עבודה עם טיימר – מד מהירות תגובה

ציוד נדרש:

- ערכת פיתוח

מהלך הניסוי

1. בניסוי זה נעשה שימוש בטיימר המהווה חלק מתוכנת הבקר. נשתמש בו בהתחלה להדלקה וכיבוי כל פקר זמן קבוע של לד כחול הממוקם על ערכת הפיתוח יחד עם הבקר. בשונה מהניסויים הקודמים, לא נשתמש בהשהיות הדורשות מהבקר לעסוק בספירת הזמן, אלא בעזרת הטיימר נפנה את הבקר לביצוע פעולות אחרות.
2. נפעיל את תוכנת ה Visual studio ונפתח בה פרויקט חדש לעבודה עם הבקר שלנו.
3. נגדיר את הלד אותו נרצה להדליק ולכבות כל פרק זמן מסוים. שימו לב שיש לעשות זאת **מחוץ** לפונקציה הראשית ()Main בכדי שיהיה ניתן לגשת אליו גם מפונקציות אחרות.

```
static OutputPort led = new OutputPort(NoaUpBasic.blue_Led, false);
```

4. כעת נגדיר את הטיימר עצמו בתוך פונקציית ה ()Main.

```
Timer t = new Timer(func, null, 0, 1000);
```

כאשר כאן:

- Func – שם הפונקציה אותה יבצע הבקר בסיום הזמן שיוקצב לו.
- Null – אובייקט מצב (לא בשימוש)
- 0 – הזמן בmSec עד הפסיקה הראשונה של הטיימר
- 1000 – הזמן ב mSec בין הפסיקה האחת של הטיימר לבאה אחריה.

5. לסיום ה ()Main נוסיף המתנה אינסופית.

```
Thread.Sleep(Timeout.Infinite);
```

6. נותר רק לממש את הפונקציה שקראנו לה func המתבצעת בסיום זמן המנייה של הטיימר. נזכור שיש לעשות זאת מחוץ ל ()Main ושהיא חייבת לקבל אובייקט כפרמטר:

```
public static void func(object state)
        {
            led.Write(!led.Read());
        }
```

כאן, הפכנו את המצב הקודם של הלד (NOT לוגי). אם הלד היה כבוי, הרי שיידלק ואם היה דלוק, יכבה.

7. בסה"כ קיבלנו את התוכנית הבאה:

```
using System;
using Microsoft.SPOT;
using Microsoft.SPOT.Hardware;
using STM32;
using System.Threading;
using System.Text;
namespace Task1
{
    public class Program
    {
        static OutputPort led = new OutputPort(NoaUpBasic.blue_Led, false);
        public static void Main()
        {
            Debug.Print("Started");
            Timer t = new Timer(func, null, 0, 1000);
            Thread.Sleep(Timeout.Infinite);
        }

        public static void func(object state)
        {
            led.Write(!led.Read());
        }
    }
}
```

8. נצרוב את התוכנה שכתבנו לבקר ע"י לחיצה עם העכבר על הלחצן Start שבסרגל הפקודות.
9. לאחר הצריבה של התוכנית, תופיע ההודעה "Started" בחלון ה Output של Visual Studio, הלד הכחול יידלק מיד ויתחיל להבהב בקצב שקבענו – כל שניה.

משימות

10. **משימה 1:** כתבו תוכנית המהבהבת בו זמנית את ארבעת הלדים שע"ג כרטיס הבקר שניתנים לשליטה. זאת ע"פ הדרישות הבאות:

צבע הלד	זמן מחזור	יחס מחזור
ירוק	0.5 שניה	50%
כתום	1 שניה	80%
אדום	2 שניות	20%
כחול	5 שניות	50%

11. **משימה 2:** כתבו תוכנית להפעלת פצצה מתקתקת ע"פ הדרישות הבאות:
    a. בלחיצת לחצן כחול הפצצה תופעת ותתחיל את הספירה לאחור עד הפיצוץ.
    b. הספירה לאחור תהיה של 15 שניות, כאשר הזמן הנותר מוצג מדי שניה בחלונית ה output של סביבת הפיתוח.
    c. הפיצוץ מסומן ע"י הבהוב כל ארבעת הלדים יחד.
    d. אין אפשרות עצור את הספירה לאחור.

ספירת זמן בין אירועים – מד מהירות תגובה

1. נפתח פרויקט חדש לעבודה עם הבקר.
2. בניסוי זה נכתוב תוכנה המודדת את מהירות התגובה של בן אדם לאירוע וויזואלי.
3. נגדיר לד ולחצן, כפי שעשינו זאת מקודם:

```
static InputPort pb = new InputPort(NoaUpBasic.pb2,
                                    true,
                                    Port.ResistorMode.PullDown);
static OutputPort led = new OutputPort(NoaUpBasic.red_Led, false);
```

4. אם מגדירים את הרכיבים האלה מחוץ לפונקציה הראשית (Main() בפרויקט זה לא חייבים לעשות זאת דווקא שם), יש לזכור להוסיף את המילה static בראש השורה.
5. נגדיר השהייה של שנייה אחת ולאחריה הגדרה של שני משתנים מסוג DateTime שכשמם כן הם, יכילו את הזמנים והתאריכים של שני אירועים: ההתחלה והסיום.

```
Thread.Sleep(1000);

DateTime dt1;

DateTime dt2;
```

6. נדליק את הלד הדום בכדי לסמן למשתמש שעליו ללחוץ על הלחצן.

```
led.Write(true);
```

7. מיד לאחר הדלקת הלד, נכניס למשתמנה הראשון שהגדרנו את התאריך והזמן כרגע.

```
dt1 = DateTime.Now;
```

8. כעט נמתין עד ללחיצת הלחצן ע"י המשתמש. מאחר ולא נדרש לבצע דבר תוך כדי ההמתנה, ניתן לעשות זאת באופן הבא:

```
while (!pb.Read());
```

זוהי למעשה לולאה ריקה שתנאי היציאה ממנה הוא הלחיצה על הלחצן.

9. מיד לאחר סיום הלולאה עלינו למדוד בשנית את הזמן כרגע, מאחר והמשתמש לחץ על הלחצן. את הזמן אחרי הלחיצה נאחסן במשתנה השני שהגדרנו.

```
dt2 = DateTime.Now;
```

10. נציג את ההפרש בין שני הזמנים שמדדנו בחלונית ה Output של Visual Studio בליווי הסברים חיוניים:

```
Debug.Print("Your response is: " + (dt2 - dt1).Seconds + " Sec. and "
            + (dt2 - dt1).Milliseconds + " mSec.");
```

11. לבסוף נכבה את הלד שכבר אין צורך בו

```
led.Write(false);
```

12. נכניס את כל התוכנית ללולאה אינסופית בכדי שתחזור על המדידה שוב ושוב.

13. בסה"כ מתקבלת תוכנה כמו זאת:

```
using System;
using Microsoft.SPOT;
using Microsoft.SPOT.Hardware;
using STM32;
using System.Threading;
using System.Text;

namespace Task
{
    public class Program
    {
        static InputPort pb = new InputPort(NoaUpBasic.pb2,
                                            true,
                                            Port.ResistorMode.PullDown);
        static OutputPort led = new OutputPort(NoaUpBasic.red_Led, false);

        public static void Main()
        {
            DateTime dt1;
            DateTime dt2;

            while (true)
            {
                Thread.Sleep(1000);
                led.Write(true);
                dt1 = DateTime.Now;
                while (!pb.Read());
                dt2 = DateTime.Now;
                Debug.Print("Your reaction is: " +
```

```
                        (dt2 - dt1).Seconds + " Sec. and " +
                        (dt2 - dt1).Milliseconds + " mSec.");
                led.Write(false);
            }
        }

    }
}
```

14. נצרוב את התוכנה לבקר ע"י לחיצה עם העכבר על בלחצן Start שבסרגל הפקודות.
15. לאחר הצריבה של התוכנית, נמתין עד להדלקת הלד האדום בכרטיס ומיד כשיידלק נלחץ על הלחצן הכחול שעל הכרטיס ונראה את מהירות תגובתנו מופיעה בחלון ה Output של Visual Studio.
16. נבצע כמה בדיקות ונחפש את הבאגים – מצבים אותם לא לקחנו בחשבון בכתיבת התוכנית ובהם היא לא פועלת כראוי.
17. ניתן לראות שקל לחזות את הזמן שבו יידלק הלד וללחוץ על הלחצן, עוד לפני שלמעשה זהינו את ההדלקה. ניתן להתגבר על הבעיה ע"י השהייה עד ההדלקה שונה בכל פעם. נשנה את התוכנית כך:

```
…
Thread.Sleep(1000);
Random rnd = new Random();
Thread.Sleep(rnd.Next(3000));
led.Write(true);
…
```

בשורות המודגשות הוספנו משתנה בשם rnd מטיפוס Random שיכול לחולל ערך אקראי בתחום רצוי של ערכים. לשם כך נשתמש בפונקציה ()Next הקיימת אצל משתנים מהטיפוס הזה. פונקציה זאת תחזיר ערך אקראי מסוג int, כפי שאנו רוצים, אך יש להגביל אותה עד הערך המקסימלי הרצוי, למשל 3000, כמו בדוגמא שלנו. מאחר ולא ניתן להגדיר ערך מינימלי של המספר האקראי, נשאיר את ההשהיה של שניה אחת שהייתה לפניכן.

18. באג נוסף של התוכנית הוא האפשרות להחזיק את הלחצן הכחול לחוץ כל הזמן ולקבל מהירות תגובה מצוינת – אפס. ניתן להתגבר על כך ע"י הוספת לולאה שתנאי היציאה שלה הוא לחצן משוחרר

```
while (pb.Read())
                {
                    Thread.Sleep(rnd.Next(3000));
                }
```

19. בדקו את הזמן התגובה שלכם! קחו ממוצע של כמה מדידות.

20. האם תוכלו למצוא באגים נוספים? כיצד ניתן לשפרם?

21. **משימה 1:** שפרו את התוכנה שכתבתם והוסיפו את החישוב של הממוצע של 5 התגובות אחרונות שנמדדו. התוכנית תציג אותה יחד עם הערך הנמדד בחלונית Output.
22. **משימה 2:** חשב והצג ממוצע של **כל** המדידות שנעשו עד כה.

בהצלחה!