# פרויקט קורס Real Time

מטרת הפרויקט להתריע על טמפרטורה קיצונית של עצמים, כאשר חיישן המרחק מזהה חפץ קרוב תתבצע מדידת טמפרטורה, אם הטמפרטורה תקינה הנורה תדלק ואם היא קיצונית הנורה תהבהב (אם חם תהבהב מהר אם קר תהבהב לאט

## משימות

### מדידת מרחק

* + משימה: task\_measure\_distance
    - קוראת באופן פריודי נתונים מחיישן HC-SR04 ומעדכנת משתנה משותף.

### מדידת טמפרטורה

* + משימה: task\_measure\_temperature
    - קוראת נתונים מהחיישן LM35 ומעדכנת נתון גלובלי

### התרעה על טמפרטורה קיצונית

* + משימה: task\_alarm
    - קוראת את נתוני המרחק והטמפרטורה המשותפים.
    - מפעילה התרעה (הבהוב נורית LED) בהתאם לטמפרטורה שנמדדה.
    - משתמשת בתור כדי לקבל בקשות מדידת טמפרטורה.

### שליטה בנורת LED

* + משימה: task\_blink\_led
    - מקבלת בקשות מהמשימה task\_alarm ומפעילה את נורת ה-LED בהתאם. (struct למצב הלד הכולל זמן וכו')

### הדפסת הודעות

* + משימה: task\_print\_message
    - מקבלת הודעות מתור המשימות (כולל זמן) ומדפיסה למסך

### [אופציונלי] הדפסה למסך

* + משימה: task\_LED\_message

## אמצעי סינכרון ותקשורת

### תורים

* תור לשמירת מדידות חיישן המרחק queueMeasuredDistances
* תור לשמירת הודעות להדפסה queueMessages

### Mutex/Semaphore

* מיוטקס להגנה על משתנה הטמפרטורה
* סמפורים להתראה על הודאות חדשות בתור (סימון ארוע שהתור מלא)

## מודולים

### מודול לניהול המשימות

* + אחראי על יצירת וניהול המשימות במערכת
    - פונקציות:
      * create\_tasks - יוצר את המשימות בתחילת התוכנית
      * create\_measure\_distance\_task יוצר את המשימה לקריאת המרחק
      * .... להוסיף פונקציות עבור כל משימה

### מודול לניהול אמצעי הסינכרון והתורים TBD

### מודול חיישן מרחק

* + אחראי על קבלת נתונים מחיישן HC-SR04.
    - פונקציות:
      * init\_distance\_sensor - אתחול חיישן המרחק.
      * read\_distance - קבלת מדידת מרחק (מחזירה או השמה למשתנה גלובלי.
      * [שליחת הודעה לתור על מדידה]

### מודול חיישן טמפרטורה

* + אחראי על קבלת נתונים מחיישן הטמפרטורה
    - פונקציות:
      * init\_temperature\_sensor- אתחול חיישני הטמפרטורה.
      * float get\_temperature- קבלת מדידת טמפרטורה מחיישן ספציפי.
      * [שליחת הודעה לתור על מדידה]

### מודול התרעה

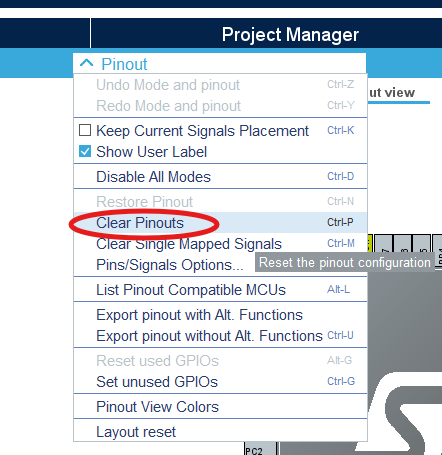
* + אחראי על ניהול ההתרעה בהתאם לטמפרטורה שנמדדה.
    - פונקציות:
      * init\_alarm- אתחול מערכת ההתרעה. [יאתחל את נורות LED]
      * alarm\_temperature - הפעלת התרעה בהתאם לטמפרטורה.

### מודול LED

* + אחראי על שליטה בנורת LED.
    - פונקציות:
      * void init\_led(void) - אתחול נורת LED.
      * void led\_on(void) - הדלקת נורת LED.
      * void led\_off(void) - כיבוי נורת LED.
      * void led\_blink(uint32\_t frequency) - הבהוב נורת LED בתדירות נתונה.

## הגדרות DCT

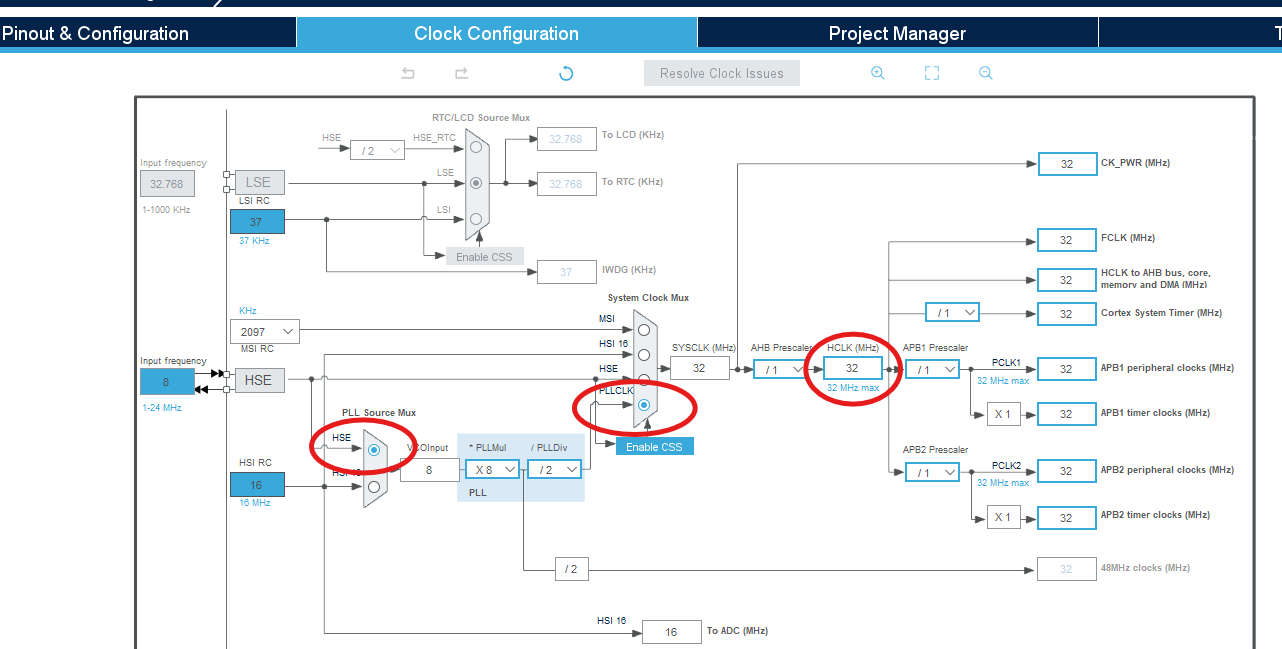
פתחי פרויקט חדש ואפסי את כל הפינים כך:



### 1. הגדירי את הטיימרים

#### 1.

#### 2.

****

#### 3.

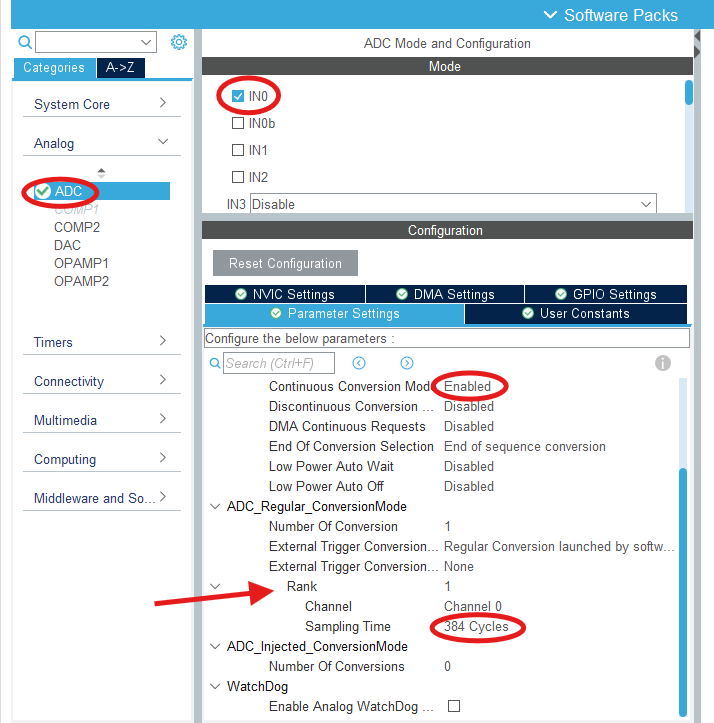
#### 4.

#### 5.

### 2. הוסיפי FREERTOS

### 3. הגדרת ADC עבור חיישן הטמפרטורה

#### 1.



#### 2.

#### 3. החיישן יהיה מחובר באופן פיזי בצורה כזו



**VCC**

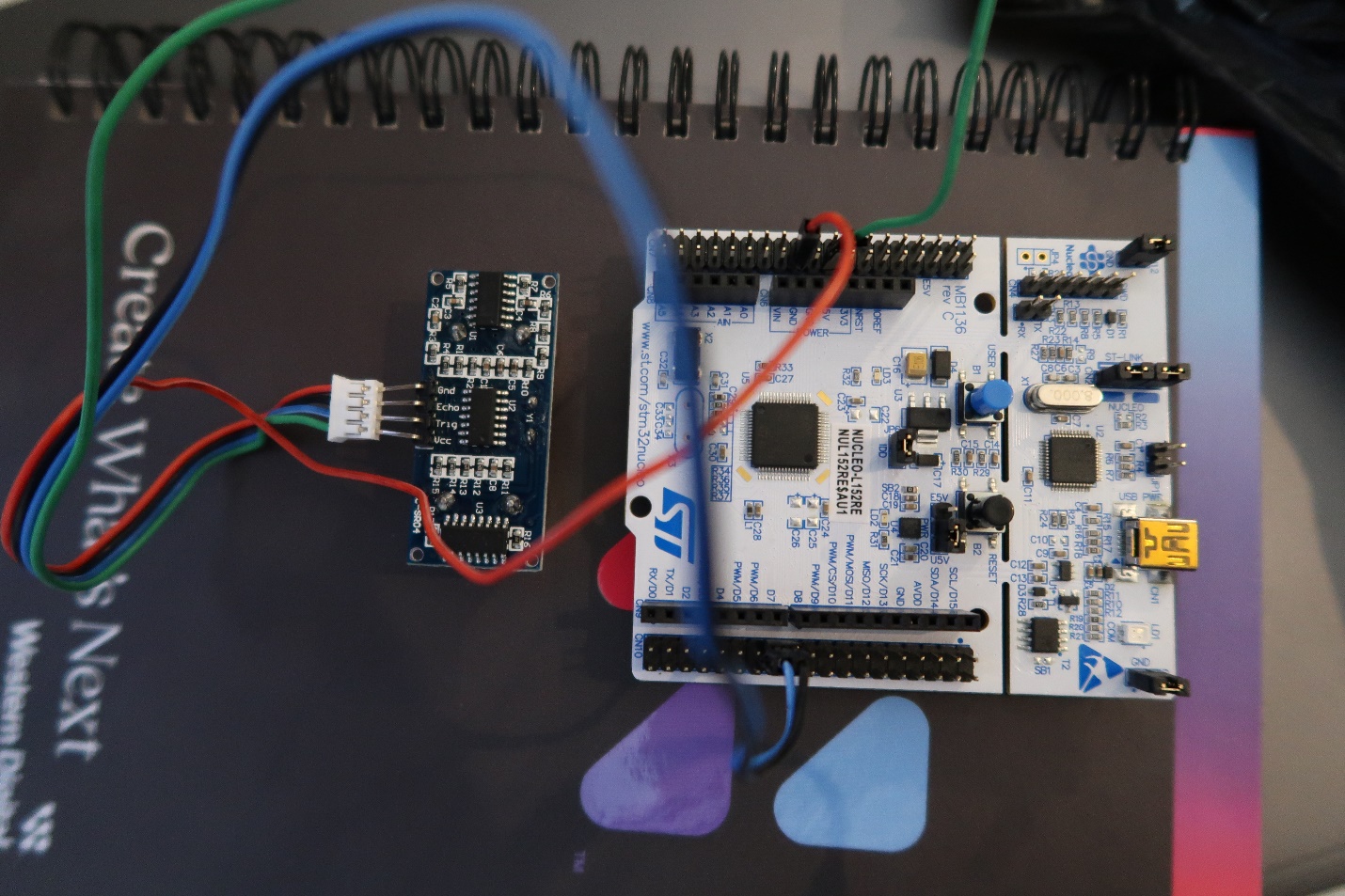
**A0**

**GND**

### 4. הגדרת הפינים עבור חיישן HC-SR04

****

**החיבור הפיזי:**



## קריאת הטמפ' מחיישן LM35

חיישן LM35 הוא חיישן טמפרטורה אנלוגי המפיק מתח פרופורציונלי לטמפרטורה את הערך האנלוגי יש להמיר למתח בטווח המסופק לחיישן (3.3V) 3300 מילי וולט.   
כדי לקרוא את המתח משתמשים ב ADC המובנה בבקר STM הממיר ערך אנלוגי לערך דיגיטלי בטווח של 12 ביטים  
כל 10 מילי וולט מייצג מעלה אחת בצלזיוס, למשל הערך 200 מייצג 20 מעלות.

**עקרונות פעולה:**

1. **קליטת מתח:** חיישן LM35 מייצר מתח פרופורציונלי לטמפרטורה. ככל שהטמפרטורה גבוהה יותר, כך המתח גבוה יותר.
2. **המרת מתח לערך דיגיטלי:** ADC מקבל את המתח מחיישן LM35 וממיר אותו לערך דיגיטלי בטווח 12 ביטים
3. **חישוב טמפרטורה:** באמצעות נוסחה ספציפית, הערך הדיגיטלי מהADC מומר לטמפרטורה בצלזיוס.

**נוסחת חישוב הטמפרטורה:**

|  |
| --- |
| Temperature = ADC Value \* (Voltage Supply / Analog Range) / Mili Volt per degree |

הצבת הערכים הנתונים:

ADC Value \* (3300 / 4096) / 10

\*טווח הערך האנלוגי הוא 212  = 4096

\*מכפילים ביחס בין המתח לטווח האנלוגי כדי להמיר את הנתון מטווח של 12 ביטים (4096) לטווח של 3300

**מימוש**:

* כדי להפעיל את המדידה יש תחילה לקרוא ל API:  
  HAL\_ADC\_Start(adc);
* אח"כ נשתמש ב Poll conversion כדי להמתין עד שהמדידה הסתיימה:  
  HAL\_ADC\_PollForConversion(adc, timeout);
* לבסוף נקרא את הערך האנלוגי שנמדד:  
  adc\_value = HAL\_ADC\_GetValue(adc);
* כדי לקבל את הטמפרטורה נשתמש בנוסחא שלעיל

## מדידת מרחק ע"י חיישן HC-SR04

חיישן HC-SR04 הוא חיישן אולטראסוני המשמש למדידת מרחק. הוא פועל על ידי שליחת גל אולטראסוני קצר וקבלת ההד שלו לאחר פגיעתו בחפץ. על ידי חישוב זמן המעבר של הגל והכפלתו במהירות הקול, ניתן לקבוע את המרחק לחפץ. (מתבסס על המשפט המתמטי ***מהירות \* זמן = דרך***)

עקרונות פעולה:

1. **שליחת גל אולטראסוני:** החיישן שולח פולס אולטראסוני קצר בתדר 40kHz.
2. **קליטת הד:** הגל האולטראסוני פוגע בחפץ וחוזר לחיישן.
3. **מדידת זמן המעבר:** החיישן מודד את הזמן שלוקח לגל לחזור מהרגע שנשלח עד לקליטת ההד.
4. **חישוב מרחק:** מרחק החפץ מהחיישן מחושב על ידי הכפלת מהירות הקול (בדרך כלל 343 מטר לשנייה) בזמן המעבר.

**נוסחת החישוב:**

Distance = (T2 – T1) \* Speed of sound

T1 - ערך הטיימר לפני שליחת הגל – זמן תחילת המדידה  
T2 - ערך הטיימר לאחר קליטת ההד – זמן סיום המדידה

ההפרש ביניהם מתקבל ביחידות מונה הטיימר,  
כדי לקבל את ההפרש בזמן יש לחלק את תוצרת ההפרש בתדירות שעון הטיימר.  
מהירות הטיימר שהגדרנו בבקר STM32 הוא 32MHz וכיוון שהגדרנו את ה Prescaler (גורם החלוקה של הטיימר, משמש לשליטה במהירות הטיימר) ל 32, **מהירות השעון היא 1MHz**

מהירות הקול היא 34,300 ס"מ בשנייה (343 מ' בשנייה)

אם כן החישוב הוא

Distance in cm = (T2 – T1) / timer speed \* cm per second speed of sound  
Distance in cm = (T2 – T1) / 1,000,000 \* 34,300

או בקיצור-  
(T2 – T1) \* 0.0343

\*ניתן לומר שהמספר 0.0343 מייצג את מהירות הקול במילי שניה (הקול עובר מרחק של 0.0343 ס"מ בכל מילי שניה)

**מימוש**:

איתחול הטיימר והפינים:

HAL\_TIM\_Base\_Start(timer); // timer init   
HAL\_GPIO\_WritePin(TRIG\_PORT, TRIG\_PIN, *GPIO\_PIN\_RESET*); // pull the TRIG pin low

קריאת הנתונים:

* הדלק את פין הטריגר למשך 10 מיקרו שניות:

מהירות הטיימר 1MHz ולכן תדירות התקדמות מונה הטיימר היא 1 מיקרו שניה; מאפסים את מונה הטיימר ואז המתנה עד שיהיה שווה ל 10, ואיפוס הפין בחזרה.

HAL\_GPIO\_WritePin(TRIG\_PORT, TRIG\_PIN, *GPIO\_PIN\_SET*); // pull the TRIG pin HIGH

\_\_HAL\_TIM\_SET\_COUNTER(sensorTimer, 0);

**while** (\_\_HAL\_TIM\_GET\_COUNTER (sensorTimer) < TIMER\_GAP); // wait for 10 us  
HAL\_GPIO\_WritePin(TRIG\_PORT, TRIG\_PIN, *GPIO\_PIN\_RESET*); // pull the TRIG pin low

* לאחר הטרגת המדידה יש להמתין עד שהחיישן ישלח אות שהתחילה המדידה ונשלח גל אולטרא סאונד (התנאי הנוסף למניעת המתנה אינסופית)

pMillis = HAL\_GetTick(); // used this to avoid infinite while loop (for timeout)

// wait for the echo pin to go high  
**while** (!(HAL\_GPIO\_ReadPin (ECHO\_PORT, ECHO\_PIN)) && pMillis + TIMER\_GAP > HAL\_GetTick());

* שמירת מונה הטיימר בזמן תחילת המדידה (שליחת הגל)

time\_start = \_\_HAL\_TIM\_GET\_COUNTER (sensorTimer);

* יש להמתין עד שההד יחזור לחיישן והחיישן יאפס את פין ה ECHO (תנאי נוסף למניעת timeout – המתנה אינסופית במקרה של תקלה)

pMillis = HAL\_GetTick(); // used this to avoid infinite while loop (for timeout)

// wait for the echo pin to go low  
**while** ((HAL\_GPIO\_ReadPin (ECHO\_PORT, ECHO\_PIN)) && pMillis + 50 > HAL\_GetTick());

* קבלת זמן סיום המדידה

time\_end = \_\_HAL\_TIM\_GET\_COUNTER (sensorTimer);

* חישוב המרחק עפ"י זמני תחילת וסיום המדידה

**Distance** = (**float**)(time\_end - time\_start) \* SPEED\_OF\_SOUND;