"), ומתחברים בצידם האחד אל מנורת הלד ובצידם השני ליציאה בצדו של מגן ההרחבה. בין החוטים יש חוט שאחראי להעביר את אות הבקרה (SIG-סיגנל), חוט לפלוס (VCC-מתח כניסה) וחוט למינוס (GND-אדמה/הארקה) שתפקידם לאפשר את סגירתו של המעגל החשמלי הנדרש, וחוט הקובע האם יסגר מעגל חשמלי באות בקרה נשלח שערך 1 או כשערך 0 (NC-סגור במצב נורמלי).

בשיעור זה נחבר את הכבל ליציאת "P2" של מגן ההרחבה, ובהתאם לכך נדאג בקוד שלנו להפעיל את המנורה דרך יציאה זו.

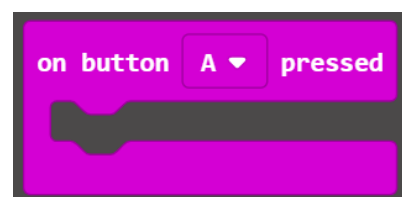


תרגול: כתוב תכנית לזיהוי לחיצה על כפתור A שבכרטיס, ובתגובה להדליק את המנורה למשך חצי שניה ואז לכבותה.

האלגוריתם (שיטה) הנדרש כולל את השלבים הבאים:

- כאשר לוחצים על כפתור A בכרטיס:
 - הדלק את המנורה
 - המתן חצי שניה
 - כבה את המנורה
 - המתן חצי שניה

ראשית נזהה כאשר כפתור A נלחץ: בסביבת הפיתוח MakeCode יש "פקודות אירוע" שתפקידן לזהות כאשר אירוע מסוים קורה ואז להריץ את כל הפקודות שתחתיהן בצורה סדרתית אחת אחרי השניה. פקודת האירוע הרלבנטית בפרוייקט זה הינה:



הפקודה נמצאת במשפחת הפקודות "Input", המכילה את הפקודות המטפלות באמצעי הקלט של כרטיס המיקרו:ביט, כדוגמת כפתורי הלחצנים שלו. הפקודה יודעת לזהות לחיצה על כפתור [A] (ניתן לשנות לכפתור [B]), ומפעילה מיידית עם הלחיצה את כל הפקודות תחתיה.

כעת כדי להדליק את המנורה נשלח לה דרך יציאת "P2" אליה היא מחוברת, את הסיגנל הדיגיטלי "1" המסמן לה להדלק בעוצמה מלאה. בצורה דומה "0" יסמן לה לכבות את האור.

הפקודה לשליחת הסיגנל הדיגיטלי הינה:



הפקודה נמצאת במשפחת "Pins" תחת Advanced (שבתחתית איזור משפחות הפקודות).

ניתן לשנות את P0 לכל אחת מיציאות המגן, ולשנות את [1] באמצעות הזזת פס הגלילה שיופיע תחתיו כשנלחץ על השדה.

בהמשך נראה כיצד ניתן לשלוט בעוצמת ההארה, על ידי שליחת סיגנל אנלוגי המתאר את עוצמת ההארה הנדרשת.

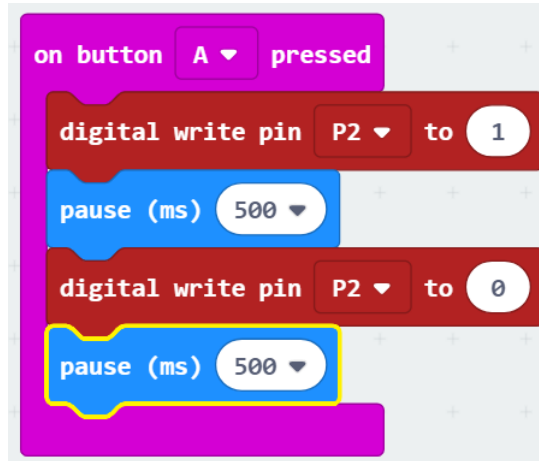
כדי להמתין בזמן ריצת התכנית כדי לאפשר למנורה לבצע את פעולה ההארה במשך חצי שניה לפני שנשלח לה את ההוראה לכבות את האור, נשתמש בפקודת ההמתנה:



הפקודה נמצאת במשפחת הפקודות Basic.

הערך מייצג מילי-שניות (אלפיות השניה), כלומר יש להקליד 500 כדי לייצג חצי שניה.

הקוד הסופי המממש את האלגוריתם נראה כך:



נקודות לדיון:

מה יקרה אם נלחץ לחיצה ארוכה על הכפתור? מה המסקנה לגבי הדרך שבה מיקרו:ביט מגדיר את אירוע לחיצה על הכפתור?

מה יקרה אם נלחץ כמה פעמים במהירות רצוף על הכפתור? מה המסקנה לגבי הדרך שבה אירועים מטופלים במיקרו:ביט?

שימו לב שניתן להקליד לשדה הערך של פקודת שליחת הסיגנל הדיגיטלי כל ערך, על אף שמדובר באות דיגיטלי בעל שני מצבים בלבד בו 0 מייצג מצב אחד ו-1 מייצג את המצב השני. נסו להזין ידנית ערכים אחרים והריצו את התכנית. מה המסקנה לגבי ערכים שונים הנשלחים כסיגנל דיגיטלי?

מה יקרה אם ננתק את המיקרו:ביט מכבל ה-USB? איך ניתן לעבוד עם הכרטיס במנותק מן המחשב? נסו לפתור את הבעיה כדי לעבוד עם הכרטיס לאחר העתקת התכנית אליו. (רמז: ערכת המיקרו:ביט כוללת תא סוללות AAA...)

שיעור 4 – מבוא לחיישנים – חיישן אור (אנלוגי)

חומרים נדרשים:

- כרטיס מיקרו:ביט
- מגן הרחבה CROWTAIL
- חיישן אור וכבל חיבור

מוטיבציה לשימוש בחיישנים

בשיעור הקודם בנינו תכנית שקבעה פעולות קבועות מראש שהתבצעו באופן זהה בכל הפעלה של התכנית ללא קשר למצבו של המיקרו:ביט או לסביבתו. כלומר פעולות העבודה ידועות מראש וקצובות בזמן - הפעלת הנורה לוותה בהקצבת זמן פעולה שנקבע מראש בזמן כתיבת התכנית למשך הדלקתה ומשך כיבויה.

מערכות לדוגמא שעובדות בתכנות קבוע מראש - מערכת השקייה ניתנת לתכנות מראש שתפעיל את הממטרות/טפטפות בשעות קבועות. מערכות רמזורים ניתנות לתכנות מראש ומקציבים זמני פעולה והמתנה קבועים לאור האדום הצהוב והירוק.

יש הרבה סיבות ומוטיבציות להשתמש בחיישנים שיאפשרו לקבל מידע על המערכת וסביבתה כדי לקבל החלטות מושכלות בזמן ריצה כיצד על המערכת לפעול.

- קבלת החלטות בזמן העבודה בהתאם לסביבה – לא תמיד נדע לקבוע מראש כיצד מערכת אמורה להתנהג, אלא בהתאם לסביבתה. כך נרצה למשל שרמזורים בצומת עמוסה יתנו עדיפות לנתיבים העמוסים כדי לשחרר את לחץ התנועה, ושמערכות השקייה יעבדו בהתאם למזג האוויר ולא יעבדו כאשר יורד גשם.
- טיפול בהפרעות לפעולת המערכת – המערכת עלולה לסבול מהפרעות שימנעו ממנה לתפקד. למשל מערכת השקייה שמתמודדת עם ממטרה תקועה או דולפת.
- להינות מיתרונות עבודה אוטומטית ללא מגע יד אדם – כאשר מערכות מבצעות תהליכים באופן אוטומטי (מיכון של תהליכים) – הם הופכים להיות אמינים יותר, מהירים יותר וזולים יותר.

דוגמאות נוספות:

זרוע רובוטית צריכה לדעת מתי לסיים את תהליך הפתיחה או הסגירה שלה ולכן צריכה לבדוק מה זווית הפתיחה שלה בזמן עבודתה.

רובוט הממוקם במבוך ועליו למצוא את היציאה – עליו לדעת מה מיקומו ביחס אליו כדי לזהות את נקודות הכניסה והיציאה ולהריץ אלגוריתם ליציאה מהמבוך (למשל אם כל קירות המבוך מחוברים ביניהם או לגבול המבוך החיצוני, אז חוצה המבוך יגיע ליציאה בסופו של דבר אם יבחר בצד ימין או שמאל וישאיר את ידו על הקיר לאורך כל חציית המבוך)

רובוט המכסח דשא אוטומטית זקוק לחיישנים שיאפשרו לו לנוע בחופשיות באמצעות הדמיית את חוש הראיה כדי לזהות מכשולים בדרכו, כדי לאפשר לעקוף/להתחמק מהם בזמן תנועתו.

דוגמאות למשימות ייחודיות שניתן לממן – ביצוע קציר באופן אוטומטי באמצעות חיישן שמזהה היכן עובר הגבול בין איזור שכבר נקצר לאיזור אותו יש לקצור, וכך הרובוט יכול לנוע באופן אוטונומי (עצמאי) לאורך הגבול ולבצע קציר באיזור הנדרש באופן אוטומטי, ביצוע שינוע (תהליך העמסה-> הסעה-> פריקה) אוטומטי של סחורות יתאפשר באמצעות חיישן שיזהה את פתחי הכניסה בעבור זרועות המלגזה האוטומטית וכדומה.

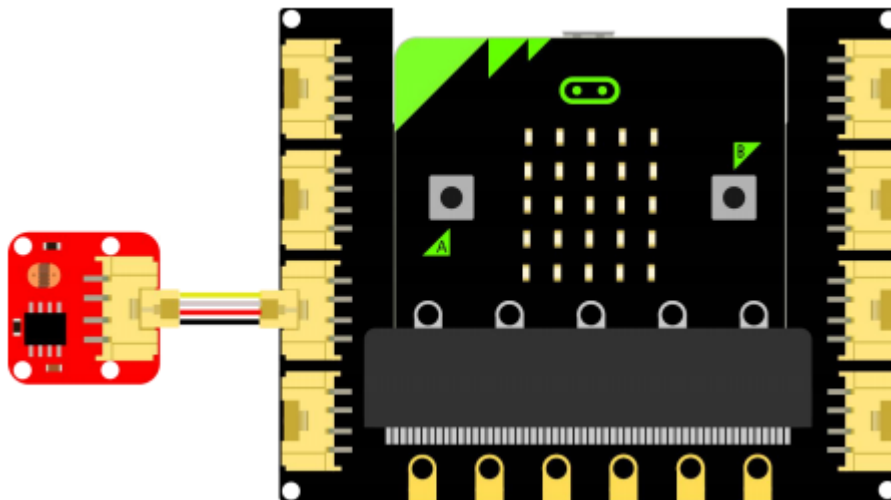
שימוש בחיישנים

רובוט משתמש בחיישנים כפי שאדם משתמש בחושים. החושים שלנו מזרימים כל הזמן מידע למוח שמעבד את המידע ומקבל החלטות בזמן אמת. כך למשל העיניים מזרימות לנו מידע מה נמצא בסביבתנו וכך למשל בזמן הליכה נוכל לעקוף מכשול אותו ראינו, או אם אנחנו רצים למישהו לתת לו חיבוק נדע להאט לפני המפגש כי להמנע מהתנגשות מכאיבה.

בצורה דומה החיישנים מזרימים כל הזמן מידע למיקרו-בקר שלנו (כרטיס המיקרו:ביט) והוא אחראי לקבל החלטות שמבוססות על המידע הזה במסגרת התכנית שהוא מריץ.

בשיעור זה נעבוד עם חיישן האור שבערכה. חיישן האור הינו חיישן אנלוגי, המעביר אל הכרטיס ערך מספרי המייצג את רמת האור הנמדדת בסביבתו של החיישן. החיישן עובד עם רכיב אלקטרוני הנמצא עליו בשם " פוטורזיסטור", שהינו נגד הרגיש לאור (פוטו=אור, רזיסטור=נגד), כך שעוצמת ההתנגדות שלו ברגע המדידה קובעת את הערך שמוחזר אל הכרטיס.

נחבר את חיישן האור האנלוגי שבערכה עם כבל ה-JST שלו אל כניסת "P1" של מגן ההרחבה, ובהתאם לכך נדאג בקוד שלנו להשתמש במידע שיוזרם לכרטיס דרך כניסה זו. בין החוטים יש חוט שאחראי להעביר את אות הבקרה (SIG-סיגנל) עם המידע על ערכו של החיישן, חוט לפלוס (VCC-מתח כניסה) וחוט למינוס (GND-אדמה/הארקה) שתפקידם לאפשר את סגירתו מעגל חשמלי הנדרש לעבודתו של החיישן, וחוט שלא בשימוש (NC-סגור במצב נורמלי).



תרגול: כתוב תכנית להצגת כמות האור הנמדדת בחדר (המידע המתקבל מהחיישן) על גבי הכרטיס.

שימו לב: מכיוון שהתכנית אמורה להציג באופן רציף את הערך הנוכחי של החיישן, יש לבדוק האופן מחזורי שוב ושוב את החיישן ולהציג כל פעם את ערכו המעודכן על גבי הכרטיס. הדרך להריץ קטע קוד בצורה מחזורית היא באמצעות שימוש ב**לולאות**.

לולאות הן "חזרות" על אותה קבוצת פקודות. לולאות מריצות את סדרת הפקודות הכלולה בתוכן שוב ושוב, כך שכשרשימת הפקודות מגיעה אל סופה בלולאה כל התהליך מתחיל שוב מתחילתו.

האלגוריתם (שיטה) הנדרש כולל את השלבים הבאים:

• בלולאה שרצה לעולמים:

○ הצג את ערכו האנלוגי הנוכחי של החיישן על גבי הכרטיס

בסביבת הפיתוח MakeCode יש "פקודות לולאה" שתפקידן להריץ את קטע הקוד שהן עוטפות מספר רב של פעמים.

רוב פקודות הלולאה נמצאות במשפחת "Loops" הירוקה, אך פקודת הלולאה הבסיסית שרצה "לעולמים" נקראת "Forever" ונמצאת ברשימת הפקודות הבסיסית הכחולה של משפחת "Basic":



הערה: שימו לב שניתן בעיקרון "לשבור" לולאת לעולמים באמצעות פקודת "Reset" שבמשפחת "Control", כך שהתכנית תאותחל ותתחיל לרוץ מהתחלה.

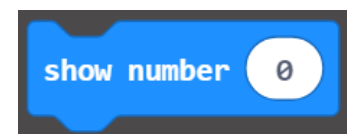
כדי לאחזר את הערך הנוכחי של החיישן נשתמש בפקודת קריאת ערכו של החיישן:



הפקודה נמצאת במשפחת "Pins" תחת Advanced (שבתחתית איזור משפחות הפקודות).

ניתן לשנות את P0 לכל אחת מכניסות המגן.

כדי להציג את הערך על גבי הכרטיס בזמן ריצת התכנית נשתמש בפקודה המאפשרת הצגת מספרים באמצעות הדלקת הורות הLED המתאימות על הכרטיס:

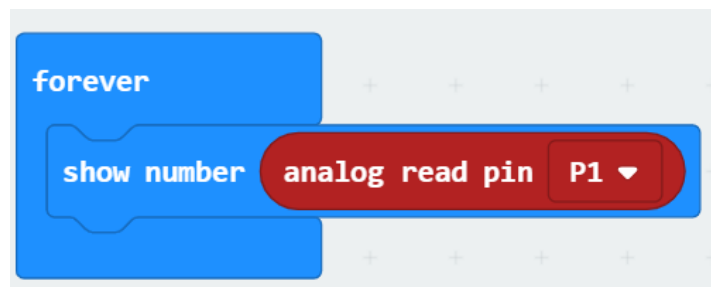


הפקודה נמצאת במשפחת הפקודות Basic.

הערך בפרמטר מייצג את המספר שיוצג על גבי הכרטיס באמצעות הדלקת נורות הלב, וכמו כל פרמטר בסביבת הפיתוח MakeCode ניתן לגרור לתוכו פקודות המחזירות ערכים כדוגמת פקודת אחזור ערכו האנלוגי של חיישן שסקרנו קודם.

שימו לב: כרטיס המיקרו:ביט כולל על גבו מטריצה של 25 נורות לד מובנות המאפשר לו להציג מידע מגוון למשתמש. (הנורות המובנות משמשות גם כחיישני אור כאשר הכרטיס מחזיר ערך של 0-255 המייצג את רמת בהירות האור מסביבו.)

הקוד הסופי המממש את האלגוריתם נראה כך:



נקודות לדין:

מה מוצג כשמחשיכים את סביבת החיישן לעומת חשיפתו לאור ישיר? מה המסקנה לגבי הערכים שמחזיר החיישן ביחס לכמות אור משתנה?

מה הערך המינימלי שהצלחתם להציג על גבי הכרטיס? מה המקסימלי? מה לדעתכם טווח הערכים של החיישן? (טווח החיישן הינו 0-1023)

שיעור 5 – לולאות בקרה וחיישן אור

חומרים נדרשים:

- כרטיס מיקרו:ביט
- מגן הרחבה CROWTAIL
- חיישן אור וכבל חיבור
- מנורת LED וכבל חיבור

בהמשך למוטיבציה לעבודה עם חיישנים, אנו רוצים לתכנן את פעילות המערכת בהתאם לתנאים הסביבתיים שלה.

אם רוצים לעבוד עם חיישן לשם קבלת מידע על התנאים הסביבתיים של המערכת יש לבצע כל הזמן בדיקה רצופה מה ערכיו הנוכחיים כדי שנוכל לפעול ברגע הנדרש לפי הצורך.

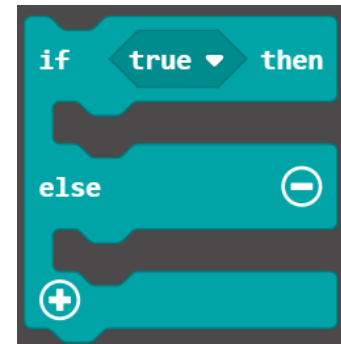
כלומר יש צורך בלולאת בקרה הבודקת בכל חזרה האם תנאי מסויים שנחיל על ערכו הנוכחי של החיישן התקיים כדי לדעת אם נדרשת פעולה, ומבצעת אותה מיידית אם התנאי מתמלא.

מה שהופך לולאה רגילה ללולאת בקרה זה **התנאי** הבודק את מצבו הנוכחי של החיישן, וביצוע הפעולות בהתאם לתוצאת הבדיקה. כלומר הלולאה תהפוך להיות לולאת בקרה כאשר נוסף לגוף הלולאה תנאי שיבדוק את החיישן ויבצע את הפקודות רק כאשר התנאי מתקיים.

שימו לב – בשיעור 3 השתמשנו בפקודת אירוע לחיצה על כפתור. הפקודה למעשה מממשת לולאת בקרה בתוכה הבודקת את מצבו של חיישן המגע (הכפתור שבכרטיס), וכאשר התנאי המוגדר בפרמטרים של הפקודה מתמלא (חיישן A נלחץ ושינה את ערכו הדיגיטלי מ-0 ל-1) היא תריץ את כל הפקודות שבה. ההרצה תתבצע חד-פעמי לכל הפעלה, כלומר בכל פעם שמצבו של החיישן משתנה בלחיצה והתנאי מתקיים ירוצו הפקודות ברצף חד-פעמי.

פקודת בדיקת תנאי

פקודה הבודקת את התנאי שהוצב בה כפרמטר ובוחרת אלו פקודות להריץ בהתאם לתוצאות בדיקת התנאי. במשפחת "Logic" שבצבע טורקיז מרוכזות פקודות התנאים, כדוגמת פקודת IF-THEN-ELSE:

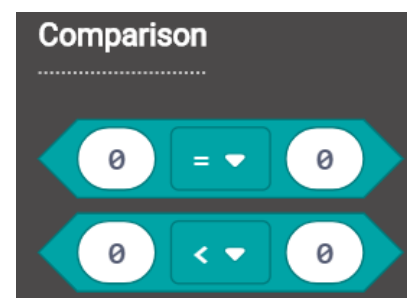


הפקודה תבדוק את התנאי, ואם הוא מתקיים כל הפקודות בחלק הראשון שמתחתיו ירוצו. אחרת (אם התנאי לא מתקיים) ירוצו כל הפקודות שבחלק השני מתחת ל- else.

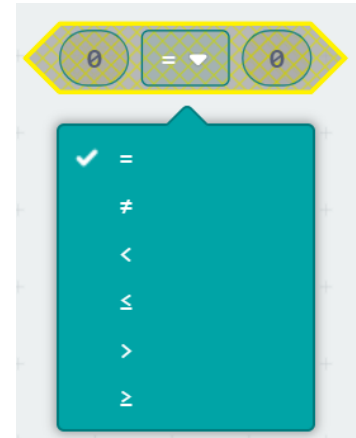
תנאי השוואה

כדי לבדוק את ערכי החיישן בהשוואה לרמת סף מבוקשת, עלינו להשתמש בפקודות המגדירות את תנאי ההשוואה ביניהם.

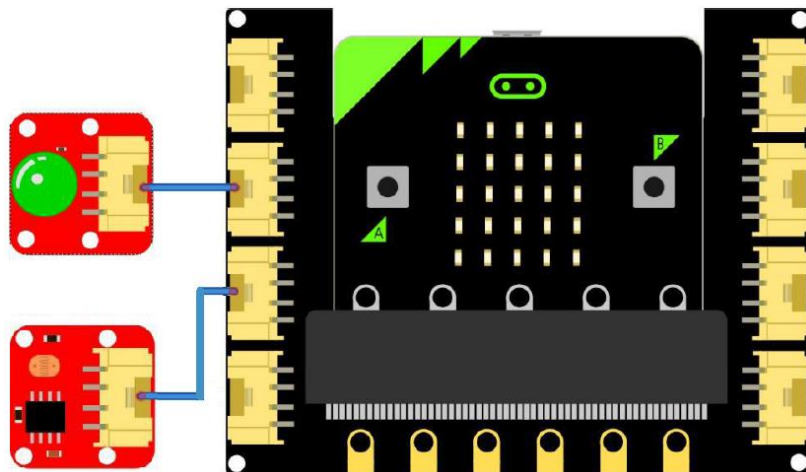
במשפחת "Logic" נוכל לבחור בפקודות המאפשרות הגדרת תנאי ההשוואה בין שני הפרמטרים בצדדי הפקודה.



גם קריטריון ההשוואה שבמרכז הינו פרמטר הניתן להגדרה בבחירת סוג קריטריון ההשוואה מהרשימה:



בשיעור זה נעבוד עם חיישן האור ונורת לד. נחבר את חיישן האור האנלוגי שבערכה עם כבל ה-JST שלו אל חיבור "P1", ואת נורת הלד לחיבור "P2" של מגן ההרחבה:



תרגול: כתוב תכנית שתשתמש בחיישן האור לזיהוי החשכה בסביבת המערכת, ותדליק באופן אוטומטי את מנורת הלד. על התכנית גם לכבות את המנורה כאשר הסביבה חוזרת להיות מוארת.

האלגוריתם (שיטה) הנדרש כולל את השלבים הבאים:

• בלולאה שרצה לעולמים:

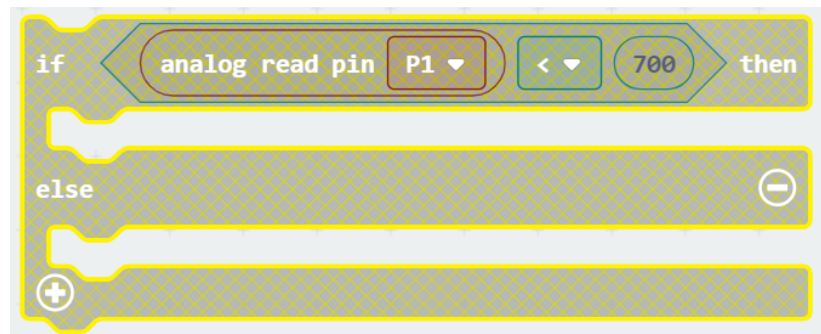
- בדוק את רמת האור בחדר
 - אם חשוך – הדלק את המנורה
 - אם מואר – כבה את המנורה

כדי לממש תנאי הבודק האם ערכו של חיישן האור קטן מתנאי הסף כאשר מוטל עליו צל (למשל בערך חיישן אור של 700 ומטה), נבנה את פקודת תנאי ההשוואה כך:



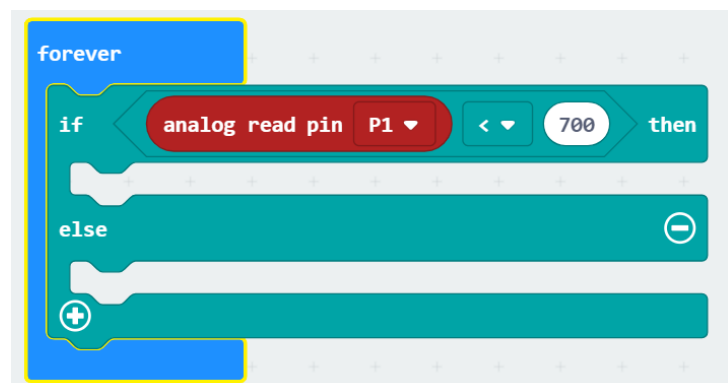
את פקודת קריאת חיישן האור נגרור לפרמטר משמאל ואת הערך 700 נזין ידנית לפרמטר מימין.

את תנאי ההשוואה נכניס לפקודת התנאי:



שימו לב: לולאת בקרה אמורה לרוץ כל הזמן כדי לזהות בזמן ריצה איך לפעול. כלומר חלק בלתי נפרד מלולאת הבקרה היא לולאת לעולמים העוטפת את בדיקת התנאי (אחרת התנאי ייבדק פעם יחידה עם תחילת התכנית ואז היא תסתיים)

לכן יש להכניס את פקודת התנאי ללולאה:



וכעת נוכל לגרור את הפקודות הרצויות לכל אלטרנטיבת התנהגות בפקודת התנאי שלנו. עלינו להוסיף לתכנית את הפקודות להאיר את מנורת הלבד המחוברת לכרטיס כאשר רמת האור יורדת והתנאי מתקיים, ולכבות אותה כשהתנאי אינו מתקיים.

הקוד הסופי המממש את האלגוריתם נראה כך:

```

forever
  if analog read pin P1 < 700 then
    digital write pin P2 to 1
  else
    digital write pin P2 to 0
  
```

נקודות לדיון:

מה קורה כשמעבירים יד מעל החיישן? מה קורה כשמעבירים יד ממש מהר מעל החיישן?

לכל חיישן יש זמני תגובה. גם הוא בודק בלולאה אינסופית מה קורה בסביבתו ושולח את המידע אל הבקר, והזמן בו הוא מבצע את התהליך הזה קובע את זמני התגובה שלו לשינויים בסביבתו. לחיישן האור שלנו יש זמן תגובה של 20-30 מילי-שניות.

שיעור 6 – משתנים וחייוש הקול (אנלוגי)

חומרים נדרשים:

- כרטיס מיקרו:ביט
- מגן הרחבה CROWTAIL
- חיישן קול וכבל חיבור
- מנורת LED וכבל חיבור

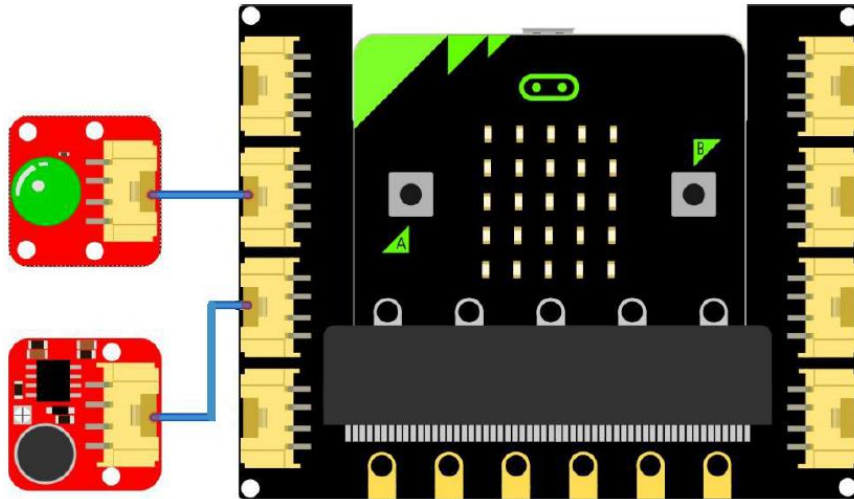
בשיעור זה נלמד על משתנים ועל חיישן הקול האנלוגי.

משתנה הוא חלק זיכרון בתוכנית המכיל נתון כלשהו שיכול להשתנות בזמן ריצת התכנית בהתאם לפקודות הניתנות לו – ניתן לבצע פעולת השמה הקובעת את ערכו החדש, פעולת הוספה או החסרה מערכו (יחסית לערכו האחרון) וכמובן פעולת איחזור ערכו הנוכחי של המשתנה.

המוטיבציה לשימוש במשתנים נעוצה בעובדה שלעיתים התכנית שלנו עובדת עם מידע שאינו ידוע מראש. כך למשל אם תכנתנו משחק וברצוננו לעדכן ניקוד של הצלחות במשחק עבור השחקן, עלינו לשמור את הניקוד המתעדכן במשתנה מכיוון שאיננו יודעים מראש מה יהיה הניקוד של השחקן. כדי לעבוד עם משתנה בתכנית עלינו ליצור אותו (ולקבוע את שמו), לאתחל אותו (לקבוע לו ערך התחלתי), לעדכן אותו במהלך התכנית כנדרש בלוגיקה של המשחק, ולאחזר לפי הצורך את ערכו לשם הצגתו או לשם קבלת החלטות המותנות בערכו.

דוגמא לשימוש במשתנה לעדכון ניקוד בתכנית של משחק כלשהו – ניצור משתנה בשם SCORE, עם תחילת התכנית נקבע את ערכו ל-0, בכל פעם שהמשתמש מבצע פעולה רצויה (הצלחה במשחק שלנו) נעלה את ערכו ב-1, ובסוף התכנית נבדוק את הערך ונציג פרצוף שמח עם הערך גדול מ-10 ופרצוף עצוב אם הערך קטן מ-10.

בשיעור זה נעבוד עם חיישן הקול ונורת לד. נחבר את חיישן האור האנלוגי שבערכה עם כבל ה-JST שלו אל חיבור "P1", ואת נורת הלד לחיבור "P2" של מגן ההרחבה:

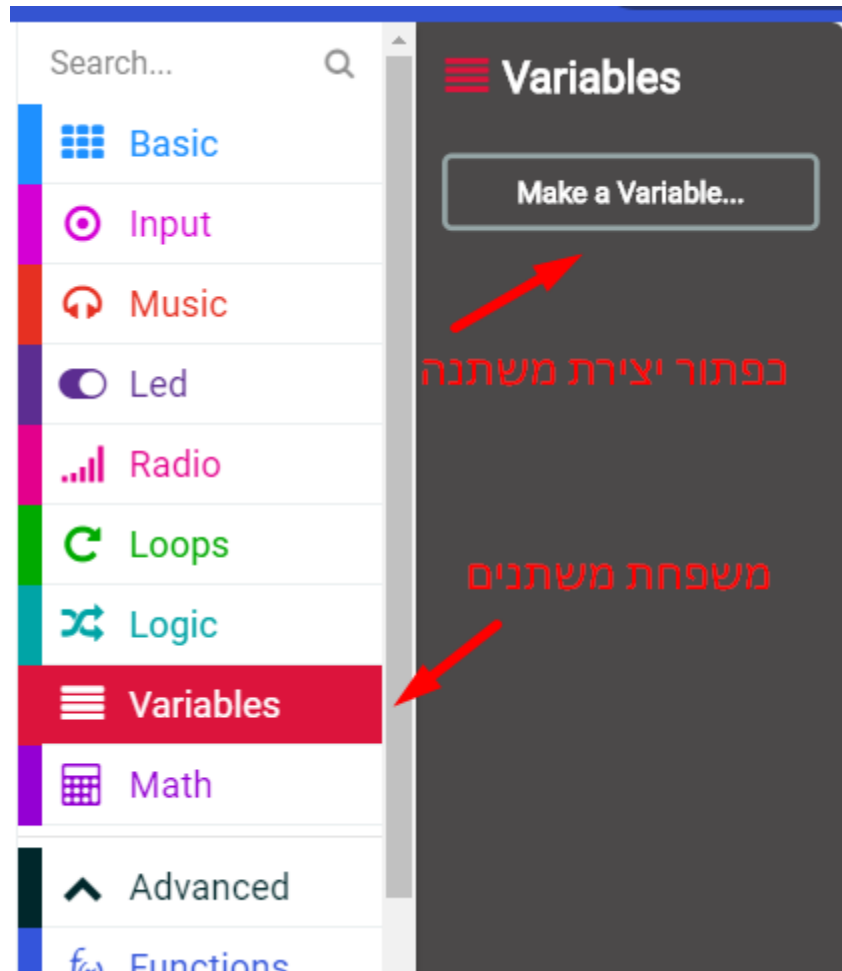


תרגול: כתוב תכנית שתשתמש בחיישן הקול, וכאשר נשמעת מחיאת כף תדליק את הנורה ותציג את רמת הרעש הנוכחית בחדר.

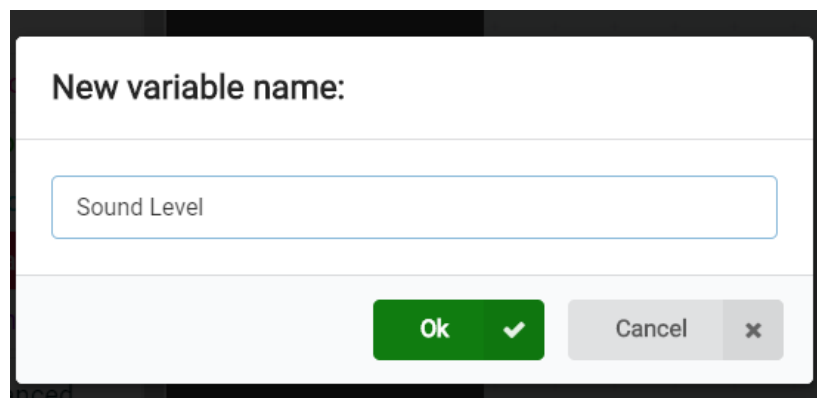
האלגוריתם (שיטה) הנדרש כולל את השלבים הבאים:

- בלולאה שרצה לעולמים:
 - בדוק את רמת הרעש הנוכחית
 - אם נשמעה מחיאת כף – הדלק את המנורה והצג את רמת הרעש
 - אחרת – כבה את המנורה

ראשית ניצור משתנה וניתן לו שם: SOUND LEVEL. נבחר את משפחת Variables (משתנים) ונלחץ על כפתור Make a Variable ליצירת המשתנה החדש.



נקליד את שם המשתנה בחלון הדיאלוג שיפתח ונלחץ על OK:

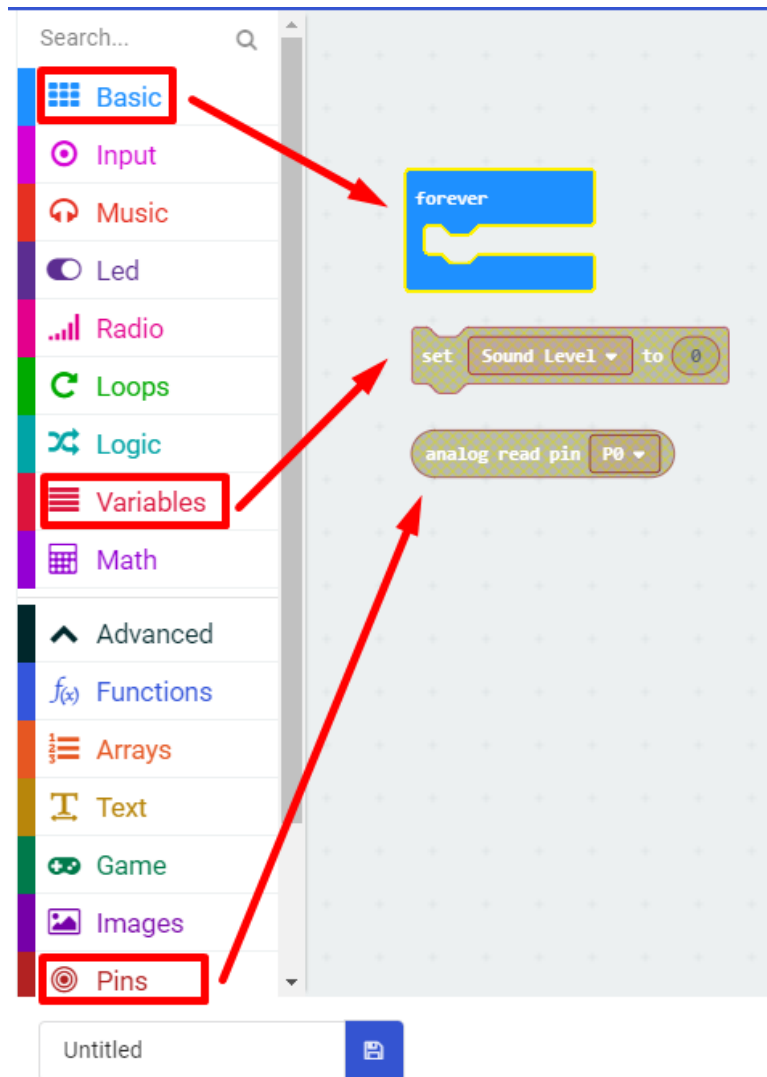


כעת בחלונית המשתנים יופיעו הפקודות המאפשרות עבודה עם המשתנה החדש שיצרנו:

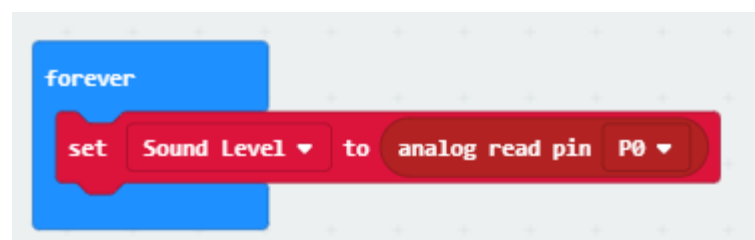
The image shows a screenshot of the 'Variables' window in a Scratch-like environment. The window title is 'Variables' and it contains a 'Make a Variable...' button. Below it, a variable named 'Sound Level' is defined. Two code blocks are attached to the variable: a 'set Sound Level to 0' block and a 'change Sound Level by 1' block. Red arrows point from Hebrew text annotations to these elements:

- An arrow points from the text 'לאיחזור ערך המשתנה הנוכחי' (To avoid the current variable value) to the 'Sound Level' variable name.
- An arrow points from the text 'לקביעת ערך חדש למשתנה' (To set a new value for the variable) to the 'set Sound Level to 0' block.
- An arrow points from the text 'לשינוי ערכו המספרי הנוכחי של המשתנה' (To change the current numerical value of the variable) to the 'change Sound Level by 1' block.

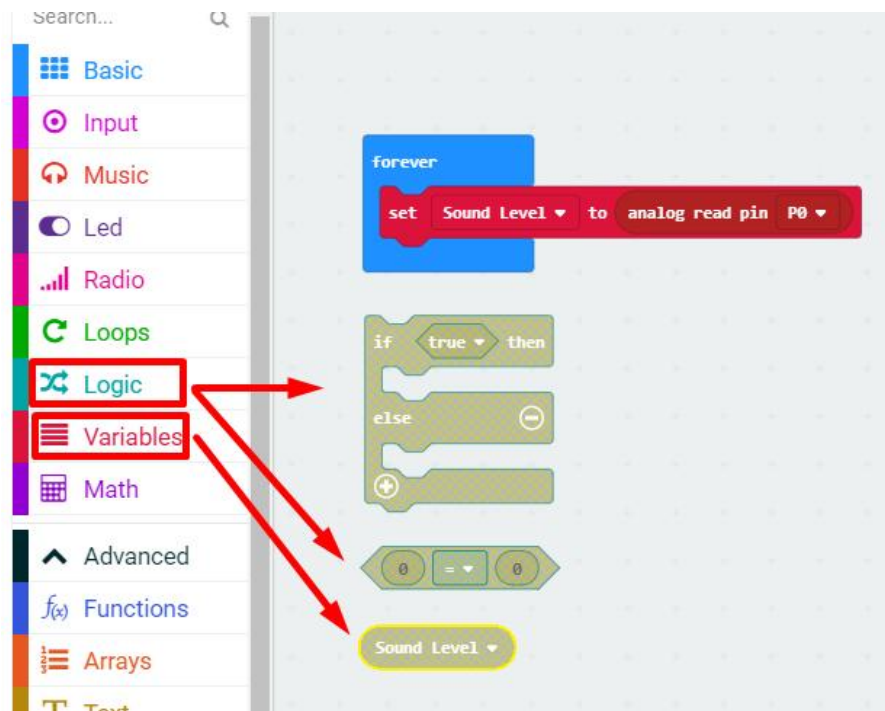
ניצור לולאה אינסופית הבודקת את ערכו האנלוגי של חיישן הקול שחיברנו לכניסה PO, ומעדכנת את ערכו של המשתנה בהתאם. ראשית נבחר את פקודת הלולאה, פקודת קביעת ערך החיישן, ופקודת קריאת החיישן:



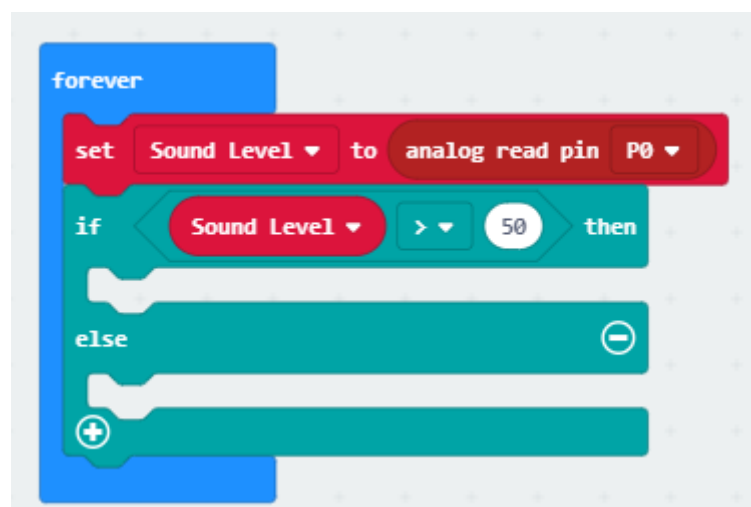
ניצור את הלולאה המבוקשת:



כעת נוסיף ללולאה את התנאי הבודק האם נשמעה מחיאת כף. נבדוק אם ערכו של המשתנה (המכיל בכל רגע נתון את הערך האחרון של חיישן הקול) גדול מרמת הסף לזיהוי צליל מחיאת כף (למשל בערך חיישן קול של 50 ומעלה). נבחר את הפקודות המתאימות:



נזין את ערך תנאי הסף בפקודת ההשוואה (נבחר את אי השוויון "גדול מ-") אל מול ערך המשתנה ונוסיף את התנאי המבוקש לפקודת התנאי וללולאה:



כעת נבחר את הפקודות הרצויות לכל אלטרנטיבת התנהגות בפקודת התנאי שלנו: עלינו להוסיף לתכנית את הפקודות להאיר את מנורת ה led ולהציג את הערך של המשתנה על גבי הכרטיס, כאשר נשמעת מחיאת כף והתנאי מתקיים. כשהתנאי אינו מתקיים עלינו לכבות את הנורה.

פקודת Show Number מספר על גבי הכרטיס נמצאת במשפחת Basic:



אנו לא נציג מספר קבוע, אלא נדאג להכניס כפרמטר את המשתנה שלנו המכיל את הערך הנוכחי של רמת הקול.

הקוד הסופי המממש את האלגוריתם נראה כך:



נקודות לדיון:

מה יקרה אם נחליף את סדר פעולות הדלקת המנורה והצגת המספר על גבי הכרטיס? מה ניתן להסיק מכך לגבי הדרך שבה הכרטיס מריץ את פקודת הצגת המספר? ממה צריך להיזהר בעבודה עם פקודה זו?

בזמן ההמתנה לפקודה לסיים להציג את המספר הלולאה "תקועה" ולכן אנו עלולים לפספס אירועים שונים, למשל מחיאות כף נוספות. יש לקחת זאת בחשבון כשנרצה להציג מידע על הכרטיס.

שיעור 7 – חיישן קרבה (אנלוגי) ונורות הLED של המיקרו:ביט

חומרים נדרשים:

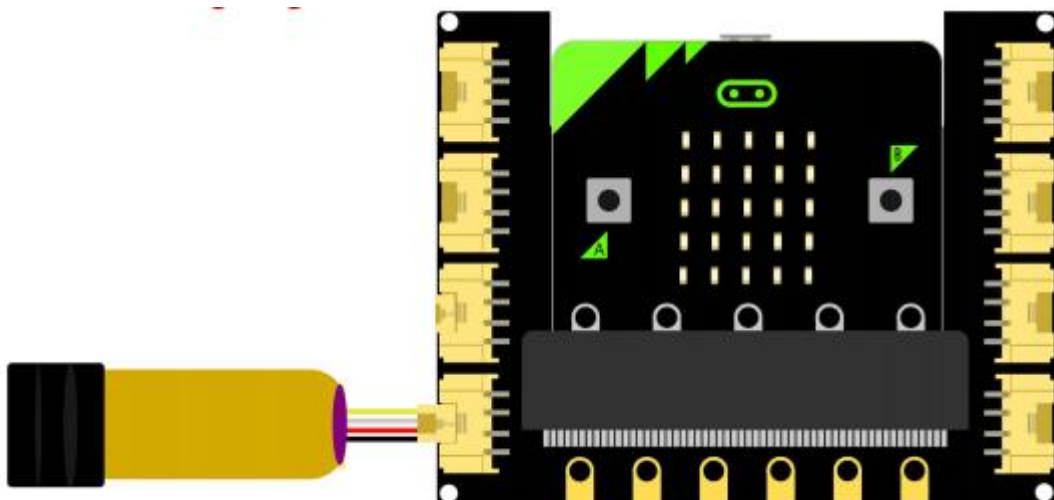
- כרטיס מיקרו:ביט
- מגן הרחבה CROWTAIL
- חיישן קרבה

חיישן הקרבה מאפשר זיהוי עצמים/מכשולים קרובים. טווח אפקטיבי עד 50 ס"מ, תלוי ברגישות שלו שניתן להגדיר באמצעות הפוטנציומטר שלו.

פוטנציומטר - רכיב חשמלי שתפקידו להתנגד למעבר זרם חשמלי דרכו ("נגד משתנה"). ייחודו של הנגד המשתנה בכך שהתנגדותו אינה קבועה וניתנת לכוונון בטווח מסוים. שינוי ההתנגדות מתבצע על ידי סיבוב הבורג הקטן שבתחתית החיישן באמצעות מברג. השינוי יקבע את המרחק האפקטיבי של החיישן ואת רמת הסף של המרחק מן המכשול שמולו שמגדירה אותו כ"קרוב".

שים לב: החיישן ידליק נורת LED קטנה עליו כאשר יזהה שהמכשול קרוב אליו, כך שברגע שנחברו למיקרו:ביט עובד הוא יקבל ממנו חשמל ויהיה ניתן לראות את פעולתו.

נחבר את חיישן הקרבה האנלוגי שבערכה עם כבל ה-JST שלו אל חיבור "P1", ואת נורת הLED לחיבור "P0" של מגן ההרחבה:



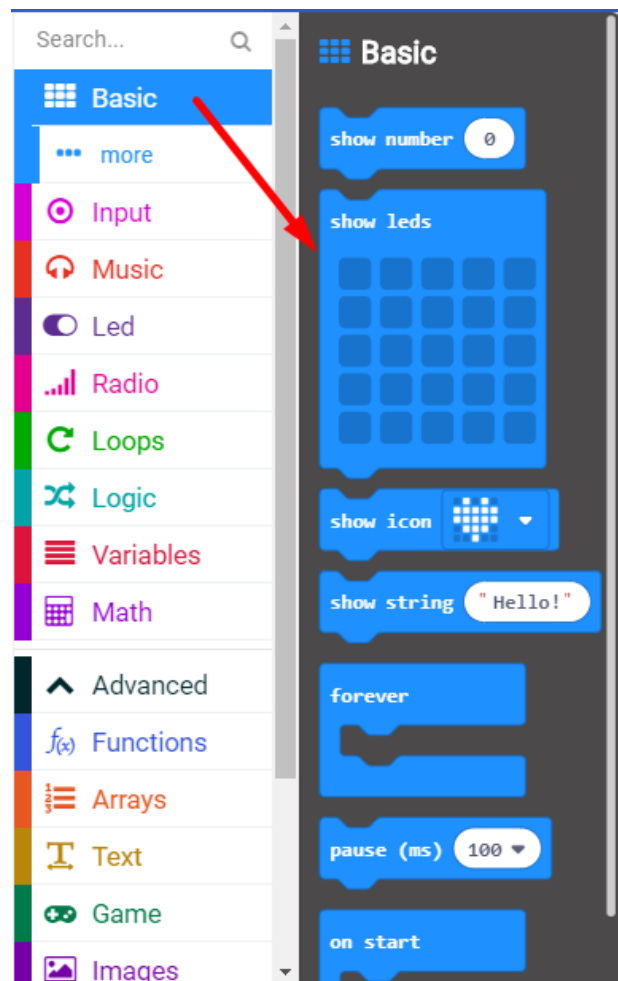
תרגול: כתוב תכנית שתשתמש בחיישן הקרבה לזיהוי מכשולים קרובים, ותשנה באופן אוטומטי את המוצג על גבי נורות הLED של כרטיס המיקרו:ביט.

האלגוריתם (שיטה) הנדרש כולל את השלבים הבאים:

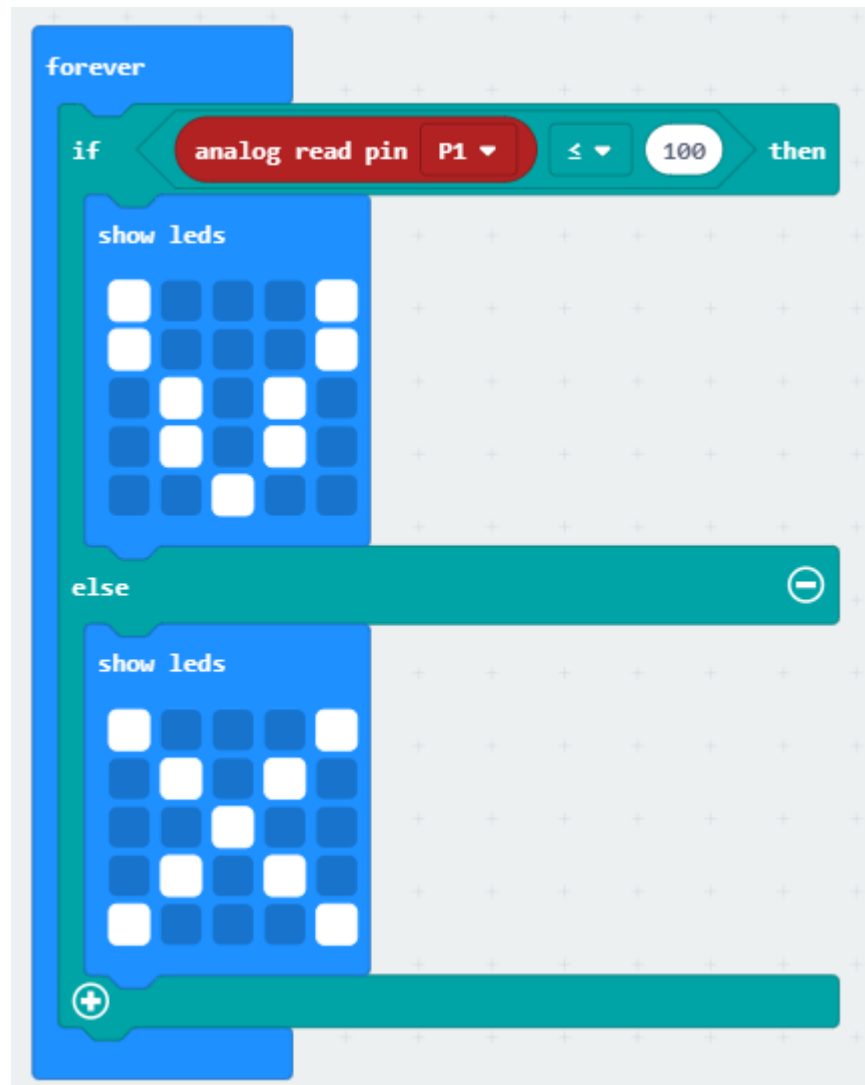
- בלולאה שרצה לעולמים:
 - בדוק האם יש מכשול קרוב
 - אם יש מכשול קרוב – סמן V באמצעות נורות הLED של המיקרו:ביט
 - אחרת – סמן X באמצעות נורות הLED של המיקרו:ביט

ראשית נקבע את ערך הסף של החיישן שממנו ומטה המכשול ייחשב "קרוב". החיישן מחזיר ערך של מאות עד אלפים אך החל ממרחק מסויים החיישן מחזיר ערך נמוך קבוע, כאמור הערכים הספציפיים תלויי רגישות וניתן לכיוונון ע"י הפוטנציומטר של החיישן. ניתן לבחור רמת סף של 100 ומטה כדי להתייחס למכשול כקרוב.

הפקודה להצגת סימונים באמצעות נורות הLED Show Leds נמצאת במשפחת Basic:



הקוד הסופי המממש את האלגוריתם נראה כך:



תרגול מתקדם – ניתן להריץ לולאה אינסופית שתציג את ערכי החיישן על כרטיס הLED (ר' שיעור 4), ולראות אלו ערכים מחזיר החיישן באילו מרחקים מהמכשול. סובבו את הפוטנציומטר עם מברג ושימו לב לשינוי ברמת הסף למרחק שהחל ממנו הLED של החיישן נדלק.

שיעור 8 – זמזם (מפעיל) ולולאה שרצה מספר קבוע מראש של חזרות

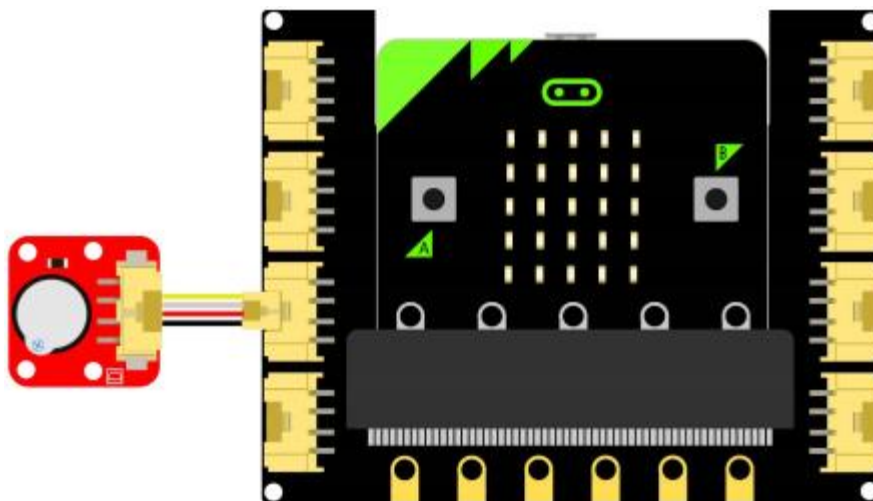
חומרים נדרשים:

- כרטיס מיקרו:ביט
- מגן הרחבה CROWTAIL
- זמזם

כאמור מפעיל הינו רכיב במערכת שאחראי על שליטה במנגנון או בתת-מערכת, למשל זמזם האחראי להשמיע קול. מפעיל דורש אות בקרה ומקור אנרגיה כדי לעבוד, כך שהזמזם דורש אות בקרה כדי לדעת מתי עליו להשמיע את הקול באמצעות סגירת מעגל חשמלי, ואנרגיה הנדרשת להפקת הצליל באמצעות סגירת המעגל החשמלי.

בשיעור זה נלמד כיצד לחבר זמזם חיצוני לכרטיס אותו נפעיל בלחיצת כפתור.

נחבר את הזמזם למגן ההרחבה באמצעות הכבל שלו ליציאת "P1" של מגן ההרחבה, ובהתאם לכך נדאג בקוד שלנו להפעיל את הזמזם דרך יציאה זו:

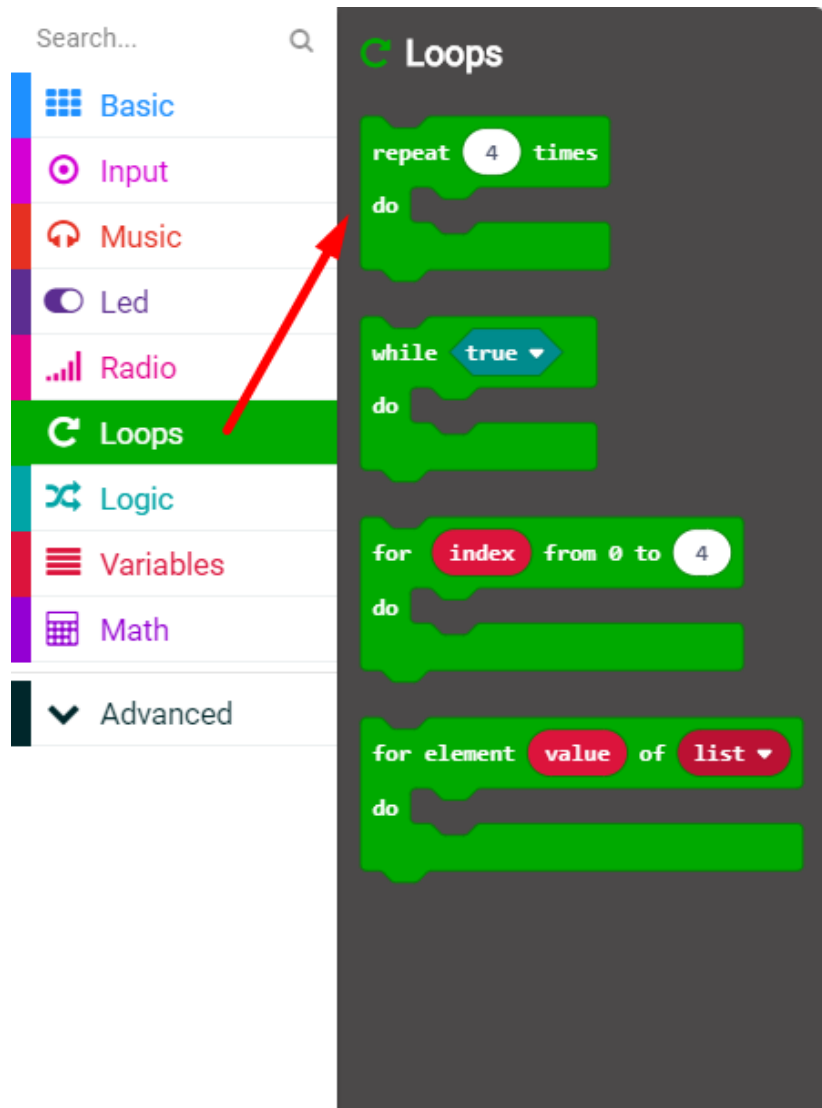


שימו לב: באיור הזמזם מופיע עם מדבקת המגן שלו - יש להסיר אותה כדי שהזמזם יעבוד כראוי

תרגול: כתוב תכנית לזיהוי לחיצה על כפתור A שבכרטיס, ובתגובה תשמיע צפופים בני חצי שניה חוזרים ונשנים בהפסקות בנות חצי שניה - במשך סך של 5 שניות. האלגוריתם (שיטה) הנדרש כולל את השלבים הבאים:

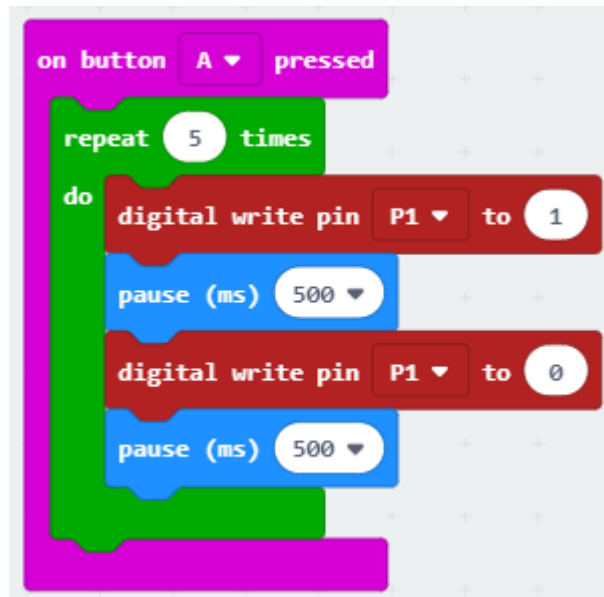
- כאשר לוחצים על כפתור A בכרטיס:
 - חזור 5 פעמים:
 - הפעל את הזמזם
 - המתן חצי שניה
 - כבה את הזמזם
 - המתן חצי שניה

הפקודה להרצת לולאה מספר ידוע מראש של פעמים נמצאת במשפחת Loops:



בפקודה נמצא פרמטר לקביעת מספר החזרות שלה, עם ערך ברירת מחדל של 4. נגרור את הפקודה למשטח העבודה ונשנה את הפרמטר ל-5 כדי שהקוד בלולאה ירוץ 5 פעמים כנדרש.

הקוד הסופי המממש את האלגוריתם נראה כך:



טיפ: אם הזמזם בקושי נשמע נסו לנשוף בחור שלו ולנקוש בתחתיתו כדי לסלק אבק שעלול להפריע לפעולתו.

תרגול נוסף:

בנו מערכת חיישן רברס לרכב, שיודעת לזהות כאשר מתקרבים למכשול (20 ס"מ ומטה למשל) ולהשמיע צפצופי אזהרה כל עוד המערכת קרובה אל המכשול.

למתקדמים: השמיעו צפצופי אזהרה בתדירות משתנה בהתאם למרחק מהמכשול – ככל שמתקרבים תדירות הצפצופים תגבר.

שיעור 9 – מנוע DC (מפעיל) ומתג בקרת זרם MOSFET

חומרים נדרשים:

- כרטיס מיקרו:ביט
- מגן הרחבה CROWTAIL
- מנוע DC (חלק מערכת V1 CROWTAIL)
- מתג בקרת זרם MOSFET (חלק מערכת V1 CROWTAIL)

מהו מנוע DC? כזכור מנוע הנו מפעיל הדורש אות בקרה ומקור אנרגיה (חשמל במקרה שלנו) וממיר את האנרגיה החשמלית לאנרגיה מכנית לשם ביצוע תפקידו. אם נחבר ספק כח חשמלי ישירות למנוע הוא יעבוד ללא הפסקה (בעוצמה התלויה בהספקה המקסימלית של המתח שמקבל), לכן אנו נחבר את המנוע אל מתג בקרת זרם, המשמש מעין "ברז" לאנרגיה החשמלית העוברת דרכו ומאפשר שליטה בהפעלת המנוע.

זרם חשמלי ישר (באנגלית DC=Direct Current), הוא זרימת מיטענים חשמליים דרך תווך מוליך במגמה קבועה, כלומר שאיננה הופכת את כיוונה. זאת בשונה מזרם חילופין, בו מגמת זרימת המיטענים מתהפכת חליפות. זרם ישר הוא תוצאה של 'מתח ישר' (מתח בקוטביות קבועה) הפועל בין קצותיו של תווך מוליך. המונח 'זרם ישר' מתייחס לכל צורה של זרימת מטענים בכיוון אחד (בין אם עוצמת הזרימה קבועה, ובין אם היא משתנה או פועמת). מתח ישר הכרחי ליצירת תנאי עבודה מתאימים עבור רוב סוגי המעגלים האלקטרוניים, וניתן לספק אותו באמצעות ספק כח, מצבר, סוללה, או תאים סולריים.

שימו לב: רשת החשמל הביתית שלנו מספקת חשמל AC. ספק הכח המתחבר לשקע חשמלי אחראי להמיר את זרם החילופין לזרם DC (באמצעות רכיב הנקרא "מיישר זרם") בעבור הרכיבים המתחברים אליו.

[לסרטון רקע על היסטוריית החשמל וזרם ישר מול זרם חילופי](#) (החינוכית: חדשות מהעבר מהדורה עולמית עונה 3 - ניקולה טסלה)

[לסרטון הסבר מעמיק על דרך עבודתו של מנוע DC](#) (לא לשכוח לשנות את הכתוביות לעברית בלחיצה על כפתור ה-"CC" ובחירת שפה Hebrew).

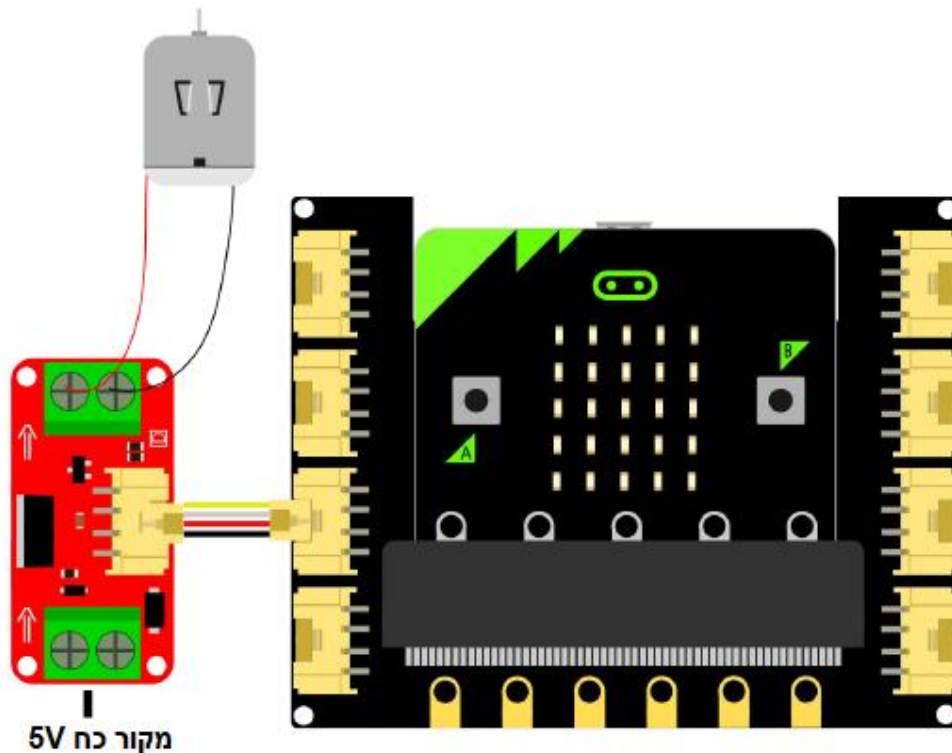
מהו מתג בקרת זרם MOSFET? מדובר ברכיב אלקטרוני בעל ראשי התיבות MOSFET: 'Metal-Oxide-Semiconductor Field-Effect Transistor'. כלומר ראשית מדובר בטרנזיסטור - לענייננו, טרנזיסטור הוא מתג חשמלי שביכולתו לווסת זרימת מטענים חשמליים. הטרנזיסטור הוא אבן הבסיס לבניית שערים לוגיים ומכאן לכל יחידות המחשב (מעבד, זיכרון וכו'). נוח לחשוב על הטרנזיסטור כעל ברז ששולט על זרימת מים בצינור, כאשר המים מסמלים זרם חשמלי. בצינור יש לחץ ולכן מים יזרמו דרכו אלא אם כן שמנו מחסום, למשל ברז שחוסם את המעבר. כלומר, פתיחת הברז תשחרר את החסימה ותאפשר את זרימת המים. הטרנזיסטור מחובר במעגל חשמלי כך שבין שני חיבורי החשמל ישנו מתח חשמלי - אם הוא פתוח זרם חשמלי יזרום דרכו ללא הפרעה (משול ללחץ בצינור). חיבור שלישי נוסף משמש כידית הברז, כך שמתח חיובי על החיבור יגרום לפתיחת הברז ולזרימה חשמלית.

שמו של הטרנזיסטור נגזר מאופן פעולתו - שדה חשמלי פועל על תעלה העשויה מוליך למחצה ומביאה למצב של היפוך של סוג נושאי המיטען שבתעלה, מה שמאפשר לזרם לזרום דרכה. ליבו של טרנזיסטור MOSFET הוא קבל MOS, אשר באמצעותו מושרה השדה החשמלי על המוליך למחצה ובכך נפתח או נסגר המתג. כלומר בסוג זה של טרנזיסטור השדה החשמלי משפיע על התכונות של מוליך למחצה וכך ניתן לשלוט על הזרם החשמלי הזורם דרכו.

איך יראה החיבור הפיסי של המתג? כניסות החשמל של המתג (פלוס ומינוס) יחוברו לספק הכח בהתאמה. יציאות החשמל של המתג יחוברו למנוע. רכיב הבקרה של המתג יחובר אל מגן ההרחבה, כך שנוכל לשלוט מתי המתג יאפשר העברת החשמל מספק הכח המחובר מצידו האחד, אל המנוע המחובר מצידו השני.

נחבר את מתג בקרת הזרם למגן ההרחבה באמצעות הכבל שלו ליציאת "P1" של מגן ההרחבה, ואת כניסות המנוע (חוטי הפלוס האדום והמינוס השחור) נחבר אל יציאות המתג בהתאמה.

* הרכיבו את הפרופלור המצורף לערכה על ציר המנוע (בלחיצה) כדי לייצר מאוורר קטן בו נשתמש בתרגול.



על מנת "לפתוח את הברז" של זרימת החשמל עלינו לספק מתח למתג, כלומר לקבוע את יציאת החשמל של P1 למצב "1". במצב "0" המתג נשאר סגור ולא יעביר חשמל אל המנוע.

תרגול: כתוב תכנית פשוטה המשתמשת בשני כפתורי המיקרו: ביט לשם הפעלה והפסקת פעולתו של מאוורר מבוסס מנוע DC.

האלגוריתם (שיטה) הנדרש כולל את השלבים הבאים:

- כאשר לוחצים על כפתור A בכרטיס:
 - הפעל את המנוע
- כאשר לוחצים על כפתור B בכרטיס:
 - הפסק את המנוע

כזכור בסביבת הפיתוח MakeCode יש "פקודות אירוע" שתפקידן לזהות כאשר אירוע מסוים קורה ואז להריץ את כל הפקודות שתחתייהן בצורה סדרתית אחת אחרי השניה. נשתמש שוב בפקודת האירוע "on button _ pressed" שבמשפחת הפקודות "Input" כדי לזהות לחיצות על כפתורי כרטיס המיקרו: ביט.

הקוד הסופי של התכנית נראה כך:

```
on button A pressed
  digital write pin P1 to 1
```

```
on button B pressed
  digital write pin P1 to 0
```

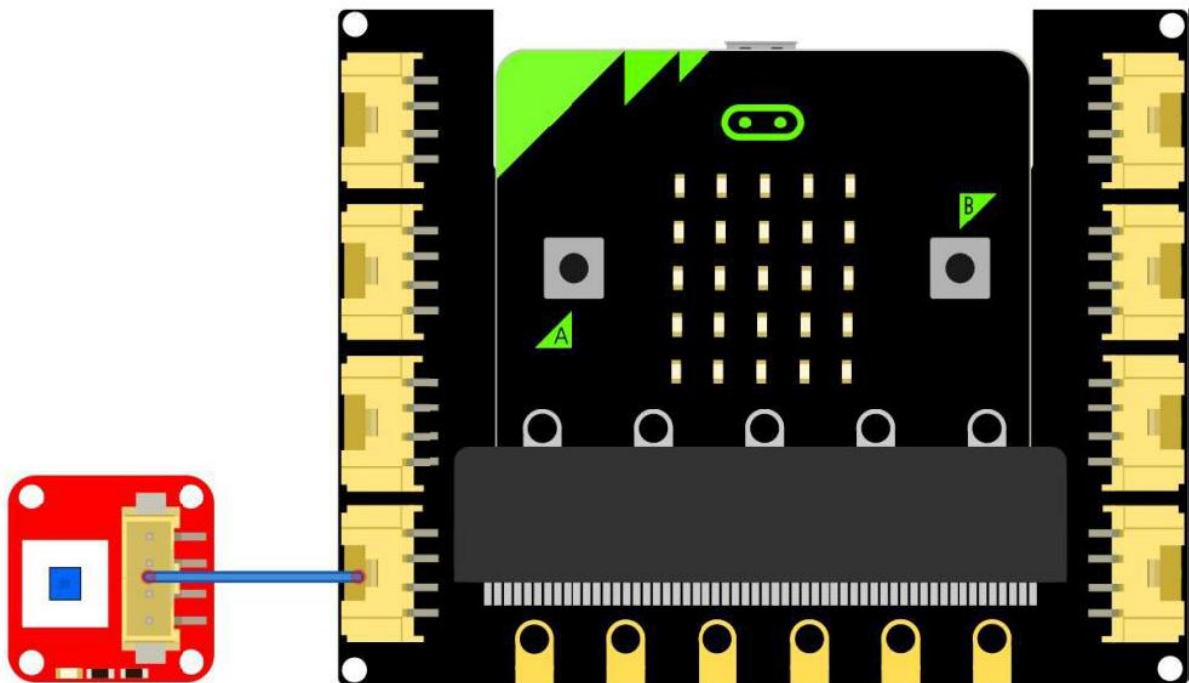
שיעור 10 – חיישן מתג (דיגיטלי) וחיישן הטמפרטורה של מיקרוביט

חומרים נדרשים:

- כרטיס מיקרו:ביט
- מגן הרחבה CROWTAIL
- חיישן מתג (חלק מערכת CROWTAIL V1)

כזכור חיישנים דיגיטליים המסוגלים לזהות שני מצבים בלבד ולהחזיר ערך בינארי של 0 או 1 המעיד על המצב הנוכחי. בשיעור זה נלמד כיצד לחבר חיישן מתג חיצוני לכרטיס, שאת מצבו נשנה בלחיצה עליו.

נחבר את החיישן למגן ההרחבה באמצעות הכבל שלו ליציאת "P0" של מגן ההרחבה:



חיישן המתג יחזיר 0 כאשר אינו לחוץ, ו-1 כאשר הוא לחוץ. שים לב: כל לחיצה בודדת על החיישן משנה את מצבו (מלחוץ ללא לחוץ ולהיפך) **בו הוא יישאר עד ללחיצה הבאה**. כלומר בלחיצה הראשונה ישתנה מצבו ללחוץ, בלחיצה השניה ישתנה שוב ללא-לחוץ, וכן הלאה.

תרגול: כתוב תכנית שמציגה את הטמפרטורה הנוכחית על גבי הכרטיס אם חיישן המתג במצב לחוץ.

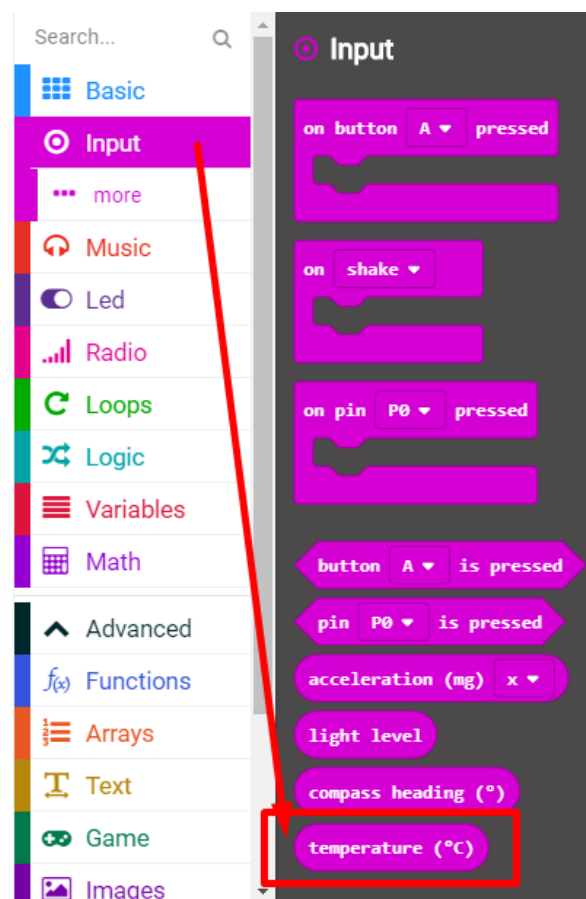
האלגוריתם (שיטה) הנדרש כולל את השלבים הבאים:

• לעולמים

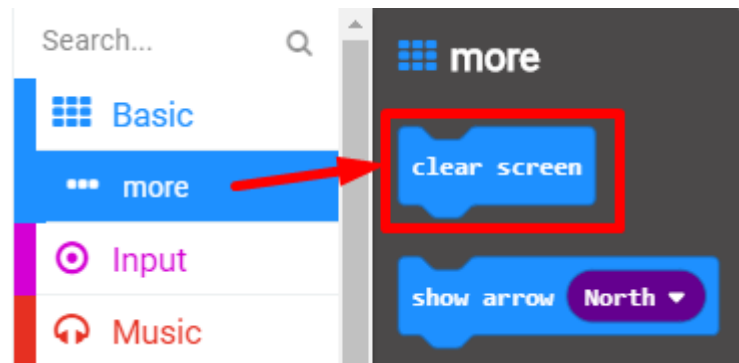
○ בדוק את מצב חיישן המתג:

- אם החיישן לחוץ – הצג את הטמפרטורה הנוכחית
- אחרת – אל תציג דבר

כדי להציג את הטמפרטורה הנוכחית נאחזר את ערכו של חיישן הטמפרטורה הפנימי של כרטיס המיקרו:ביט (ערך מספרי המייצג מעלות צלזיוס). פקודת האחזור temperature של משפחת Input:



כדי לנקות את מסך המיקרו: ביט יש להפעיל את פקודת ה- clear screen שבמשפחת Basic (בעמוד הפקודות השני תחת "more..."):



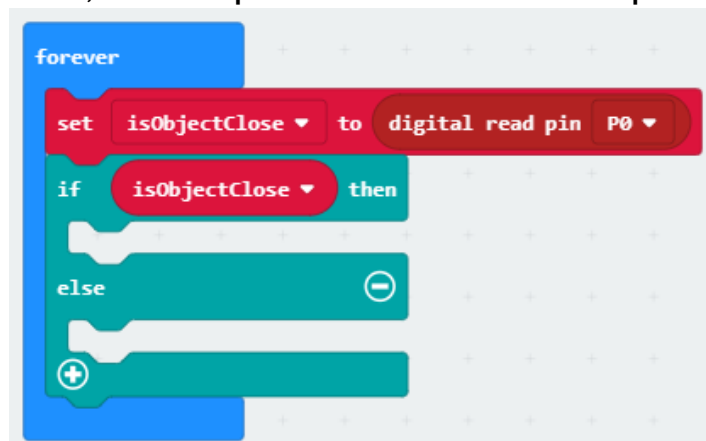
לבדיקת מצב החיישן יש להשתמש בפקודת ה: digital read pin שבמשפחת Pins :



שים לב: לא ניתן לגרור ישירות את פקודת אחזור זו לתוך פקודת התנאי כפרמטר הבדיקה.

ישנן 2 שיטות עיקריות לבדיקת מצבו של חיישן דיגיטלי:

1- נאחסן את ערכו הנוכחי של החיישן במשתנה, ואותו נבדוק בפקודת התנאי:



כאמור הערך 1 משמעו true בבדיקת תנאי (כאשר הינו לחוץ) ו-0 משמעו false בבדיקת תנאי (כאשר החיישן אינו לחוץ).

2- נשתמש בפקודת השוואה של ערך החיישן ל: "1" (או "0") ואת פקודת השוואה נבדוק בפקודת התנאי.

```
forever
  if digital read pin P0 = 1 then
  else
```

בתרגול זה נשתמש בשיטה השנייה הבודקת את ערכו של החיישן בתנאי השוואה.

הקוד הסופי נראה כך:

```
forever
  if digital read pin P0 = 1 then
    show number temperature (°C)
  else
    clear screen
```

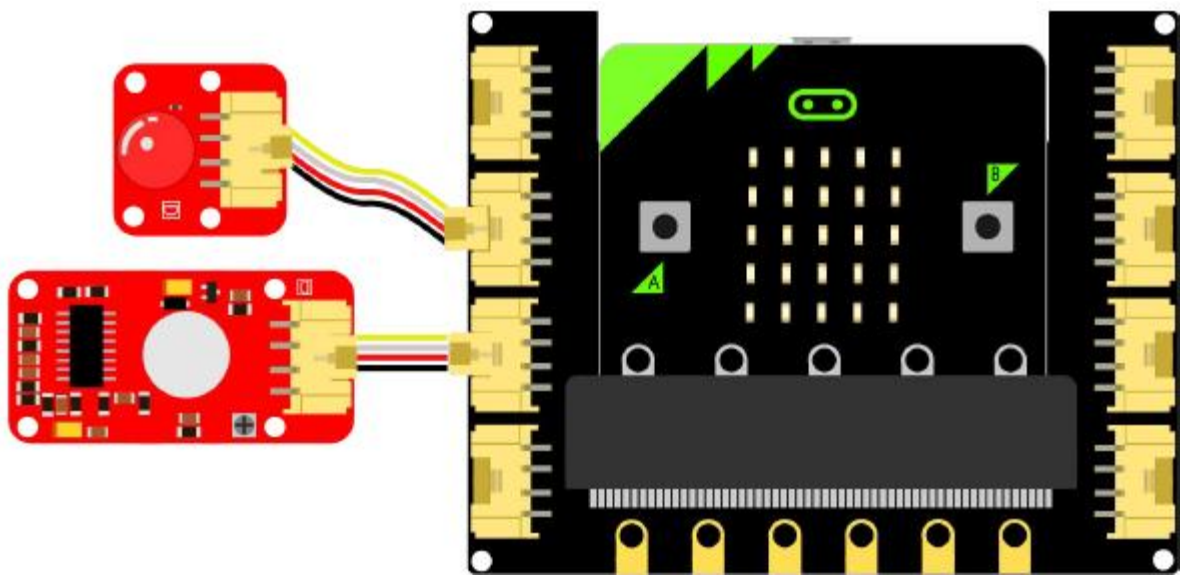
שיעור 11 – חיישן תנועה (דיגיטלי)

חומרים נדרשים:

- כרטיס מיקרו:ביט
- מגן הרחבה CROWTAIL
- חיישן תנועה (חלק מערכת CROWTAIL V1) וכבל חיבור
- מנורת LED וכבל חיבור

כזכור חיישנים דיגיטליים המסוגלים לזהות שני מצבים בלבד ולהחזיר ערך בינארי של 0 או 1 המעיד על המצב הנוכחי. בשיעור זה נלמד כיצד לחבר נורת לד וחיישן תנועה דיגיטלי לכרטיס, על מנת שלכשתזוהה תנועה לידו תדלק נורת הלד.

נחבר את החיישן למגן ההרחבה באמצעות הכבל שלו לכניסת "P1" של מגן ההרחבה, ואת הנורה נחבר ליציאת "P2" של המגן:



*** שימו לב לחבר את החיישן לכניסת P1 בעלת המתח המוגבה (5V) ולא לכניסת P0 שלה אין מספיק כח כדי לאפשר לחיישן לעבוד כשורה**

חיישן התנועה מחזיר 0 כאשר אינו מזהה כל תנועה במרחב סביבו, ו-1 כאשר הוא מזהה את התנועה. טווח הזיהוי תלוי ברגישות שלו שניתן להגדיר באמצעות הפוטנציומטר שלו. כזכור פוטנציומטר הנו רכיב חשמלי שתפקידו להתנגד למעבר זרם חשמלי דרכו ("נגד משתנה"), ושינוי ההתנגדות הקובע את הטווח האפקטיבי של החיישן מתבצע על ידי סיבוב הבורג הקטן שעל גבי החיישן באמצעות מברג.

תרגול: בנה מערכת התראה שתדליק את נורת הLED אם נקלטה תנועה במרחב לידה.

האלגוריתם (שיטה) הנדרש לתכנית לביצוע המטלה כולל את השלבים הבאים:

• לעולמים

○ בדוק את מצב חיישן התנועה:

- אם החיישן מזהה תנועה – הדלק את נורת הLED
- אחרת – כבה את נורת הLED

על מנת לקבל את מצבו הנוכחי חיישן התנועה נשתמש במשתנה שנקרא לו Motion, ובכל איטרציה של לולאת העולמים שלנו נעדכן את ערכו בהתאם לסטטוס הנוכחי של חיישן התנועה הדיגיטלי. (המשתנה יקבל ערך 0/1)

לאחר מכן נבצע בדיקת if-then-else שתממש את הבקרה בלולאה, ובהתאם למצבו הנוכחי של החיישן תחליט אם להדליק או לכבות את הנורה. כזכור ערך 1 משמעו אמת (True) וערך 0 משמעו שקר (False), כך שניתן להכניס את המשתנה עצמו כבדיקה המתבצעת להחליט מתי להדליק את הנורה.

דוגמת קוד סופי אפשרי לפתרון:

```
forever
  set Motion to digital read pin P1
  if Motion then
    digital write pin P2 to 1
  else
    digital write pin P2 to 0
```

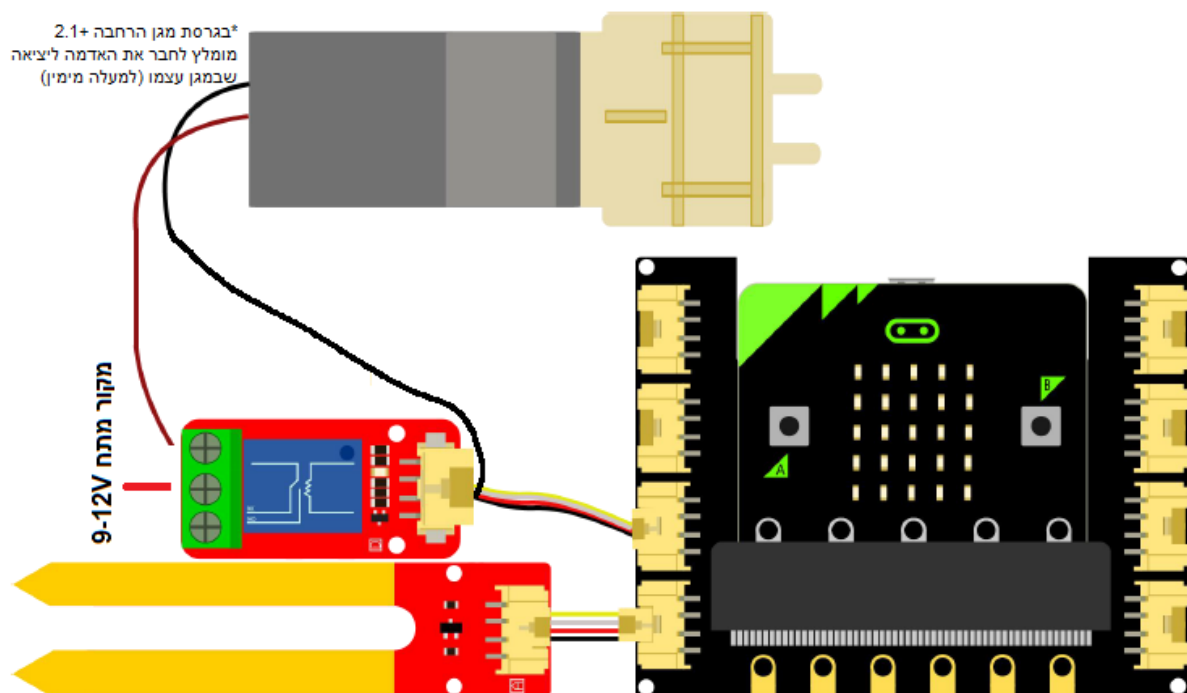
שיעור 12-13 – חיישן לחות (אנלוגי) + ממסר + משאבת מים

בשיעור כפול זה נבצע פרוייקט בניית מערכת השקייה אוטומטית המשתמש בחיישן לחות כדי לזהות מתי נדרשת השקייה, במשאבת מים על מנת להזרים את המים ממקור המים אל העציץ, ובממסר כדי לספק למשאבת המים חשמל במתח גבוה הדרוש לה לשם ביצוע עבודתה.

חומרים נדרשים:

- כרטיס מיקרו:ביט
- מגן הרחבה CROWTAIL
- חיישן לחות (חלק מערכת CROWTAIL V1)
- ממסר (חלק מערכת CROWTAIL V1)
- משאבת מים (חלק מערכת CROWTAIL V1)
- עציץ

להלן תרשים חיבור וחיווט רכיבי המערכת:



* לחיבורי כניסת ויציאת המים מהמשאבה נחבר 2 פיסות שנגזור מצינור הפלסטיק השקוף הארוך המגיע בערכה.

** אספקת החשמל לממסר במתח של 9-12 וולט תתבצע בחיבור מקור כח חיצוני מתאים (בטריה למשל) או בחיבור ליציאת ה-12 וולט המובנית במגן ההרחבה בדגם 2.1 ומעלה (בתנאי שחיברנו למגן את ספק הכח 12 וולט המגיע בערכה המתאימה).

ראשית נבדוק אילו ערכים מחזיר חיישן הלחות כדי לקבוע מהי רמת הסף הנדרשת להשקייה. חיישן הלחות הנו חיישן אנלוגי אותו נועצים באדמה, ולכשנסגר מעגל חשמלי בין 2 "זרועות" החיישן הוא מחזיר ערך המייצג עד כמה האדמה בה הוא נעוץ לחה. כעקרון הרמות הצפויות הן 0-300 לאדמה יבשה, 300-500 לאדמה לחה, ו-500-750 למים. ראשית עלינו לבדוק מהי רמת הסף הנדרשת להשקייה עבור אדמת העציץ הספציפית שלנו – סוגי אדמה שונים סופחים מיים בצורה שונה, וסוגי צמחים שונים זקוקים לרמת לחות שונה בהשקייה.

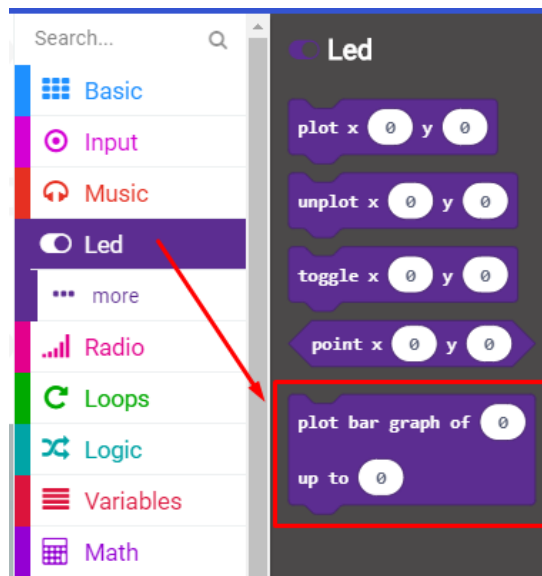
תרגול: בנה תכנית שתציג את רמת הלחות של האדמה בצורה גרפית ובצורה מספרית.

האלגוריתם (שיטה) הנדרש לתכנית לביצוע המטלה כולל את השלבים הבאים:

• לעולמים

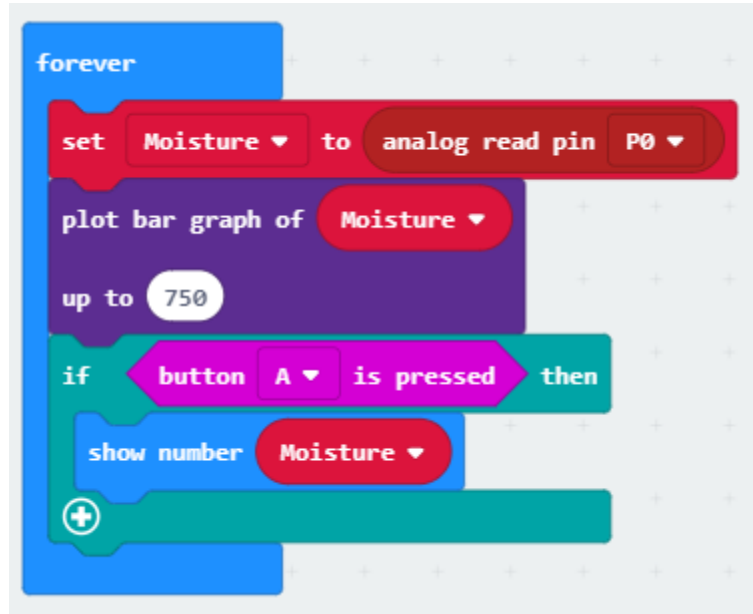
- הצג בצורה גרפית את ערך חיישן הלחות
- בדוק את מצב כפתור A במיקרוביט:
- אם הכפתור לחוץ – רשום במסך המיקרוביט את ערך החיישן

על מנת להציג ערך בצורה גרפית ניתן להשתמש בפקודה "plot Bar Graph", שלוקחת ערך משתנה, ובהתאם ליחס שלו אל ערך המקסימום הקבוע מדליקה מספר נורות לד על הבקר. הפקודה נמצאת במשפחת Led:



כאמור הפרמטר הראשון יהיה הערך של חיישן הלחות, והפרמטר השני יהיה ערך המקסימום אליו יגיעה המשתנה (במקרה שלנו = 750).

קוד סופי אפשרי לפתרון יראה כך:



שימו לב – גם אם תשימו את החיישן בין 2 כפות ידיים הסוגרות עליו תקבלו קריאת לחות...

כדי לבדוק מה רמת הלחות המבוקשת לעציץ שלנו, נמלא כוס קטנה באדמת עציץ יבשה לביצוע הבדיקה. נבדוק מהי הקריאה בחיישן לאדמה היבשה, ונשקה אותה תוך בדיקת קריאת החיישן כדי לראות כיצד ההשקיה משפיעה על הקריאות. רשמו לעצמכם מה הייתה הקריאה באדמה יבשה, ומהי הקריאה של מידת הלחות המבוקשת לאחר שהשקנו כנדרש.

כעת ניתן להשתמש במשאבת המים כדי להזרים מים לעציץ כאשר רמת הלחות נמוכה מרמת הסף שקבענו. ראשית נוודא שמשאבת המים עובדת כנדרש. כאמור משאבת המים זקוקה לרמה גבוהה של מתח כדי לעבוד כנדרש, וכדי להעביר לה מתח גבוה נשתמש ברכיב הממסר שבערכה.

ראשית, אתם עשויים לשאול את עצמכם מה בין הממסר לטרנזיסטור MOSFET שכבר למדנו. לשניהם מטרה משותפת – לשלוט במעבר של זרם חשמלי אחד באמצעות רכיב המופעל בזרם אחר. אך בניגוד לטרנזיסטור הממסר מבודד את המעגלים השונים ומאפשר עבודה בטוחה עם המיקרו-בקר הרגיש (עובד ברמת מתח נמוכה) כדי לטפל במתחים וזרמים גבוהים. הממסר במודול שלנו (הקוביה הכחולה שעל הכרטיס) מיועד לעד 24 וולט של זרם ישיר (אם כי מומלץ תמיד לחבר פחות מהרמה המקסימלית המוצהרת כדי לא להסתכן בקצר).

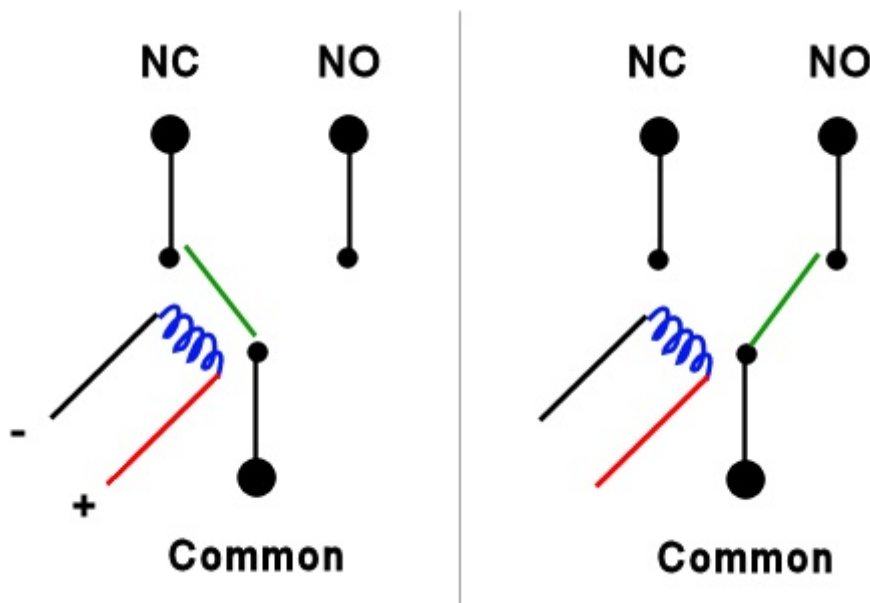
שימו לב גם למתח ההפעלה של הממסר - היחידה מופעלת במתח נמוך יחסית שמספק מגן ההרחבה ($\sim 3V$) כך שהשליטה על העברת הזרם החשמלי הגבוה מתבצעת באמצעות זרם אחר נמוך שמסוגל לספק המיקרו-בקר.

יש 2 אפשרויות חיווט לרכיב, תלוי האם אנו רוצים שהסיגנל מהמיקרוביט יפעיל את הרכיב (Normally Close) או יכבה את הרכיב (Normally Open). מכיוון שאנו מעוניינים להפעיל את המשאבה נבחר בקונפיגורציית NC המודגמת להלן:



כאשר המיקרוביט שולח סיגנל לממסר, הממסר מעביר את הזרם שמגיע מרגל ה-Common החוצה דרך רגל ה-Normally Close.

כיצד הממסר עובד? סוג הממסר שלנו הוא אלקטרו-מכני, והשרטוט הסכמטי הבא מציג בכלליות את מה שקורה בתוך הממסר (שימו לב – בממסר עצמו, לא במודול השלם):



שרטוט סכמטי של מה שקורה בממסר. מימין מצב רגיל, משמאל מופעל (באמצעות הסיגנל).

המעגל השולט הוא סליל (בכחול): כאשר מעבירים דרכו זרם בעוצמה נמוכה מהמיקרוביט, נוצר שדה מגנטי שמושך לשונית מתכת (בירוק). הלשונית הזו סוגרת מעגל בין רגל הכניסה Common לרגל היציאה NC. כשאין זרם, הלשונית חוזרת למקומה המקורי וסוגרת את המעגל בין הכניסה לרגל היציאה השניה (NO). בכל מקרה, כאשר הלשונית יוצרת מגע פיזי עם יציאה כלשהי נשמע קליק חד ואופייני.

תרגול: בנה תכנית שתפעיל ותפסיק את עבודת המשאבה לסירוגין כל 2 שניות. האלגוריתם (שיטה) הנדרש לתכנית לביצוע המטלה כולל את השלבים הבאים:

• לעולמים

- הפעל את המשאבה
- המתן 2 שניות
- הפסק את המשאבה
- המתן 2 שניות

כדי להפעיל ולהפסיק את עבודת המשאבה המחוברת לממסר, נשלח לו סיגנל דיגיטלי עם השהיה כדי לשלוט בפעולתו (ערך 1 = הפעלה, ערך 0 = הפסקה).

קוד סופי אפשרי לפתרון ייראה כך:



תרגול סופי:

תכנת את המערכת להתחיל השקיייה כאשר האדמה יבשה, ולהפסיק את ההשקיייה כאשר האדמה רוויה במים.

שים לב: המשאבה עלולה להתיז בעוצמה מים, מומלץ לבצע ניסויים בתוך כוור למניעת נזקים!