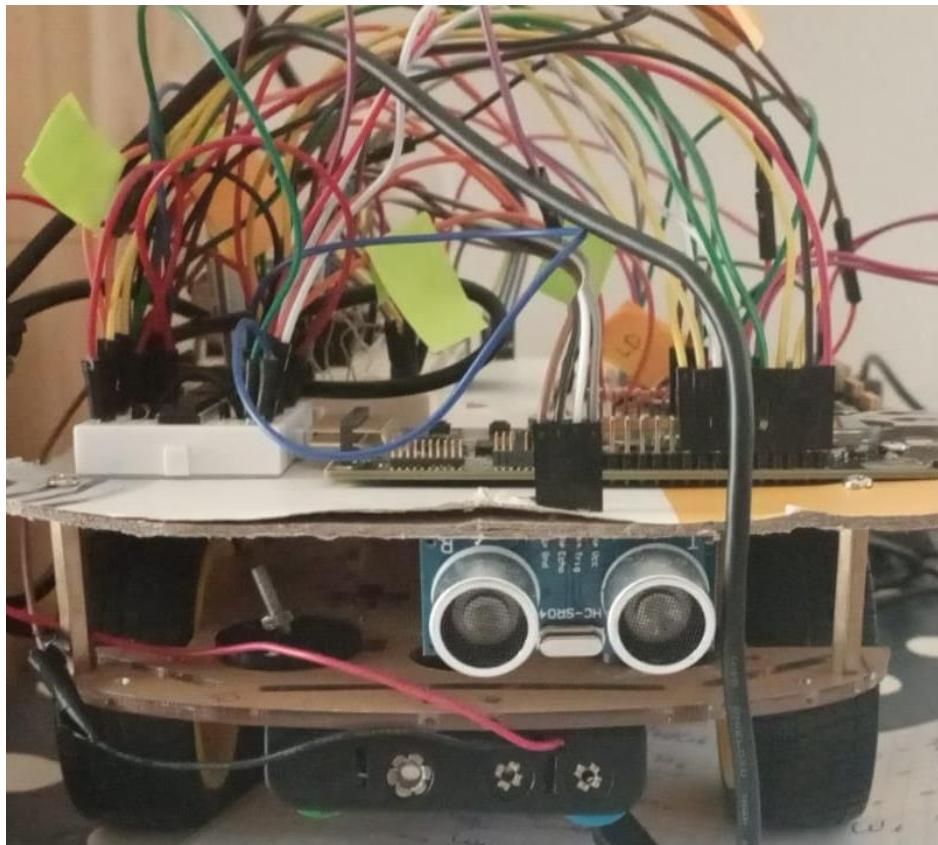


פרויקט מסכם ב邏יקו בקרים - רכב נשלט bluetooth

שם הסטודנט, תעודה זהות, מייל: [סער גוזמן](mailto:saar.gozlan@e.braude.ac.il), 204188403
שם הסטודנט, תעודה זהות, מייל: [אורן דנילוב](mailto:oren14dani@gmail.com), 203824545

מנחה: [גל ליביו](#)
מרצה: [פאל טרייף](#)



תוכן עניינים

1	פרויקט מסכם בマイקו בקרים - רכב נשלט BLUETOOTH
2	תוכן עניינים
4	פונקציונליות המערכת:
4	רכיבים.....
6	תרשים מלבים.....
6	פונקציונלי.....
6	מפורט.....
7	תרשים זרימה.....
7	פעולת המערכת.....
8	פעולת המשמש.....
8	קונfigורציה רכיבים 8051:
8	הקצת משאבים בברק :
9	I/O PORTS
9	SYSTEM CLOCK
9	INTERRUPTS
10	UART0 AND TIMER1
10	<i>baude rate</i>
10	השניה בין מדידה למדידה
10	BUZZER
11	TIMER 0
11	<i>אוז trigger</i>
11	PCA
12	SPI
13	קונfigורציה הרכבים ARM:
13	הקצת משאבים בברק :
13	I/O PORTS & CROSSBAR
13	שעון המערכת , O _{SC}
13	UART0
14	TIMER3 AND PWM
14	הבנת תפקוד ה PWM
14	אילזמים
15	אופן המימוש בפרויקט
15	איתחול הרכיב
16	פונקציות.....
16	listen
16	פסיקות:
16	פונקציות הפעלה למדידת המרחק
16	פונקציות להפעלת הזמוץ
17	שליחה התראת קירבה לBT
17	ברק EFM32WG:ARM
17	פונקציות איתחול:
17	פונקציית הדלקה/כיבוי:
17	פונקציות בקרה
17	פקודות תנועה:

חזר לתוכן עניינים

הסברים נוספים לשיקול התכנון.....	
17	
17	ניתוח פולס ECHO
17	אופן מדידת אורך זמן פולס ECHO על ידי המרכיבת:
18	חישוב הזמן
18	חישוב המרחק
19	איך מיצרים פולס טריגר באורך המתאים?
19	איך מחשבים מחלק מתח לרגל O ? ECHO
21	מימוש התוכנית.....
21	EFM8UB2:8051
21	Main
22	Interrupts
23	general.h
24	פונקציות
29	EFM32WG:ARM
29	main
29	init_device.c
33	תוצאות.....
33	תמנוני וסתוני הדוגמה לפעולת המערכת
33	הדוגמה להימנעות ממכשולים
33	הדוגמה לשילטה והיגוי ע"י המשמש
33	נסיעת ניסיין בשיטה
34	צילום בSCOPE של אות-trigger ואות-ECHO
35	צילום בSCOPE של אות PWM עברו CYCLE DUTY מסוים:
36	צילום בSCOPE של אות בקרה כפי שנקלט בטלפון הנייד דרך BT :
37	שימוש בסקאץ DEBUG
42	תהליך העבודה ותיאור תקלות שהתעוררו במהלך הפרויקט.....

פונקציונליות המערכת:

1. שליטה וניהוט על כיוון ומהירות הנסיעה מהפלאפון באמצעות Bluetooth, על ידי שליחת אותות בקרה בצורה

תווים:



-	F - תנועה קדימה.
-	V - תנועה אחורה (רוורס).
-	S - עצירה.
-	R - סיבוב ימינה.
-	L - סיבוב שמאליה.
0	PWM מצב מהירות מנוע עם 75%
1	PWM מצב מהירות מנוע עם 85%
2	PWM מצב מהירות מנוע עם 95%
	case insensitivity



2. מערכת למניעת התנגשויות בעצמים בדרך, בעלת 2 חיישני (mdi מרחק) אולטרה סוני

חיצוניים, מלפנים ומאחור.

3. מייצרת התראה במידה והאובייקט קרוב מדי, התראה גם נשלחת דרך BT אל המשתמש.

4. זמזם התראה מתחילה לפעול מתחת 30 ס"מ.

5. הזזם משנה את תדריות הצלצולים ככל שהמרחק קטן.

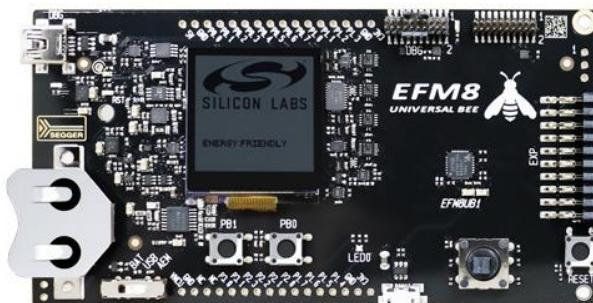
רכיבים

המערכת מבוססת מיקרו בקרים ומכליה את הרכיבים הבאים:

- מיקרו בקר EFM32WG:ARM של חברת Silicon Labs



- מיקרו בקר EFM8UB2:8051 של חברת Silicon Labs



- פלטפורמת רכב - 4 מנועי DC ו 4 גלאלים.

- 2 דרייברים L293D לשיליטה על המנוע.



- 2 לוחצים.
- 2 חיישני (mdi מרחק) אולטרא סוני.



- ערוץ תקשורת טורי UART.
- מושדר BT מסוג HC-06.

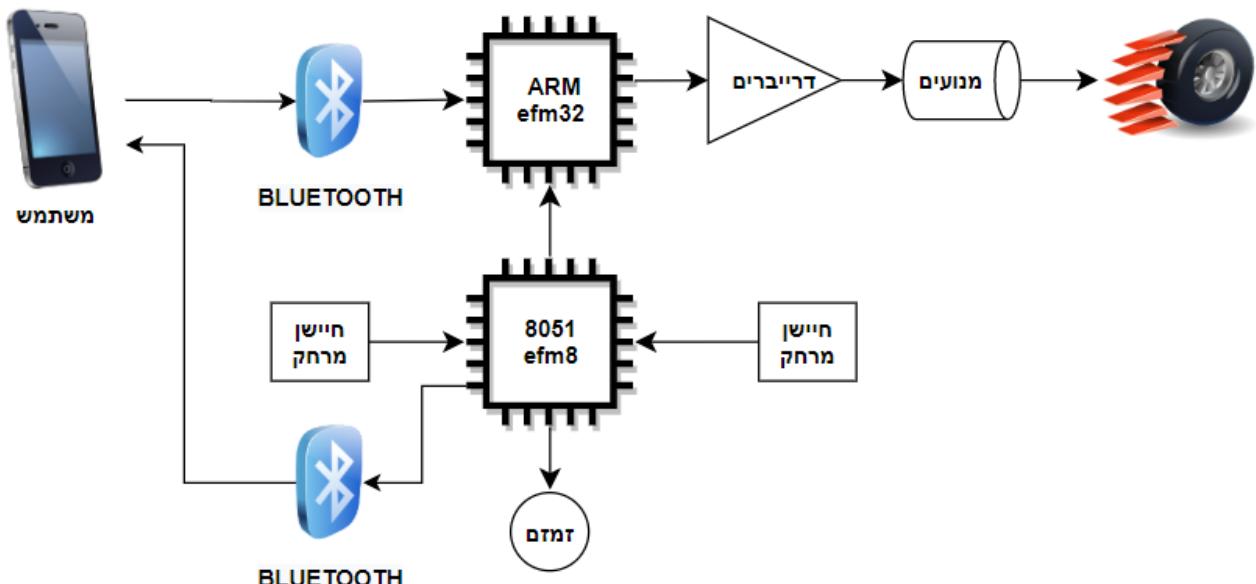


- זמזהם (באזר).

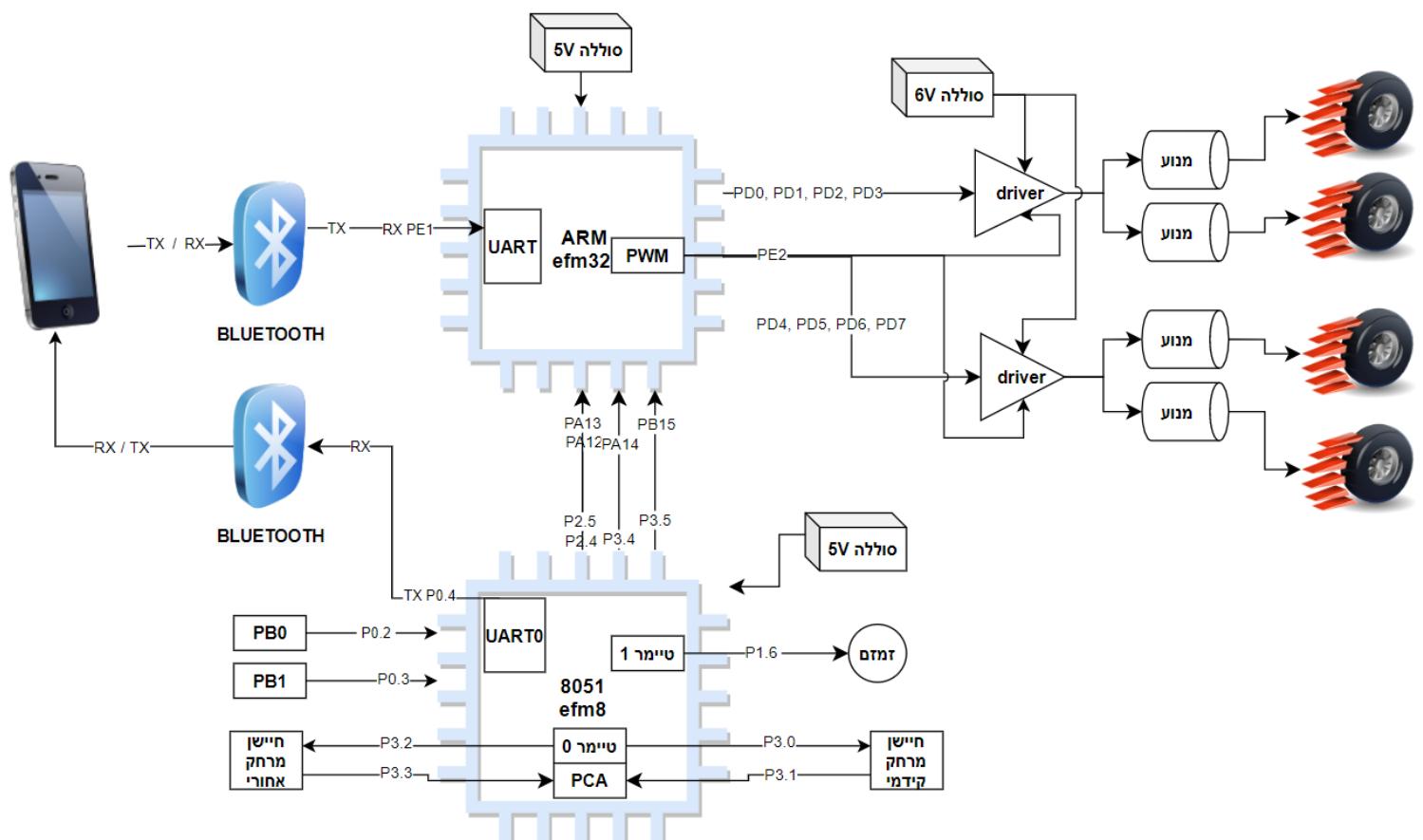


תרשים מלכניים

פונקציונלי

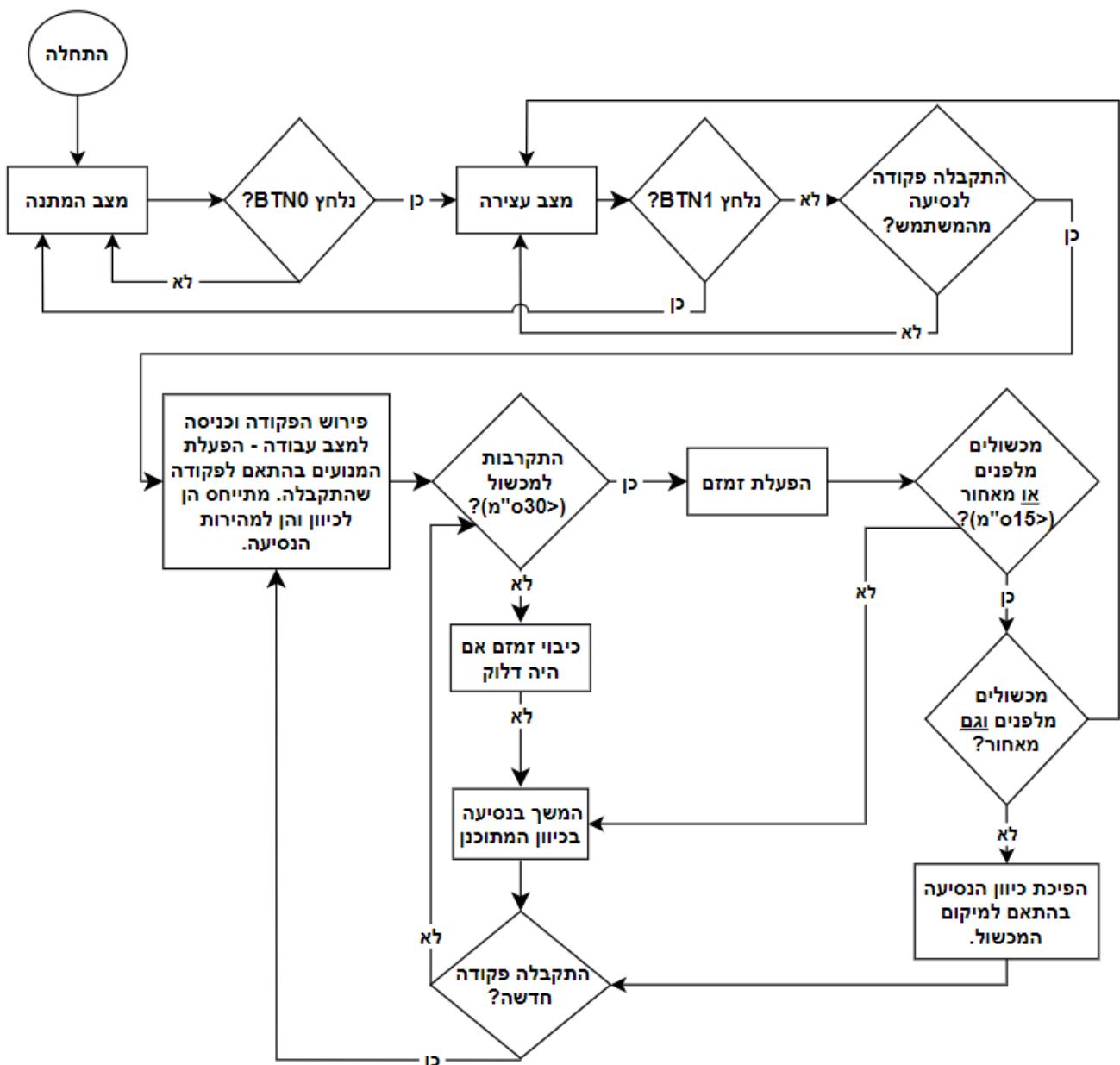


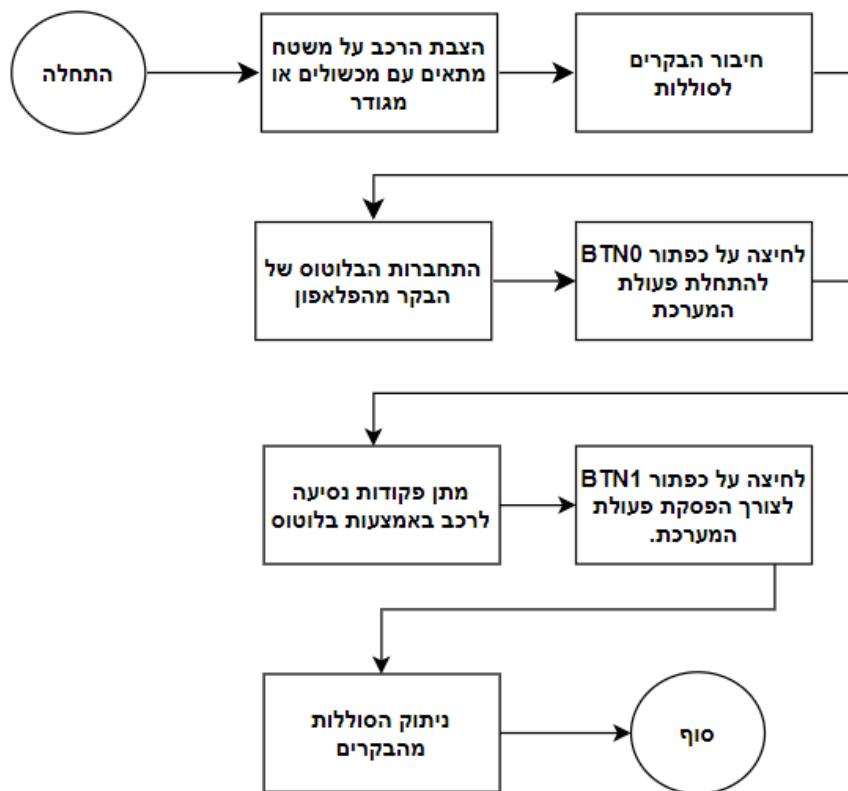
מפורט



תרשים זרימה

פעולות המערכת



**קונפיגורציה רכיבים 8051:**

הקצתה מישאים בברק :

8051 efm8							
SYSTEM	T1	T1	T1	T0	PCA CH0	PCA CH1	
הקצאה	system clock	UART0 baude rate	השייה בין מדידות	קביעת תדר הבאזור	trigger signal fornt and back (אולטרה סוני)	ECHO signal front (קדמי)	ECHO signal back (אולטרה סוני) (אחורית)
תדר [Hz]	24M	9600	16	משתנה בהתאם למרחק	100k	2M	2M
מדוע נסוף		baud rate	62.5 msec	משתנה בהתאם למרחק	10 usec	CAPTURE mode	CAPTURE mode

8051 efm8			
func	port	pin	in/out
PB0	0	2	in
PB1	0	3	in
BTN0 (efm8 to efm32)	2	4	out
BTN1 (efm8 to efm32)	2	5	out
alert back (to efm8)	3	4	out
alert front (to efm8)	3	5	out
buzzer / LED	1	6	out
Trigger front	3	0	out
Trigger back	3	2	out
Echo front / PCA ch0	3	1	in
Echo back / PCA ch1	3	3	in
UARTtx_BTrx	0	4	out

system clock

Properties	Properties
Properties of Clock Control	Properties of HFOSC
Clock Control	HFOSC
Property	Value
Clock Select	
SYSCLK	24.000 MHz
USBCLK	48.000 MHz
Select System Clock Source	High-Frequency Oscillator / 2
Select USB Clock Source	High-Frequency Oscillator
SYSCLK Output Settings	
SYSCLK Crossbar Output Synchronization	Not synchronized
Property	Value
High Frequency Oscillator Control	
HFOSC Undivided Frequency	48.000 MHz
HFOSC Divided Frequency	48.000 MHz
Enable Oscillator	Enabled
Enable Oscillator Suspend	Disabled
Oscillator Frequency Divider Control	Divide by 1

השתדלנו לבחור תדר שעון למערכת מהיר ככל הניתן כדי שפעולות העבודה יהיו כמה שיותר מהירות. ובנוסף לקבלת תדרים מתאימים לכל הטימרים השונים וגם בשביל הופך בחרנו תדר שעון מערכת זוגי כדי לקבל את ה1 מגה הרץ בשבילים בחלוקת.

interrupts

Properties	Properties
Properties of Interrupts	Properties of External Interrupts
Interrupt Enables	External Interrupts
Property	Value
Code Generation Options	
Generate Interrupt Functions	Disabled
Extended Interrupt Enable 1	
Enable ADC0 Conversion Complete Interr	Disabled
Enable ADC0 Window Comparison Interr	Disabled
Enable Comparator0 (C0) Interrupt	Disabled
Enable Comparator1 (C1) Interrupt	Disabled
Enable Programmable Counter Array (PC	Disabled
Enable SMBus (SMB0) Interrupt	Disabled
Enable Timer 3 Interrupt	Enabled
Enable USB (USB0) Interrupt	Disabled
Extended Interrupt Enable 2	
Enable SMBus1 Interrupt	Disabled
Enable Timer 4 Interrupt	Disabled
Enable Timer 5 Interrupt	Disabled
Enable UART1 Interrupt	Disabled
Enable VBUS Level Interrupt	Disabled
Interrupt Enable	
Enable All Interrupts	Enabled
Enable External 0 Interrupt	Enabled
Enable External 1 Interrupt	Enabled
Enable SPI0 Interrupt	Enabled
Enable Timer 0 Interrupt	Disabled
Enable Timer 1 Interrupt	Disabled
Enable Timer 2 Interrupt	Enabled
Enable UART0 Interrupt	Disabled
Property	Value
INT0/INT1 Configuration	
INT0 Polarity	Active low
INT0 Port Pin Selection	P0.2
INT1 Polarity	Active low
INT1 Port Pin Selection	P0.3

UART0 and Timer1

baude rate

שימוש ב timer1 לצורך איפשר התקשרות UART0, על ידי ספירת Tbit ב UART שמש בtimer1 לבקר הUART, הורדנו ב 8051 את האופציה לתקשרות UART. ואנו מחייבים מתי להפעיל את השטימר, וצריך גם להפסיק אותו.

$$N = 2^8 - \frac{f_{clock}}{f_{timer}} = 2^8 - \frac{2M}{9600 * 2} = 152$$

נשים לב ש כדי לקבל קצב של 9600 יצרנו תדר כפול בשטימר.

Properties	
Properties of Timers	
TIMER Setup	
TIMER 0/1	TIMER 2
TIMER 3	TIMER 4
TIMER 5	
Property	Value
Clock Control 0	
Timer 0/1 Prescale	SYSCLK / 12
> Timer 0	
> Timer 0 Firmware Control	
Timer 1	
Mode	Mode 2, 8-bit Counter/Time..
Clock Frequency	2.000 MHz
Clock Source	Use the SCA prescale clock
Timer or Counter	Timer mode
Timer Running State	Stopped
Timer Switch 1: Run Control	Stop
Timer Switch 2: Gate Control	Disabled
> External Interrupt Settings	

Properties	
Properties of Timers	
TIMER Setup	
TIMER 0/1	TIMER 2
TIMER 3	TIMER 4
TIMER 5	
Property	Value
> Timer 0 Mode 2: 8-bit Counter/Timer wi	
> Timer 0 Overflow for Peripherals	
Timer 1 Mode 2: 8-bit Counter/Timer wi	
Clock Source Frequency	2.000 MHz
Clock Source Period	500.000 ns
Target Overflow Frequency	19200 (0x4B00)
Timer Init Overflow After	52.000 us
Timer Init Value	152 (0x98)
Timer or Counter	Timer mode
Timer Reload Overflow Frequency	19.231 kHz
Timer Reload Overflow Period	52.000 us
Timer Reload Value	152 (0x98)
Timer 1 Overflow for Peripherals	
SMBus0 SCK Frequency	6.410 kHz
UART0 Baud Rate	9.615 kHz

השניה בין מדידה למדידה

בנוסף, אנו משתמשים ב timer1 לביצוע ספירת השניה בין מדידה למדידה, על פי ספירה של Tbit בכל פעם שטימר מגיע לאירוע (זיכרון השטימר עובד בתדר של $2 * 9600$). ההשניה אצלו היא 1/16 שניות.

buzzer

לצורך חיווי טוב יותר הגדרנו את הזמן לשנות את תדר הצלצלים שלו בהתאם למרחק של האובייקט. ככל שהאובייקט קרוב יותר כך התדר עולה, עד שהוא מגע לפונקציית המתנה של החז' שנייה בין הדגימות. כדי למשוך זאת בחרנו להתלבש על פונקציית המתנה של החז' שנייה בין הדגימות. כך שהתדר המינימלי שיכלנו לאפשר לזמן היה 2 הרץ. פונקציית המתנה מתבססת על טימר 1 שוגלש כל 52 מיקרו שניות. לעומת זאת לאחר 9600 גליות תעבור חז' שנייה.

ומכאן אפשר לגזר את זה ולתגใจ:
שאחרי 4800 גליות תעבור רביע שנייה
אחרי 2400 גליות שמנית שנייה וכו'.
כלומר מתקיים:

$$f_{buzzer} = \frac{1}{N_{\text{גליות}} * 52 * 10^{-6}} = \frac{19200}{N_{\text{גליות}}} = \frac{19200}{\left(\frac{\left(\frac{9600}{8} \right)}{scale} \right)} = 19200 * \frac{scale}{1200} = 16 * scale[\text{Hz}]$$

ולכן ניתן להציג סמכות נוספת לפונקציה המתנה ולנצל תוכנה זאת לצורך בקרת זמן.

הfonקציה (`waitingNbuzzzer()`) מקבלת את מצב הזמן המתבקש (`ton_state`) ומפנה את מצבו הלוגי של הזמן רק כאשר ערכו של `ton_state` נערך בין 1 ל 4. לפיה הגדרה הבאה בחרנו את הנקודות שבהם הזמן יעבור היפוך במצבו הלוגי, כולם נדילק ונכבה אותם במקצת קבוע לפי המרחק, באופן הבא:
נסמן מרחק ב- X.

- $30 \geq X$ אין שינוי.
- $20 \leq X < 30$ כל 1200 גליות נבצע היפוך. [Hz]
- $15 \leq X < 20$ כל 600 גליות נבצע היפוך. [Hz]

- $15 < X \leq 10$ כל 300 גלישות נבצע היפוך. $[Hz] = 64$
- $10 < X \leq 5$ כל 150 גלישות נ被执行 היפוך. $[Hz] = 128$
- $X > 5$ אין שינוי.

זיכרון של מילכתית קיימת פונקציה בשם `alarm_func` שדואגת לספק '1' לזמןם בעת חירום. בעת חירום קבועה לפיה פונקציה בשם `alarm_state` שיודעת להזכיר על חירום לפיה המרחק. וכן כשיין שינוי הכוונה היא יותר לומר שמיירת מצב. המצביע נקבע מילכתית לפיה שתי הפונקציות שהוזכרו עכשו.

timer 0

trigger

שימוש בטימר 0 לספירת זמן של אות *trigger* הנשלח לחישון מרחק, כך שתתקבל דגל גלישה כל $sec \mu 10$.
לכן הטימרעובד בתדר $100kHz = \frac{1}{10\mu sec}$.

אנחנו מחליטים מתי להפעיל את הטימר על פיהו. חישוב ערכי התחלת של טימר 0 במצב 8 סיביות עם טעינה אוטומטית:

$$N = 2^8 - \frac{f_{clock}}{f_{timer}} = 2^8 - \frac{2M}{100k} = 236$$

Properties

Properties of Timers

TIMER Setup	TIMER 0/1	TIMER 2	TIMER 3	TIMER 4	TIMER 5
Property	Value				
▼ Clock Control 0					
Timer 0/1 Prescale	SYSCLK / 12				
▼ Timer 0					
Mode	Mode 2, 8-bit Counter/Timer..				
Clock Frequency	2.000 MHz				
Clock Source	Use the SCA prescale clock				
Timer or Counter	Timer mode				
Timer Running State	Stopped				
Timer Switch 1: Run Control	Stop				
Timer Switch 2: Gate Control	Disabled				
▼ Timer 0 Firmware Control					
Timer Start or Stop	TRO(TCON.4)				
Timer Interrupt Cleared by	By Hardware				
Timer Interrupt Enable Flag	ET0(IE.1)				
Timer Interrupt Pending Flag	TF0(TCON.5)				
Timer Overflow Flag	TF0(TCON.5)				
▶ Timer 1					
▶ External Interrupt Settings					

Properties

Properties of Timers

TIMER Setup	TIMER 0/1	TIMER 2	TIMER 3	TIMER 4	TIMER 5
Property	Value				
▼ Timer 0 Mode 2: 8-bit Counter/Timer wi					
Clock Source Frequency	2.000 MHz				
Clock Source Period	500.000 ns				
Target Overflow Frequency	100000 (0x186A0)				
Timer Init Overflow After	10.000 us				
Timer Init Value	236 (0xEC)				
Timer or Counter	Timer mode				
Timer Reload Overflow Frequency	100.000 kHz				
Timer Reload Overflow Period	10.000 us				
Timer Reload Value	236 (0xEC)				
▼ Timer 0 Overflow for Peripherals					
ADC0 Conversion Start Signal	100.000 kHz				
PCAO Clock Source	100.000 kHz				
SMBus0 SCK Frequency	33.333 kHz				
▶ Timer 1 Mode 2: 8-bit Counter/Timer wi					
▶ Timer 1 Overflow for Peripherals					

PCA

ערוץ 0 משמש למדידת אות ECHO במקש עם רכיב מודד מרחק אולטרא סוני אחריו.
ערוץ 1 משמש למדידת אות ECHO במקש עם רכיב מודד מרחק אולטרא סוני קידמי.
בעלית את אות ECHO עושים דגימה קש של המונה בערוץ 0 ובירידת האות עושים דגימה חסop. הפירוט לביצוע וניתוח הדגימה פורט כבר בחלק אחר בדו"ח.

[חזר לתוכן עניינים](#)

Properties					
Properties of PCA					
PCA 0	Channel 0	Channel 1	Channel 2	Channel 3	Channel 4 / WDT
Property	Value				
▼ PCA Start / Run					
PCA Counter/Timer Run Control	Stop				
PCA Counter/Timer Run Status	Stopped				
▼ Watchdog Control					
Clock Source	PCA Clock				
Enable Watchdog Timer	Disabled				
Lock Watchdog Timer	Unlocked				
▼ PCA Counter/Timer Configuration					
PCA Clock Frequency	2.000 MHz				
PCA Clock Period	500.000 ns				
Select PCA Counter/Timer Pulse	SYSCLK / 12				
PCA Counter/Timer	0 (0x0)				
PCA Counter/Timer Idle Control	Normal				
Enable PCA Counter/Timer Overflow Int	Disabled				

Properties					
Properties of PCA					
PCA 0	Channel 0	Channel 1	Channel 2	Channel 3	Channel 4 / WDT
Property	Value				
▼ PCA Control					
Channel Capture/Compare Mode	Capture triggered by positive edge				
▼ PCA Channel					
Capture/Compare Flag Interrupt	Enabled				
Capture/Compare Register	0 (0x0)				

Properties					
Properties of PCA					
PCA 0	Channel 0	Channel 1	Channel 2	Channel 3	Channel 4 / WDT
Property	Value				
▼ PCA Control					
Channel Capture/Compare Mode	Capture triggered by positive edge				
▼ PCA Channel					
Capture/Compare Flag Interrupt	Enabled				
Capture/Compare Register	0 (0x0)				

SPI

Properties	
Properties of SPI 0	
SPI 0	
Property	Value
▼ View	
View	Simple
▼ Control	
SPI Enable	Enabled
Bus Mode (Master)	Master 3-wire mode
▼ Configuration	
SPI Mode	Master
▼ Clock Rate	
SYSCLK	24.000 MHz
SPI Clock Frequency (Target)	1000000
SPI Clock Frequency (Actual)	1.000 MHz
SYSCLK Divider Coefficient (SPIOKR)	11 (0xB)

קונפיגורציה הרכיבים :ARM

הקצתת משאבים בבקר :

ARM efm32			
	SYSTEM	UART	T3
הקצאה	system clock	UART	PWM
תדר [Hz]	1M	9600	1M
מידע נסוף		baud rate	CAPTURE mode

I/O PORTS & CROSSBAR

ARM efm32			
func	port	pin	in/out
motor 1 input1	D	0	out
motor 1 input2	D	1	out
motor 2 input1	D	2	out
motor 2 input2	D	3	out
motor 3 input1	D	4	out
motor 3 input2	D	5	out
motor 4 input1	D	6	out
motor 4 input2	D	7	out
PWM (to motor drivers)	E	2	out
BTN0 (efm8 to efm32)	A	12	in
BTN1 (efm8 to efm32)	A	13	in
efm8 stop alert back	A	14	in
efm8 stop alert front	B	15	in
UART0 TX	E	0	OUT
UART0 RX	E	1	IN

בקוד ביצענו את האיתחולים הרלוונטיים בפונקציית `:Device_Init`

```
void Device_Init(void) {
    Set_Clock_Gpio();
    Engine_Init();
    UART_Init();
}
```

וכל רכיב קיבל אתחול משלו.

שעון המערכת , GPIO

```
void Set_Clock_Gpio(void){
    CMU_HFRCOBandSet(cmuHFRCOBand_28MHz); // Set HFRCOCTRL (High Freq. RC0sc.) to 1 MHz
    CMU_ClockEnable(cmuClock_GPIO, true); // Enable GPIO peripheral clock
}
```

מגדיר את השעון הראשי ומאפשר שעון לGPIO.

UART0

הUART בבקר ARM משתמש אותנו לקבלת של אותות בקרה מהמשתמש בצורה טווים (char).

חזר לתopic עניינים

```

void UART_Init(void){
    CMU_ClockEnable(cmuClock_GPIO, true);
    CMU_ClockEnable(cmuClock_UART0, true);

    GPIO_PinModeSet(gpioPortE, 0, gpioModePushPull, 1); // U0_TX is push pull
    GPIO_PinModeSet(gpioPortE, 1, gpioModeInput, 1); // U0_RX is input

    // Start with default config, then modify as necessary
    USART_InitAsync_TypeDef config = USART_INITASYNC_DEFAULT;

    config.baudrate = 9600; // CLK freq is 1 MHz
    config.autoCsEnable = true; // CS pin controlled by hardware, not firmware

    config.enable = usartDisable; // Make sure to keep USART disabled until it's all set up
    USART_InitAsync(UART0, &config);

    // Set and enable USART pin locations
    UART0->ROUTE = UART_ROUTE_CLKPEN
        | UART_ROUTE_CSPEN
        | UART_ROUTE_TXPEN
        | UART_ROUTE_RXPEN
        | UART_ROUTE_LOCATION_LOC1;

    USART_Enable(UART0, usartEnable);
}

```

Timer3 and PWM

טיימר 3 הוא אחראי על תזמון ה-PWM.

הנתן תפקיד ה-PWM

PWM משמש אותנו לשיליטה במתכונת המנוח ופועל גם של הגלגים. בעזרתו שינויuty של אותותינו יכולים לשנות במדויק את המנוח של הדריבר, אשר אחראי על פועלות המנוחים. ככל שగובה יותר, כך ממוצע המתנה היוצא הוא גבוה יותר, וכך הדריבר מקבל מתח ממוצע גבוה יותר ומאנץ את המנוחים בהתאם. לדוגמה:



עבור 75% מתקובל מתח גובה יחסית, ולכן המנוחים יסתובבו מהר. עבור 25% מתקובל מתח נמוך יחסית, ולכן המנוחים יסתובבו לאט.

אילוצים

לאחר עיון ב data sheet של דרייבר המנוח מצאנו 2 אילוצים:

1. התדר המקסימלי שניתן לספק לדרייבר הוא 5kHz .

$$top\ value = \frac{timer3_{FREQ}}{OUT_{FREQ}}$$

ছিৰ লিপি উচ্চিন

লকন, গড়েন আট তাদৰ শল টাইমাৰ ৩, কলোৱা PWM, লহিয়ত **2.3k**Hz.

২. মতাখ মিনিমাল শনচাব '১' লগি হো 2.3. কলোৱা, আল মতাখ শমতাখত 2.3.3 লোলত ইচ্ছব লজ্জেল অৱু' লগি.

মাছৰ ওভেক্ট মোচিয়া মক্সিমুম শল 3.3 অনো মবিনিম চি হেক্যুল duty cycle মিনিমাল আপশৰি বাওত PWM হো:

$$\text{min duty cycle} = \frac{2.3}{3.3} = 0.697 = 69.7\%$$

ৱেল মন্ত লহিমনু মমচৰ শবো রেশ বাওত PWM উলোল লশব আট ফুলত মুৰুক্ত অনো গড়েন আট হেক্যুল duty cycle কেতন
বিয়োৱা আপশৰি ল 75% (৫% মুল গৱোল তথ্বতো).



ক্ৰ অনো যোলিম লহাইয়া মক্ষম ল শিনো মহিৰত বাওণ হো:

মমচৰ ০ - duty cycle 75%.

মমচৰ ১ - duty cycle 85%.

মমচৰ ২ - duty cycle 95%.

আপো হমিমুষ বৰ্পোক্ত

ৱেল মন্ত লক্বোৱা আট হেক্যুল duty cycle নবেৰ আট চিষুব বাও লচৰু কৰিয়ত হুৰু অলো টাইমাৰ ৩ যোৱা, ক্তলোত বাও রেচৰি
বিয়াহ:

$$\text{compare value} = \frac{\text{timer3}_FREQ}{OUT_FREQ} \cdot \frac{\text{duty cycle}}{100} = \text{top value} \cdot \frac{\text{duty cycle}}{100}$$

লমুশা, মহনোচ্ছা নিতন লহীন শল টাইমাৰ সোৱা উল মন্ত লস্ফোৱ জৰুৰ মচৰু শলম, গম এফা শহো বদৰ
হো মগিয় ল compare value শবো মোৰাই আট আটো. ক্ৰ মক্ষিলিম আট হেক্যুল duty cycle রেচৰি.



আিথাল হৰ্কিব

```

void PWM_Init(void){
    CMU_ClockEnable(cmuClock_TIMER3, true); // enable Timer3 clock.
    GPIO_PinModeSet(gpioPortE, 2, gpioModePushPull, 0); // Configure port E pin 2 as digital output (push-pull)

    uint32_t top;
    uint32_t compare_val;
    top=timer3_FREQ/OUT_FREQ;
    compare_val = ((timer3_FREQ*1.0)/(OUT_FREQ*1.0) )*(DUTY_CYCLE/100.0);

    TIMER_TopSet (TIMER3,top); // set top timer 3.
    TIMER_CounterSet(TIMER3, 0); // start counter from 0 up-count mode.
    TIMER_CompareSet(TIMER3, 2, compare_val); // set cc2 compare value 25 duty cycle.
    TIMER_CompareBufSet(TIMER3, 2, compare_val); // set cc2 compare buffer value 25 duty cycle.
    TIMER3->ROUTE = (1 << 16)| (1 << 2) ; // connect pwm output to port E pin 2.
}

```

```

חוור לתוכן עניינים
// setup timer channel configuration for pwm.
TIMER_InitCC_TypeDef timerCCInit =
{
    .eventCtrl=timerEventEveryEdge,
    .edge=timerEdgeNone,
    .prsSel=timerPRSSELCh0,
    .cufoa=timerOutputActionNone,
    .cofoa=timerOutputActionSet,
    .cmoa=timerOutputActionClear,
    .mode=timerCCModePWM,
    .filter=false,
    .prsInput= false,
    .coist=false,
    .outInvert=false,
};

//configuration for timer3 on channel 2.
TIMER_InitCC(TIMER3, 2, &timerCCInit);
// setup timer configuration for pwm.
TIMER_Init_TypeDef timerPWMIInit =
{
    .enable= true,
    .debugRun=true,
    .prescale=timerPrescale1,
    .clkSel=timerClkSelHFFPerClk,
    .fallAction=timerInputActionNone,
    .riseAction=timerInputActionNone,
    .mode=timerModeUp,
    .dmaClrAct=false,
    .quadModeX4=false,
    .oneShot=false,
    .sync=false,
};

// pwm configuration for timer3.
TIMER_Init(TIMER3, &timerPWMIInit);

```

פונקציות

Listen

פסיקות:

בתוכנית יש שימוש רק בפסיקות חיצונית:

- PB0 משמש להתחלה פעולת המערכת.
- PB1 משמש להפסקת פעולה המערכת.

פונקציות הפעלה למדידת המרחק

ע"י שימוש בPCA ויכולת capture בעלייה ובירידה של אות echo , נמצא את הזמן שבין העלייה לירידה של האות במיקרו שניות:

```
uint32_t triggerNcapture_back();
uint32_t triggerNcapture_front();
```

בנוסף, ע"י שימוש בנוסחה עם מהירות הקול נקבל את המרחק :

```
new_distance = (triggerNcapture() / 58);
```

בדיקה האם המרחק השתנה:

```
bool check_distance_change(int new_val, int old_val);
```

פונקציות להפעלת הזזום

פונקציות להפעלת הזמן ולקביעת טון זזום לפי המרחק.

```
int buzzer_ton_state(int distance_mesasured);
void toggle_buzz();
```

חזר לתוכן עניינים

```
void waitingNbuzzertOn(int ton_state);
int power_of2(int val);
void alaram_func(bool emergency_f, bool emergency_b);
```

שליחה התראת קירבה ל-BT

פונקציית שימושת בערוץ תקשורת UART וודוחפת לבלווטו - כלומר שידור אות התראה בערוץ תקשורת BLUETOOTH לפלאפון.

```
void alaram_func(bool on_off);
bool alaram_state(int distance_mesauered);
```

בקר EFM32WG:ARM

פונקציות אתחול:

```
void Device_Init(void);
void Set_Clock_Gpio(void);
void PWM_Init(void);
void UART_Init(void);
void Engine_Init(void);
```

פונקציית הדלקה/כיבוי:

גלי נאות, כפטור 0 למשעה נלחץ בברker 1 8051 והאות שלו מעובר ישירות לARM שמאפשר לו תחילת עבודה. הרעיון שמאחורי זה, הוא שלא ניתן להתחיל נסעה כל עוד החישנים מכובדים. כפטור 1 תפקידו לכבות את החישנים וכן גם את מערכת ניהול המנוע. (גם הוא נלחץ ב8051) כפטור 1 סופה עצירה של המנועים במקום.

```
bool Button_0();
bool Button_1();
```

פונקציות בקרה

משמשות להגדרת מצב העבודה הבא של המנועים. מתבצע בפועל דרך שליטה מרוחק על ידי המשמש או דרך התערבות של החישני מרחוק.

```
uint8_t bluetooth_handler(uint8_t state);
uint8_t obstacle_detection(uint8_t state);
```

פקודות תנועה:

הפונקציות שמשנות את המצב הנוכחי של המנועים לפי המצב הבא שהוגדר על ידי פונקציות הבקרה קודם לכך. ניתן לשנות את הפעולה לנסעה קדימה/ אחורה/ שמאלה/ימינה/עצירה וגם לשנות את מהירות הנסעה.

```
void Engine_driver_Control(uint8_t state);
void PWN_Change_dutycycle(uint8_t speed_state);
void Move_forward();
void Move_back();
void Move_right();
void Move_left();
void Stop();
```

הסברים נוספים לשיקולי התכנון

ניתוח פולס ECHO

אופן מדידת אורך זמן פולס ECHO על ידי המערכת:

לאחר שליחת אות *trigger* לחישן אולטרא סוני (מד מרחוק), המערכת נכנסת במצב המתנה כאשר ברקע רץ מונה *PCA* - המערכת ממתינה לזיהוי עליית אות *ECHO*.

ברגע הזיהוי מתבצע *capture* לערך של מונה *PCA* והמערכת שומרת את הערך ברגיסטר העלייה, תוך שמקביל מונה *PCA* ממשיך לרוץ.

cutת המערכת ממתינה לזיהוי ירידת ה*echo*. ברגע זיהוי הירידה מתבצע שוב *capture* לערך של המונה והמערכת שומרת את הערך ברגיסטר הירידה.

בשלב זה מפסיקים את פעולת המונה.

המערכת כעת מבצעת חישוב לאורך הפלס, ב-2 אופנים שונים כתלות במקרה:

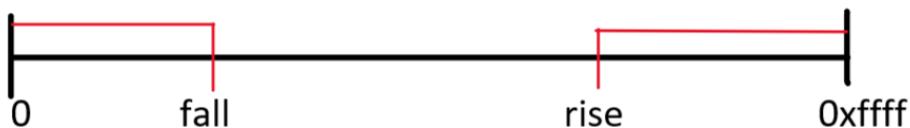
1. במקרה הסטנדרטי - ערך המונה בירידת האות גדול מערך המונה בעליית אותה, וכן גם ערכי הרגיסטרים שלהם בהתאם וערךו של רגיסטר העלייה גדול משל רגיסטר הירידה:



לכן אורך האות (המסומן באדום) הוא ההפרש בין רגיסטר הירידה לרגיסטר העלייה:

$$ECHO_{count} = REG_{PCA_{FALL}} - REG_{PCA_{RISE}}$$

2. במקרה השני - אם ערכו של רגיסטר העלייה גדול משל רגיסטר הירידה - משמע שהמדידה התחליה בשלב מתקדם של מונה PCA לקריאת סוף יcollת הספירה שלו, מושך לפניו שהוא התאפס והחל לספר מחדש.



במקרה זה חישוב המונה הוא סכום הפרש בין הגודל המקסימלי של המונה, לרגיסטר הירידה עם רגיסטר העלייה:

$$ECHO_{count} = 0xffff - REG_{PCA_{RISE}} + REG_{PCA_{FALL}}$$

חישוב הזמן

בשני המקרים, לאחר רגיסטר PCA עולה ב-1 בכל פעם שעובר זמן מהזור 1, התוצאה שהתקבלת מעשה מיצגת את מספר מחזורי השעון הספיק המונה לספר עד שבוצע ה *capture*.

על מנת לקבל את התוצאה ביחידות זמן ניקח את הגודל הזה ומכפילים בזמן המוחזר של טימר PCA.

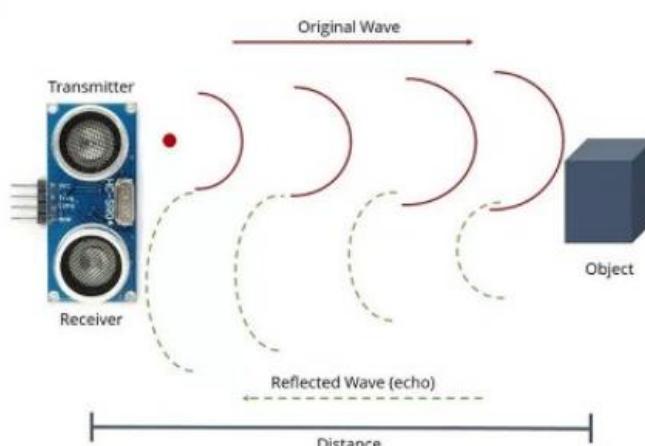
$$ECHO_{time} = ECHO_{count} \cdot \text{period time} [\mu\text{sec}]$$

חישוב המרחק

כעת נשמש בנוסחה הפיזיקלית:

$$Distance = time \cdot velocity$$

לצורך חישוב המרחק שעובר אות ECHO מרגע שעזב את החישין, פגע בעצם במרחב והוזר בחזרה אל החישין, כתלות ב מהירות האות שהיא מהירות הקול.



נשים לב שהאות בمسעו הולך וחזור למעשה עבר את המרחק *distance* פעמיים, לכן נחלק את התוצאה פי 2.

$$Distance = \frac{time \cdot velocity}{2}$$

$$\begin{aligned} Distance [m] &= \frac{ECHO_{time} [\mu sec] \cdot Speed\ of\ Sound\ in\ air \left[\frac{m}{sec} \right]}{2} \\ &= \frac{ECHO_{time} [\mu sec] \cdot 343 \left[\frac{m}{sec} \right]}{2} = ECHO_{time} [\mu sec] \cdot \frac{343}{2} \left[\frac{m}{sec} \right] \end{aligned}$$

נעבור ליחידות מרחק בcm:

$$\begin{aligned} distance [cm] &= ECHO_{time} [\mu sec] \cdot \frac{343}{2} \left[\frac{cm}{c\ sec} \right] = ECHO_{time} [\mu sec] \cdot \frac{343}{2} \cdot \frac{1}{c} \left[\frac{cm}{sec} \right] \\ &= ECHO_{time} [\mu sec] \cdot \frac{343}{2} \cdot \frac{1}{c} \left[\frac{cm}{\mu sec} \right] \approx ECHO_{time} [\mu sec] \cdot \frac{1}{58} \left[\frac{cm}{\mu sec} \right] \end{aligned}$$

$$distance [cm] \approx \frac{ECHO_{time}}{58} \left[\mu sec \cdot \frac{cm}{\mu sec} \right]$$

לפי הנוסחה שקיבילנו - נקבל את המרחק שנמדד בסנטימטר כאשר נחלק מדידת הזמן של האות פי 58. לכן גם נודא לקחת את זמן המחזור במיקרו שניות.

הערה: הרגישות שלנו תהיה לפי ס"מ, ויהיה עיגול מטה למספרים שלמים (משתמשים במשתנה *int*). כמובן, עברו מרחק של 45.8 ס"מ נראה פשוט 45 ס"מ.

איך מייצרים פולס טריגר באורך המתאים?

מגדירים טימר שיגולש בדיק למשך 10 מיקרו שניות.
מגדירים פונקציית טריגר שתפקידה הוא:

1. לאפס את הטיימר.
2. להציג יציאה מתאימה ל'1' לוגי.
3. להפעיל את הטיימר.
4. להמתין עד הגלישה.
5. לאחר הגלישה להציג את היציאה ל'0' לוגי.
6. חזרת דגל הגלישה בסוף התהילה.

ונרץ יצרנו פולס טריגר באורך המתאים.

איך מחשבים מחלק מתח לרجل ECHO ?

מחברים את החיבור באופן הבא:

1. חיבור מתח הזנה VCC.
2. חיבור אדמה GND.
3. חיבור טריגר.
4. חיבור ECHO לסקופ מדידה.

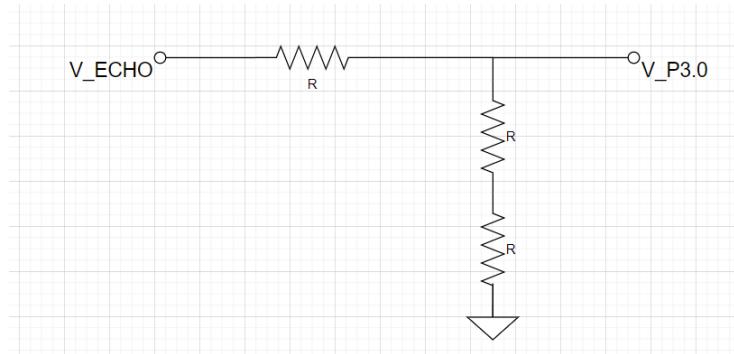
ומבצעים את הפעולות הבאות לאחר מכן:

1. נתונים טריגר לחישון.
2. מודדים את המתח שיוצא ברגל ECHO.
3. מגלים שהמתוך 5 וולט.
4. בודקים במדריך המשמש מה המתוך הרצוי שיש לתת בכניסות.
5. מגלים שהוא 3.3 וולט.
6. מבצעים מחלק מתח בהתאם באופן הבא:

$$\begin{aligned} \nu_{p3.0} &= \nu_{echo} * \nu_{p3.0} \\ \frac{\nu_{p3.0}}{\nu_{echo}} &= \frac{3.3}{5} = 0.66 \end{aligned}$$

ולכן נסיק לקחת 3 נגדים זהים. וnochbar באופן הבא :

[חזר לתopic עכינים](#)



מימוש התוכנית

EFM8UB2:8051

Main

```
50 // -----
51 int main(void)
52 {
53     uint32_t distance_mesauered_b,distance_mesauered_fr;
54     uint32_t new_distance_b,new_distance_fr;
55     uint32_t ton_state_f,ton_state_b;
56     bool b_change,f_change;
57     bool emeregency_b,emeregency_f;
58     enter_DefaultMode_from_RESET (); // Call hardware initialization routine
59     PCA0CN0_CCF0=0;
60     PCA0CN0_CCF1=0;
61     distance_mesauered_b=0;
62     distance_mesauered_fr=0;
63     emeregency_b=false;
64     emeregency_f=false;
65     b_change=false;
66     f_change=false;
67     ton_state_f=0;
68     ton_state_b=0;
69
70
71 }
```

חוור לתוכן עניינים

```
72         alaram_func(emergency_b, emergency_f); // send SOS signs & turn on buzzer OR turn off
73     }
74     if (button1_pressed == 1) // stop the process and avoid the waiting loop
75     {
76         alert_signal_b = 0;
77         alert_signal_f = 0;
78         buzzer = 0;
79         button1_pressed = 0;
80     } | break;
81     }
82     if(ton_state_f>ton_state_b)
83         waitingNbuzzerTon(ton_state_f);
84     else
85         waitingNbuzzerTon(ton_state_b);
86     }
87     }
88     if (button1_pressed == 1) //in case the you outside the loop
89     {
90         buzzer = 0;
91         button1_pressed = 0; //avoid false button1_pressed when it will start a new process
92     }
93 }
94 }
95 }
```

Interrupts

```
5 #include "general.h"
6 extern uint32_t button0_pressed;
7 extern uint32_t button1_pressed;
8 extern uint32_t in_process;
9 //-----
10 // INT0_ISR
11 //-----
12 // INT0 ISR Content goes here. Remember to clear flag bits:
13 // TCON::IE0 (External Interrupt 0)
14 //-----
15 SI_INTERRUPT (INT0_ISR, INT0_IRQn)
16 {
17     TCON_IE0 = 0;
18     button0_pressed = 1;
19     button1_pressed = 0;
20     in_process = 1; // to indicate start periodic process
21     button0_out=1;
22     button1_out=0;
23 }
24 }
```

חוור לתוכן עניינים

```

25 //-----
26 // INT1_ISR
27 //-----
28 // INT1 ISR Content goes here. Remember to clear flag bits:
29 // TCON::IE1 (External Interrupt 1)
30 //-----
31 SI_INTERRUPT (INT1_ISR, INT1_IRQn)// button1 - stop priodic mesure process
32 {
33   TCON_IE1 = 0;
34   button1_pressed = 1;
35   button0_pressed = 0;
36   in_process = 0; // to indicate end periodic proccess
37   button0_out=0;
38   button1_out=1;
39 }
40
41
--
```

general.h

קובץ קוד שמכיל הגדרת קבועים ועוד פעולות שנעשות טרם הקומpileציה.

```

1 // Owner : Oren Danilov ID: 203824545
2 // Saar Gozlan ID: 204188403
3 // Project: efm8_8051_Sensor_control
4 //=====
5
6 #ifndef GENERAL_H
7 #define GENERAL_H
8 //-----
9 // Includes and defines
10 //-----|
11 #include <SI_EFM8UB2_Register.Enums.h> // SFR declarations
12 #include "InitDevice.h"
13 #include "bsp.h"
14 #include "disp.h"
15 #include "tick.h"
16 #include "render.h"
17 #include <string.h>
18 #include <stdio.h>
19 #include <stdlib.h>
20
21 //-----
22 // defines
23 //-----
24 #define TRIGGER_DELAY 0.5 // between trigger pulses [m sec]
25 #define BAUD_RATE 9600 // UART frequency [1/sec]
26 #define MINIMAL_DISTANCE 15 // [cm]
27 #define ALARM_STRING "SOS!"
28
29 //-----
30 // pins definition
31 // -----
32 SI_SBIT (button0 , SFR_P0, 2); // P0.2
33 SI_SBIT (button1 , SFR_P0, 3); // P0.3
34 SI_SBIT (button0_out , SFR_P2, 4); // P2.4
35 SI_SBIT (button1_out , SFR_P2, 5); // P2.5
36 SI_SBIT (trigger_signal , SFR_P3, 0); // P3.0
37 SI_SBIT (echo_signal , SFR_P3, 1); // P3.1
38 SI_SBIT (trigger_signal_b , SFR_P3, 2); // P3.2
39 SI_SBIT (echo_signal_b , SFR_P3, 3); // P3.3
40 SI_SBIT (alert_signal_b , SFR_P3, 4); // P3.4
41 SI_SBIT (alert_signal_f , SFR_P3, 5); // P3.5
42 SI_SBIT (buzzer , SFR_P1, 6); // P1.6
43 SI_SBIT (UARTtx_BTrx , SFR_P0, 4); // P0.4
--
```

```
52//-----
53 // functions signature
54 //
55 uint32_t triggerNcapture_back();
56 uint32_t triggerNcapture_front();
57 void waitingNbuzzerTon(uint32_t ton_state);
58 bool check_distance_change(uint32_t new_val, uint32_t old_val);
59 void alarm_func(bool emeregency_f, bool emeregency_b);
60 bool alaram_state(uint32_t distance_mesauered);
61 uint32_t buzzer_ton_state(uint32_t distance_mesauered);
62 void toggle_buzz();
63 uint32_t power_in2(uint32_t val);
64//void distance_int_to_text(uint32_t distance_mesauered,char* str);
65 //void print_LCD(uint32_t distance_mesauered_f,uint32_t distance_mesauered_b);
66 #endif
```

פונקציות

```
97//-----
98 // functions
99 //
100 uint32_t power_of2(uint32_t val) // 2^ton_state
101 {
102     uint32_t x=1;
103     while(val!=0){
104         x*=2;
105         val--;
106     }
107     return x;
108 }
```

```
110 void waitingNbuzzerTon(uint32_t ton_state)
111 {
112     uint32_t overflow_counter; // use Tbit_counter to count delay of 0.5 sec
113     uint32_t buzz_cnt; // use buzz_cnt to change the buzzer frequency
114     uint32_t scale;
115     uint32_t baude_rate_percent;
116     if(ton_state>0)
117         scale = power_of2(ton_state); // = 2^(ton_state)
118     else
119         scale=-1;
120     buzz_cnt=0;
121     overflow_counter=0;
122     TCON_TR1=1; // start timer 1
123
124     baude_rate_percent=0.125*BAUD_RATE;
125     while(overflow_counter<=baude_rate_percent)//wait 0.0625sec
126     {
127         if(ton_state>=1 && ton_state<=4){
128             if(buzz_cnt == (baude_rate_percent/scale)){ //toggle every 0.5sec/scale
129                 toggle_buzz();
130                 buzz_cnt=0;
131             }
132         }
133     }
```

חוור לתוכן עניינים

```
133     while(TCON_TF1==0); // wait for timer1 overflow (52u sec)
134     TCON_TF1 = 0; // clear overflow flag
135     overflow_counter++;
136     buzz_cnt++;
137 }
138
139 TCON_TR1=0; // stop timer 1
140 delay_end=1;
141 }
142
143 void distance_int_to_text(uint32_t distance_mesaured, char* str)...
144 void alaram_func(bool emeregency_b, bool emeregency_f)
145 {
146     uint32_t i=0;
147     if (emeregency_b == true || emeregency_f == true) // send SOS signs & turn on buzzer
148     {
149         if(emeregency_b&&emeregency_f){
150             alert_signal_b = 1;
151             alert_signal_f = 1;
152         }
153         -
154
155         else if (emeregency_b == true ){
156             alert_signal_b = 1;
157             alert_signal_f=0;
158         }
159         else if (emeregency_f == true){
160             alert_signal_b = 0;
161             alert_signal_f = 1;
162         }
163         buzzer = 1;
164
165         //use UART0 to send alarm to the user:
166         SCON0_TI=0;
167         TCON_TR1=1;
168         for(;i<4;i++)
169         {
170             SBUF0=ALARM_STRING[i];
171             while(SCON0_TI==0);
172             SCON0_TI=0;
173         }
174         TCON_TR1=0;
175     }
176
177     else{ //turn off buzzer
178         buzzer = 0;
179         alert_signal_b = 0;
180         alert_signal_f = 0;
181     }
182 }
```

חזר לתוך עניינים

```
215     }
216 }
217
218 uint32_t triggerNcapture_back()
219 {
220     uint32_t REG_PCA_RISE=0;
221     uint32_t REG_PCA_FALL=0;
222     uint32_t delta_reg=0;
223     uint32_t i=0;
224
225     // ---- trigger ----
226
227     // manual reload value
228     TH0 =0xff;
229     TL0 =0xf1;
230
231     P3_B0 = 1; // turn on one tact before start timer to ensure 10us pulse minimum
232     for(;i<=1;i++){
233         TCON_TR0 = 1; // start timer 0
234         while (TCON_TFO == 0); // wait for 10usec
235         TCON_TFO = 0; // clear overflow flag
236     }
237     P3_B0=0;
238     TCON_TFO = 0; // clear overflow flag
239     TCON_TR0= 0; // stop
240
241
242     ...
243
244     // ---- ECHO capture ----
245
246     // reset pca0 register value
247     //get ready to catch echo signal rise
248     PCA0CPM0=PCA0CPM0_CAPP__ENABLED;
249
250     if(PCA0CN0_CF)
251         PCA0CN0_CF=0;
252     if(PCA0CN0_CCF0)
253         PCA0CN0_CCF0=0;
254     PCA0CN0_CR = 1; // start PCA0
255     while(PCA0CN0_CCF0==0);//waiting echo rise
256     REG_PCA_RISE=PCA0CPH0*0x100 + PCA0CPL0;
257     PCA0CN0_CCF0=0;//clear the flag
258     PCA0CN0_CF=0;
259     //get ready to catch echo signal fall
260     PCA0CPM0=PCA0CPM0_CAPN__ENABLED;
261     while(PCA0CN0_CCF0==0);//waiting echo fall, delay_end is for security
262
263     REG_PCA_FALL = PCA0CPH0*0x100 + PCA0CPL0; // save how many clocks the ECHO signal run
264     PCA0CN0_CR = 0; // stop PCA0
265     PCA0CN0_CCF0 = 0;
266     if(PCA0CN0_CF)
267         PCA0CN0_CF=0;
268
269     // PCA0 clock period is 0.5usec then it can count for about (0xffff)*0.5usec = 32.7msec
270     // we know that the max echo time is about 36ms > 32.7msec => the pca clock start again
```

חוור לתוכן עניינים

```
269     if(REG_PCA_FALL<REG_PCA_RISE)
270         delta_reg= 0xffff - REG_PCA_RISE + REG_PCA_FALL;
271     else
272         delta_reg= REG_PCA_FALL- REG_PCA_RISE;
273
274     // PCA0 clock period is 0.5usec
275     delta_reg *= 0.5; //echo_time[usec]
276     return delta_reg;
277 }
278
279 uint32_t triggerNcapture_front()
280 {
281     uint32_t REG_PCA_RISE=0;
282     uint32_t REG_PCA_FALL=0;
283     uint32_t delta_reg=0;
284     uint32_t i=0;
285
286     // ---- trigger ----
287
288     // manual reload value
289     TH0 =0xff;
290     TL0 =0xf1;
291
292     trigger_signal_b = 1; // turn on one tact before start timer to ensure 10us pulse minimum
293     for(;i<=1;i++){
294         TCON_TR0 = 1; // start timer 0
295         while (TCON_TF0 == 0); // wait for 10usec

296         TCON_TF0 = 0; // clear overflow flag
297     }
298     trigger_signal_b=0;
299     TCON_TF0 = 0; // clear overflow flag
300     TCON_TR0= 0; // stop
301
302
303     // ---- ECHO capture ----
304
305     // reset pca0 register value
306     //get ready to catch echo signal rise
307     PCA0CPM1=PCA0CPM1_CAPP__ENABLED;
308
309     if(PCA0CN0_CF)
310         PCA0CN0_CF=0;
311     if(PCA0CN0_CCF1)
312         PCA0CN0_CCF1=0;
313     PCA0CN0_CR = 1; // start PCA0
314     while(PCA0CN0_CCF1==0); //waiting echo rise
315     REG_PCA_RISE=PCA0CPH1*0x100 + PCA0CPL1;
316     PCA0CN0_CCF1=0; //clear the flag
317     PCA0CN0_CF=0;
318     //get ready to catch echo signal fall
319     PCA0CPM1=PCA0CPM1_CAPN__ENABLED;
320     while(PCA0CN0_CCF1==0); //waiting echo fall, delay_end is for security
321
322     REG_PCA_FALL = PCA0CPH1*0x100 + PCA0CPL1; // save how many clocks the ECHO signal run
```

חזר לתוך עניינים

```
323     PCA0CN0_CR = 0; // stop PCA0
324     PCA0CN0_CCF1 = 0;
325     if(PCA0CN0_CF)
326         PCA0CN0_CF=0;
327
328     // PCA0 clock period is 0.5usec then it can count for about (0xffff)*0.5usec = 32.7msec
329     // we know that the max echo time is about 36ms > 32.7msec => the pca clock start again
330     if(REG_PCA_FALL<REG_PCA_RISE)
331         delta_reg= 0xffff - REG_PCA_RISE + REG_PCA_FALL;
332     else
333         delta_reg= REG_PCA_FALL- REG_PCA_RISE;
334
335     // PCA0 clock period is 0.5usec
336     delta_reg *= 0.5; //echo_time[usec]
337     return delta_reg;
338 }
339
340 bool check_distance_change(uint32_t new_val, uint32_t old_val){
341     if(new_val!=old_val)
342         return true;
343     return false;
344 }
345
346 void toggle_buzz() // turn on and of the
347 {
348     buzzer=!buzzer;
349 }
```

```
351 uint32_t buzzer_ton_state(uint32_t distance_mesauered){
352     uint32_t state;
353     if(distance_mesauered>=30)
354         state=0;
355     else if (distance_mesauered<30 && distance_mesauered>=20)
356         state=1;
357     else if(distance_mesauered<20 && distance_mesauered>=15)
358         state=2;
359     else if(distance_mesauered<15 && distance_mesauered>=10)
360         state=3;
361     else if(distance_mesauered<10 && distance_mesauered>=5)
362         state=4;
363     else if(distance_mesauered<5)
364         state=5;
365     else // dont care
366         state=0;
367     return state;
368 }
369
370 bool alaram_state(uint32_t distance_mesauered)
371 {
372     if(distance_mesauered<MINIMAL_DISTANCE)
373         return true;
374     return false;
375 }
376
```

main

```
32 int main() {
33     CHIP_Init();
34     Device_Init();
35     uint8_t state,next_state ,speed_state ,speed_next_state ;
36     state='S'; // initlize engines state to stop
37     next_state='S';
38     speed_state='0';
39     speed_next_state='0';
40     while (1){
41         if(Button_0()){
42             while(1){
43                 uint8_t tmp;
44                 tmp=bluetooth_handler(state); // USER INTERFACE CONTORL
45                 if(tmp >='0' && tmp <= '9')
46                     speed_next_state=tmp;
47                 else if(tmp >='A' && tmp <= 'Z')
48                     next_state=tmp;
49
50                 if(state!='S')//USE THE MACHINE SENSEORS
51                     next_state=obstacle_detection(next_state);
52
53                 if(Button_1()){// return to idle
54                     next_state='S';
55                     state=next_state;
56                     Engine_driver_Control(state);//FORCE STOP
57                     break;
58                 }
59                 if(next_state == 'R' || next_state == 'L')
60                     speed_next_state='2'; // use low speed for safety when turn right or left
61                 if((speed_next_state!=speed_state) && (speed_next_state!=0)){
62                     speed_state=speed_next_state;
63                     PWN_Change_dutycycle(speed_state);
64                 }
65
66                 if((next_state!=state) && (next_state!=0)){
67                     state=next_state;
68                     Engine_driver_Control(state);
69                 }
70             }
71         }
72     }
73 }
```

init_device.c
קובץ איתחול למערכת

חוור לתוכן עניינים

```
4 #include "em_device.h"
5 #include "em_chip.h"
6 #include "em_cmu.h"
7 #include "em_timer.h"
8 #include "em_gpio.h"
9 #include "em_usart.h"
10
11 #define OUT_FREQ 1000 // timer3(PWM) Desired frequency [Hz].
12 #define DUTY_CYCLE 75 // PWM Duty Cycle %
13 #define timer3_FREQ 28000000 // timer3 system frequency [Hz]
14
15 void Device_Init(void);
16 void Set_Clock_Gpio(void);
17 void PWM_Init(void);
18 void UART_Init(void);
19 void Engine_Init(void);
20 void Stop();
21 void Device_Init(void) {
22     Set_Clock_Gpio();
23     Engine_Init();
24     UART_Init();
25 }
26
27 void Set_Clock_Gpio(void){
28     CMU_HFRCOBandSet(cmuHFRCOBand_28MHz); // Set HFRCOCTRL (High Freq. RCOsc.) to
29     CMU_ClockEnable(cmuClock_GPIO, true); // Enable GPIO peripheral clock
30 }
31 void PWM_Init(void){
32     CMU_ClockEnable(cmuClock_TIMER3, true); // enable Timer3 clock.
33     GPIO_PinModeSet(gpioPortE, 2, gpioModePushPull, 0); // Configure port E pin 2
34
35     uint32_t top;
36     uint32_t compare_val;
37     top=timer3_FREQ/OUT_FREQ;
38     compare_val = ((timer3_FREQ*1.0)/(OUT_FREQ*1.0) )*(DUTY_CYCLE/100.0);
39
40     TIMER_TopSet (TIMER3,top); // set top timer 3.
41     TIMER_CounterSet(TIMER3, 0); // start counter from 0 up-count
42     TIMER_CompareSet(TIMER3, 2, compare_val); // set cc2 compare value 25 duty
43     TIMER_CompareBufSet(TIMER3, 2, compare_val); // set cc2 compare buffer value 2
44     TIMER3->ROUTE = (1 << 16)| (1 << 2) ; // connect pwm output to port E pin 2.
```

חוור לתוך עניינים

```
46 // setup timer channel configuration for pwm.  
47 TIMER_InitCC_TypeDef timerCCInit =  
48 {  
49     .eventCtrl=timerEventEveryEdge,  
50     .edge=timerEdgeNone,  
51     .prsSel=timerPRSELCh0,  
52     .cufoa=timerOutputActionNone,  
53     .cofoa=timerOutputActionSet,  
54     .cmoa=timerOutputActionClear,  
55     .mode=timerCCModePWM,  
56     .filter=false,  
57     .prsInput= false,  
58     .coist=false,  
59     .outInvert=false,  
60 };  
61 //configuration for timer3 on channel 2.  
62 TIMER_InitCC(TIMER3, 2, &timerCCInit);  
63 // setup timer configuration for pwm.  
64 TIMER_Init_TypeDef timerPWMIInit =  
65 {  
66     .enable= true,  
67     .debugRun=true,  
68     .prescale=timerPrescale1,  
69     .clkSel=timerClkSelHFPerClk,  
70     .fallAction=timerInputActionNone,  
71     .riseAction=timerInputActionNone,  
72     .mode=timerModeUp,  
73     .dmaClrAct=false,  
74     .quadModeX4=false,  
75     .oneShot=false,  
76     .sync=false,  
77 };
```

חזר לתopic עניינים

```
78     // pwm configuration for timer3.
79     TIMER_Init(TIMER3, &timerPWMIInit);
80 }
81 void UART_Init(void){
82     CMU_ClockEnable(cmuClock_GPIO, true);
83     CMU_ClockEnable(cmuClock_UART0, true);
84
85     GPIO_PinModeSet(gpioPortE, 0, gpioModePushPull, 1); // U0_TX is push pull
86     GPIO_PinModeSet(gpioPortE, 1, gpioModeInput, 1); // U0_RX is input
87
88     // Start with default config, then modify as necessary
89     USART_InitAsync_TypeDef config = USART_INITASYNC_DEFAULT;
90
91     config.baudrate = 9600; // CLK freq is 1 MHz
92     config.autoCsEnable = true; // CS pin controlled by hardware, not firmware
93
94     config.enable = usartDisable; // Make sure to keep USART disabled until it's a
95     USART_InitAsync(UART0, &config);
96
97     // Set and enable USART pin locations
98     UART0->ROUTE = UART_ROUTE_CLKPEN | UART_ROUTE_CSPEN | UART_ROUTE_TXPEN
99             | UART_ROUTE_RXPEN | UART_ROUTE_LOCATION_LOC1;
100
101    USART_Enable(UART0, usartEnable);
102 }
103 void Engine_Init(void){
104     PWM_Init();
105     for(int i=0;i<=7;i++)
106         GPIO_PinModeSet(gpioPortD, i, gpioModePushPull, 0); // Configure port D pin
107
108     /* Configure port A pin 12 as button 0
109      Configure port A pin 13 as button 1
110      Configure port A pin 14 as alarm_Front */
111     for(int i=12;i<=14;i++)
112         GPIO_PinModeSet(gpioPortA, i, gpioModeInput, 0);
113
114     // Configure port B pin 15 as alarm_back
115     GPIO_PinModeSet(gpioPortB, 15, gpioModeInput, 0);
116
117     Stop();
118 }
```

תוצאות

תמונות וסרטוני הדוגמה לפעולת המערכת

הדוגמה להימנעות ממכתלים

<https://youtu.be/KgyjMnLCW2s>

הדגמה לשיליטה והיגוי ע"י המשתמש

https://youtu.be/Darm0xmn_ds

בסיום ניסיון בשטח

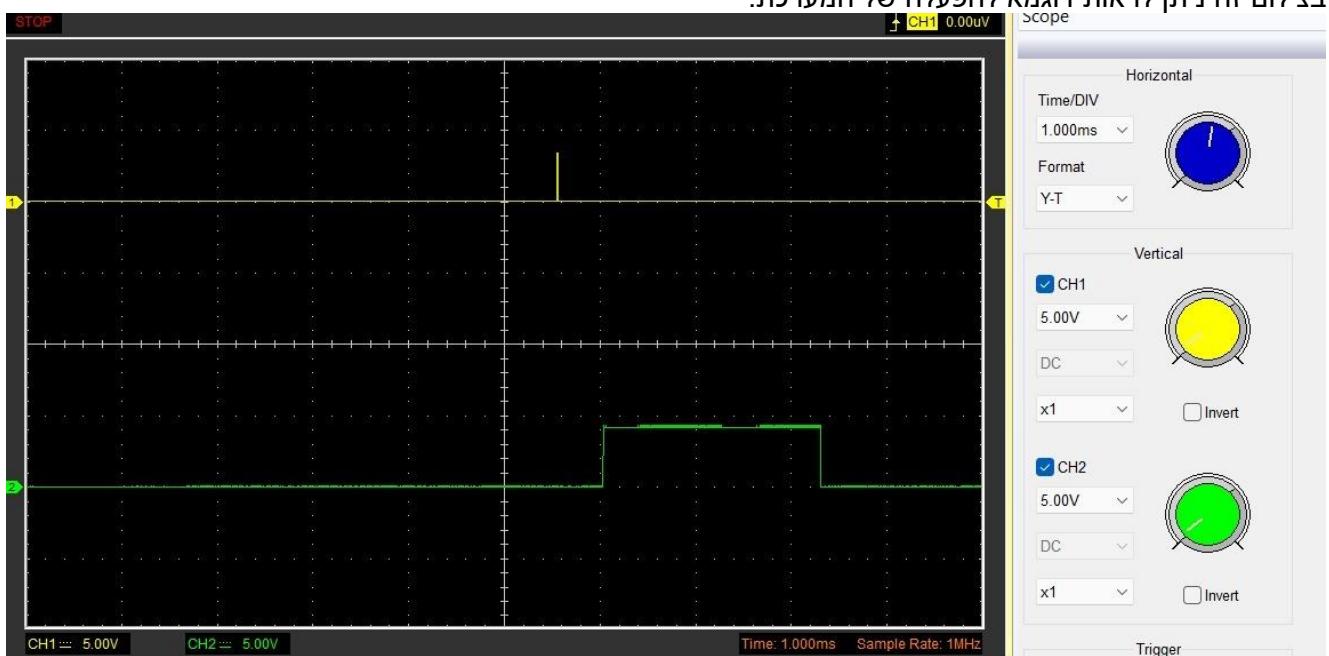
<https://youtu.be/SBY9tPuCl8s>





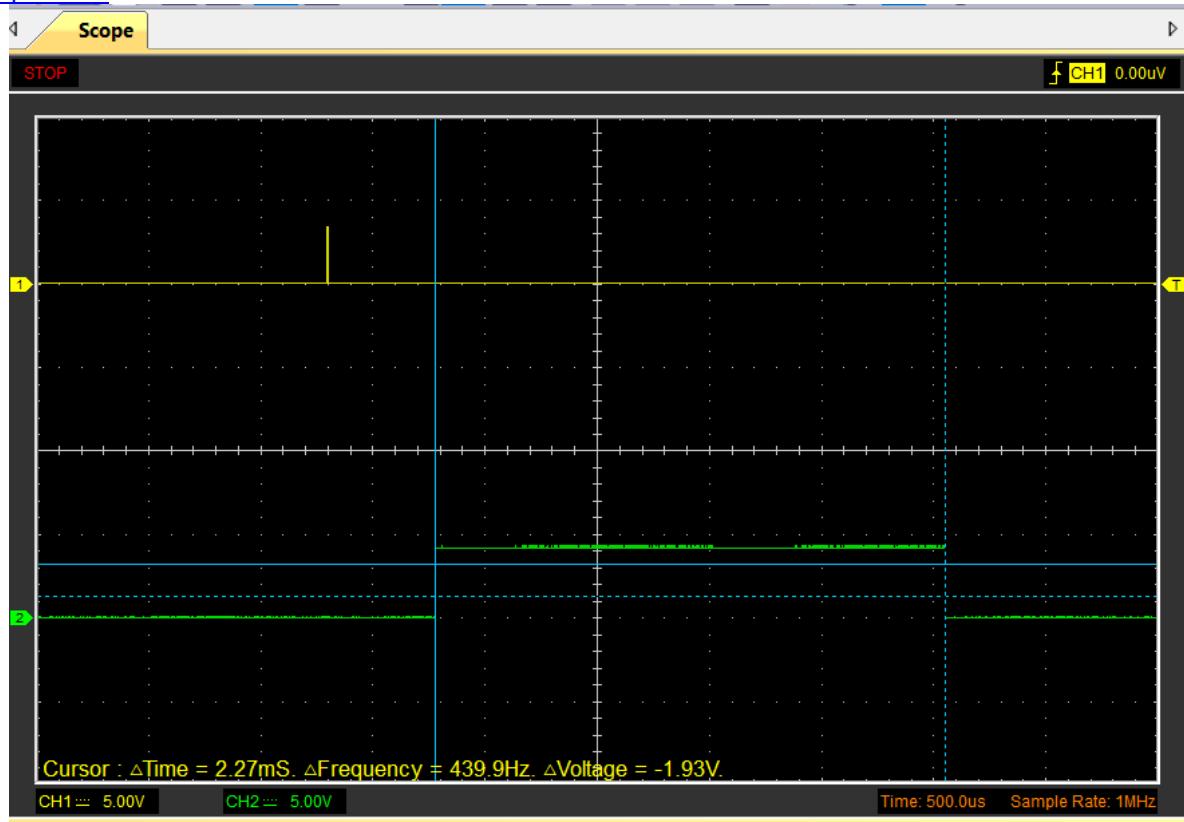
צילום ב scope של אות echo ואות trigger

בצלום זה ניתן לראות דוגמא להפעלה של המערכת:



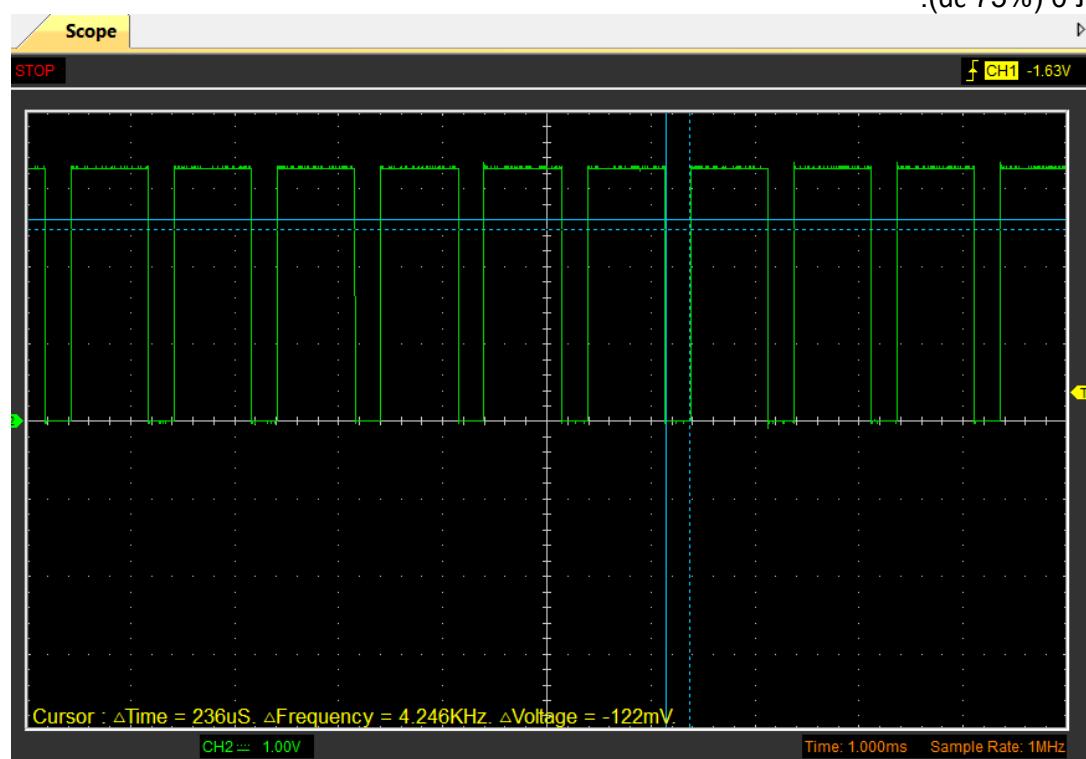
- אות trigger בצהוב.
- אות ECHO בירוק.

תמונה נוספת עם מדידה של אורך פולס ה-ECHO:

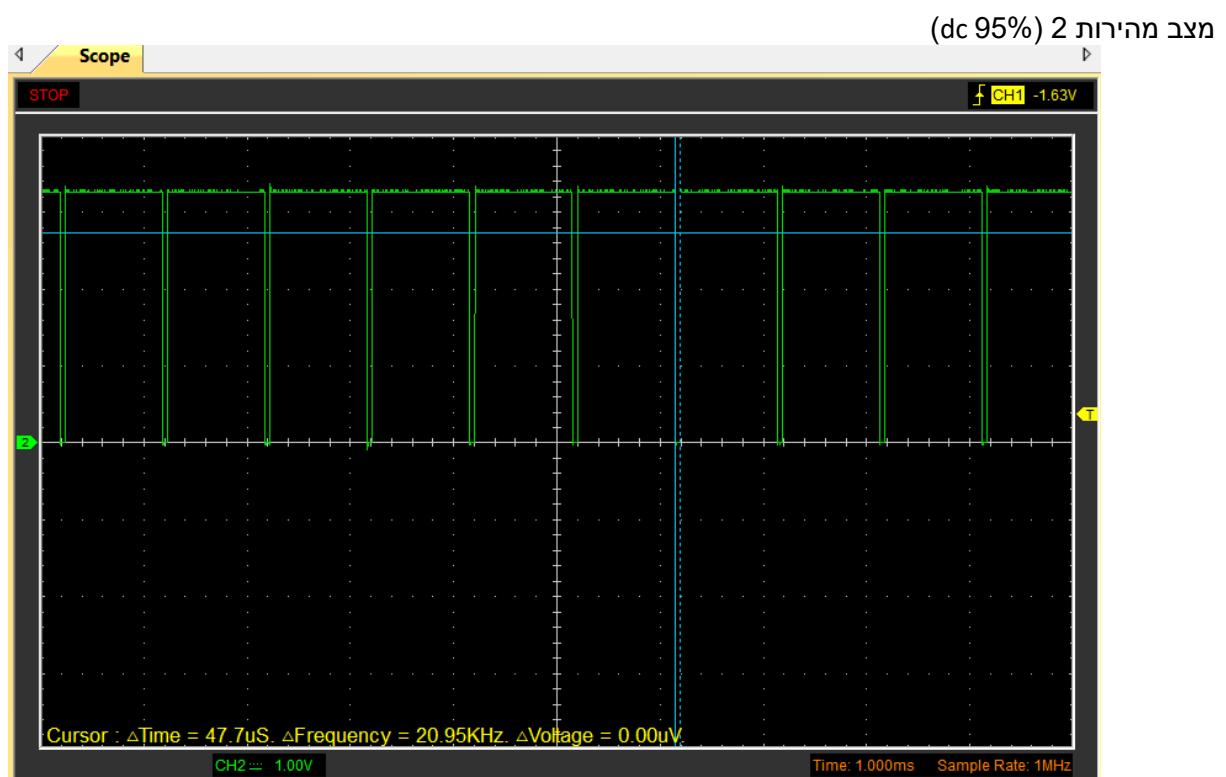
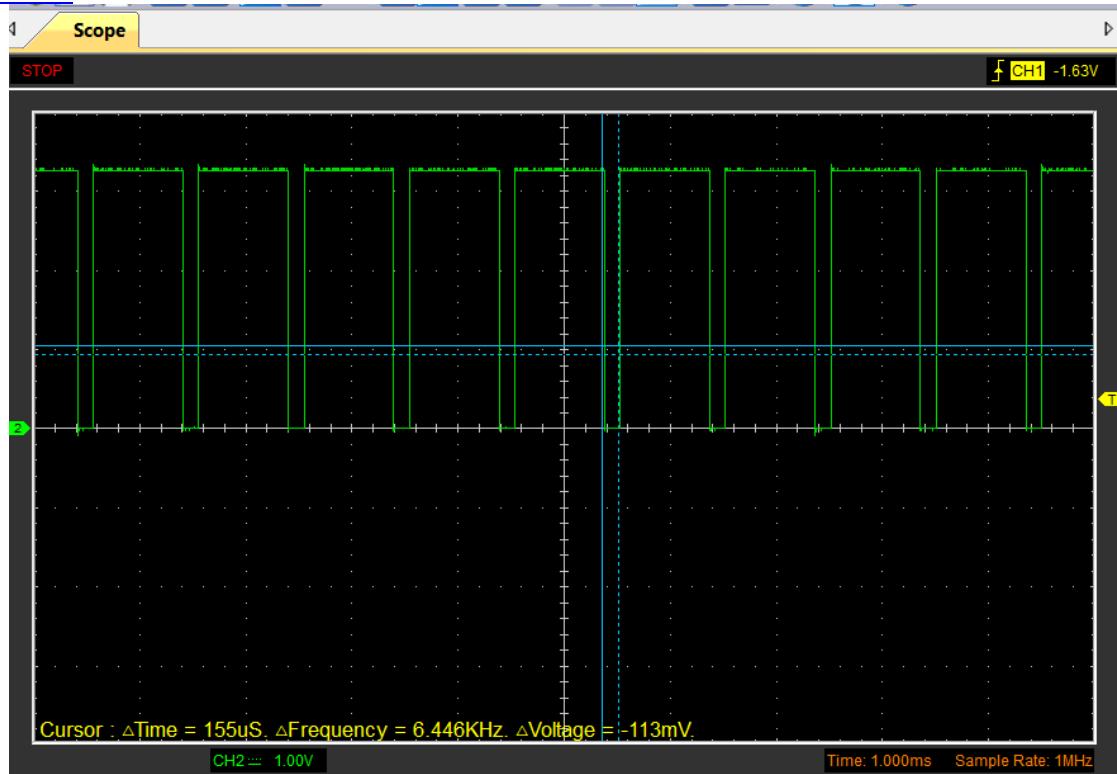


צילום ב scope של אות PWM עבור מסויימת:

: (dc 75%)



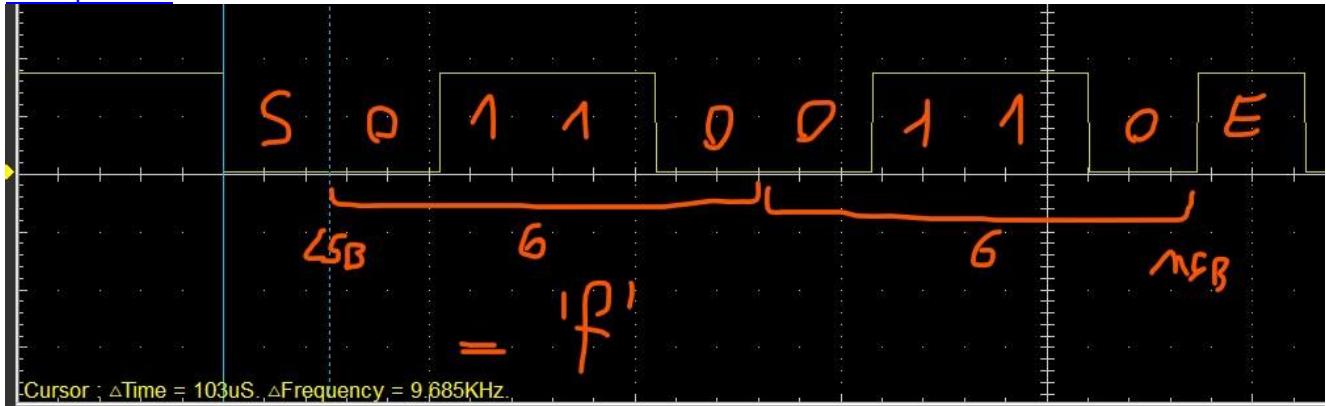
: (dc 85%)



ציילום ב scope של אות בקרה כפי שנקלט בטלפון הנייד דרך ערוץ BT :

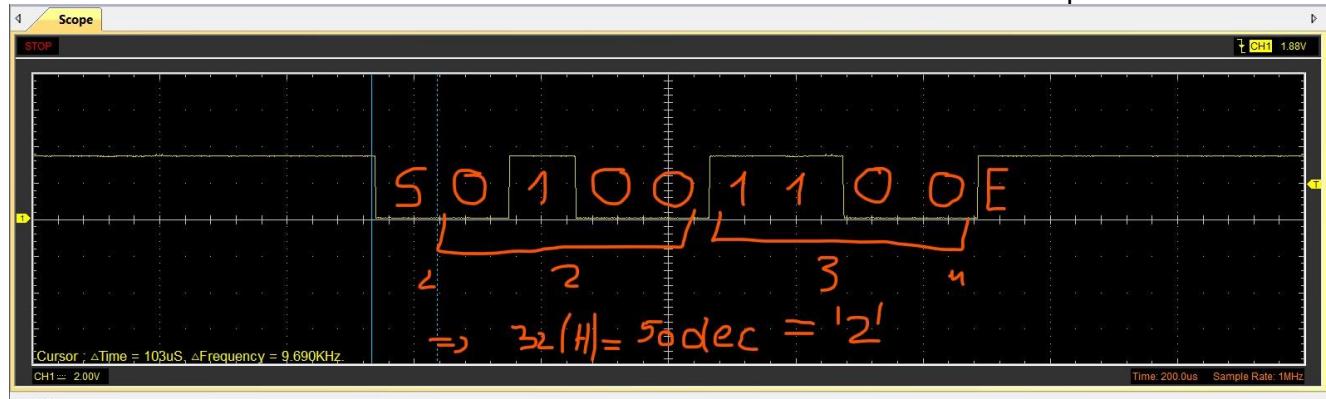
נכשלה לתפוס את פעולה UART באמצעות הסקופ בAKER ששולח אות בקרה לתנועת הרכיב קידמה - התו 'f' :

חזר לתוך עניינים



קיבלונו 66 בהקסה, שזה 102 בדצימלי.
ואכן לפי טבלת אסיקי התו 'f' הוא בעל ערך של 102.

ונסה שוב עברו את בקרה לשינוי מהירות סיבובי המנוע למצב מהירות 2 - התו '2':



קיבלונו 32 בהקסה, שזה 50 בדצימלי.
ואכן לפי טבלת אסיקי התו '2' הוא בעל ערך של 50.

Debugging

נראה דוגמא להציג תוכן הרגיסטרים וערך משתנים:
עלית דגל הגלישה של AT לאחר השידור

160 void alarm_func(bool on_off)	
161 {	
162 int i=0;	
163 if (on_off == true) // send SOS signs &	
164 {	
165 buzzer = 1;	
166 SCON0_TI=0;	
167 TCON_TR1=1;	
168 for(;i<4;i++){	
169 SBUFO=ALARM_STRING[i];	
170 while(SCON0_TI==0);	
171 SCON0_TI=0;	
172 }	
173 TCON_TR1=0;	
174 }	
175 else //turn off buzzer	
176 buzzer = 0;	
177 }	
178	
179 int triggerNcapture()	
239	
240 void print_LCD(int distance_mesasured)	
241 {	
242 uint8_t i;	
243 uint8_t Y;	
244 int row_number;	
245 char const_text[] = "the distance is: ";	
246 char cm_text[] = "cm" ;	
247 char distance_text[6]; // max: 300 , min	
248 char* str_vec[2];	
249	

Test Peripheral	Test Peripheral Registers
TIMER 0/1	TIMER 0/1 Registers
TIMER 2	TIMER 2 Registers
TIMER 3	TIMER 3 Registers
TIMER 4	TIMER 4 Registers
TIMER 5	TIMER 5 Registers
TIMER Setup	TIMER Setup Registers
UART 1	UART 1 Registers
UART 0	UART 0 Registers
SBUFO	UART0 Serial Port Data Buffer
SCON0	UART0 Serial Port Control
SMODE	0x0 - 8-BIT (8-bit UART with Vari... Serial Port 0 Operation Mode
MCE	0x0 - MULTI_DISABLED (Ignore le... Multiprocessor Communication E...
REN	0x0 - RECEIVE_DISABLED (UART0 ... Receive Enable
TB8	0x0 - CLEARED_TO_0 (In Mode 1, ... Ninth Transmission Bit
RB8	0x0 - CLEARED_TO_0 (In Mode 0, ... Ninth Receive Bit
TI	0x1 - SET (UART0 transmitted a by... Transmit Interrupt Flag
RI	0x0 - NOT_SET (A byte of data has... Receive Interrupt Flag
USB Library	USB Library Registers
USB	USB Registers
Supply Monitor	Supply Monitor Registers
Voltage Reference	Voltage Reference Registers
Voltage Regulators	Voltage Regulators Registers
External RAM	External RAM Registers
Name : SBUFO	
Hex:0x0	
Decimal:0	

חזר לתוכן עניינים

```
100 // TCON_TF1==0) || CON_STATUS==1) && CON_STATUS!=4){\n101     if(buzz_cnt == (BAUD_RATE/scale)){\n102         toggle_buzz();\n103         buzz_cnt=0;\n104     }\n105     while(TCON_TF1==0); // wait for timer\n106     TCON_TF1 = 0; // clear overflow flag\n107     overflow_counter ++;\n108     buzz_cnt++;\n109 }\n110\n111 TCON_TR1=0; // stop timer 1\n112 delay_end=1;\n113 }\n114\n115 void distance_int_to_text(int distance, char
```

עליית דגל הגלישה של טימר 1:		TIMER Registers
		TIMER Setup Registers
	CKCON0	Clock Control 0
	CKCON1	Clock Control 1
	TCON	Timer 0/1 Control
	TF1	Timer 1 Overflow Flag
	TR1	Timer 1 Run Control
	TF0	Timer 0 Overflow Flag
	TR0	Timer 0 Run Control
	IE1	External Interrupt 1
	IT1	Interrupt 1 Type Select
	IE0	External Interrupt 0
	IT0	Interrupt 0 Type Select
>	TMOD	Timer 0/1 Mode

```

1    _L1:    MOV A, @R0
2    ADD A, B
3    MOV B, A
4    INC R0
5    JNE _L1, _L2
6
7    // ---- trigger ----
8
9    // manual reload value
10   TH0 =0xff;
11   TL0 =0xf1;
12
13   P3_B0 = 1; // turn on one tact before s
14   for(i<1;i++){
15       TCON_TR0 = 1; // start timer 0
16       while (TCON_TF0 == 0); // wait for
17       TCON_TF0 = 0; // clear overflow fla
18   }
19   P3_B0=0;
20   TCON_TF0 = 0; // clear overflow flag
21   TCON_TR0= 0; // stop
22
23   // ---- ECHO capture ----
24

```

> 0101 TIMER 5		TIMER 5 Registers
> 0101 TIMER Setup		TIMER Setup Registers
> 0101 CKCON0	0x0	Clock Control 0
> 0101 CKCON1	0x4	Clock Control 1
✓ 1010 TCON	0x35	Timer 0/1 Control
0101 TF1	0x0 - NOT_SET (Timer 1 did not o...)	Timer 1 Overflow Flag
0101 TR1	0x0 - STOP (Stop Timer 1.)	Timer 1 Run Control
0101 TF0	0x1 - SET (Timer 0 overflowed.)	Timer 0 Overflow Flag
0101 TR0	0x1 - RUN (Start Timer 0 running.)	Timer 0 Run Control
0101 IE1	0x0 - NOT_SET (Edge/level not de...)	External Interrupt 1
0101 IT1	0x1 - EDGE (INT1 is edge trigger...)	Interrupt 1 Type Select
0101 IE0	0x0 - NOT_SET (Edge/level not de...)	External Interrupt 0
0101 ITO	0x1 - EDGE (INT0 is edge trigger...)	Interrupt 0 Type Select
> 1010 TMOD	0x22	Timer 0/1 Mode
> 0101 UART 1		UART 1 Registers
> 0101 UART 0		UART 0 Registers
1010 SBUF0	0x0	UART0 Serial Port Data Buffer
1010 SCON0	0x40	UART0 Serial Port Control
0101 SMODE	0x0 - 8 BIT (8-bit UART with Vari...)	Serial Port 0 Operation Mode

```

214     PCA0CNO_CNF = 4; // start PCA0
215     while(PCA0CNO_CCF0==0); //waiting echo r
216     REG_PCA_RISE=PCA0CPH0*0x100 + PCA0CPL0;
217     PCA0CNO_CCF0=0; //clear the flag
218     PCA0CNO_CF=0;
219     //get ready to catch echo signal fall
220     PCA0CPMO=PCA0CPM0_CAPN__ENABLED;
221     while(PCA0CNO_CCF0==0); //waiting echo f
222
223     REG_PCA_FALL = PCA0CPH0*0x100 + PCA0CPL
224     PCA0CNO_CR = 0; // stop PCA0
225     PCA0CNO_CCF0 = 0;
226     if(PCA0CNO_CF)
227         PCA0CNO_CFF=0;
228
229     // PCA0 clock period is 0.5usec then it
230     // we know that the max echo time is ab
231     if(REG_PCA_FALL>REG_PCA_RISE)
232         delta_reg= 0xffff - REG_PCA_RISE +
233     else
234         delta_reg= REG_PCA_FALL- REG_PCA_RI
235
236     // PCA0 clock period is 0.5usec
237     delta_reg *= 0.5; //echo time[usec]

```

דוחי עליית פולס האקוֹן	
> Channel 1	Channel 1 Registers
> Channel 2	Channel 2 Registers
> Channel 3	Channel 3 Registers
> Channel 4 / WDT	Channel 4 / WDT Registers
▼ PCA 0	PCA 0 Registers
> PCA0	PCA Counter/Timer Word
▼ PCA0CN0	PCA Control 0
► CF	0x0 - NOT_SET (The PCA counter/t... PCA Counter/Timer Overflow Flag
► CR	0x1 - RUN (Start the PCA Counter/... PCA Counter/Timer Run Control
► CCF4	0x0 - NOT_SET (A match or captur... PCA Module 4 Capture/Compare ...
► CCF3	0x0 - NOT_SET (A match or captur... PCA Module 3 Capture/Compare ...
► CCF2	0x0 - NOT_SET (A match or captur... PCA Module 2 Capture/Compare ...
► CCF1	0x0 - NOT_SET (A match or captur... PCA Module 1 Capture/Compare ...
► CCF0	0x1 - SET (A match or capture occ... PCA Module 0 Capture/Compare ...
> PCA0MD	PCA Mode
> PMU	PMU Registers
► P0	P0 Registers
► P1	P1 Registers
► P2	P2 Registers

```
213     PCA0CN0_CR = 1; // start PCA0
214     while(PCA0CN0_CCF0==0); //waiting echo r
215     REG_PCA_RISE=PCA0CPHO*0x100 + PCA0CPL0;
216     PCA0CN0_CCF0=0; //clear the flag
217     PCA0CN0_CCF=0;
218     //get ready to catch echo signal fall
```

עדיון ביצוע ערך		עדיון ביצוע ערך
>	Port Config	Port Config Registers
▼	Channel 0	Channel 0 Registers
▼	PCA0CP0	PCA Channel 0 Capture Module W...
▼	PCA0CPH0	PCA Channel 0 Capture Module Hi...
▼	PCA0CPL0	PCA Channel 0 Capture Module L...

```
221  
222 REG_PCA_FALL = PCA0CPH0*0x100 + PCA0CPL0  
223 PCA0CN0_CR = 0; // stop PCA0  
224 PCA0CN0_CCF0 = 0;  
225 if(PCA0CN0_CF)  
226     PCA0CN0_CF=0;  
227
```

ערכי הילכידה בעט ירידה:	
> Port Config	Port Config Registers
▼ Channel 0	Channel 0 Registers
▼ PCA0CPO	PCA Channel 0 Capture Module W...
PCA0CPH0	PCA Channel 0 Capture Module Hi...
PCA0CPLO	PCA Channel 0 Capture Module L...
> PCA0CPMO	PCA Channel 0 Capture/Compare ...
> Channel 1	Channel 1 Registers

זהוי את אקו בירידה :

```

    ןחוּ לְטוֹכַן עֲנִינִים
20     while(PCA0LINE_CCF0==0); //Waiting echo to
21
22     REG_PCA_FALL = PCA0CPH0*0x100 + PCA0CPL
23     PCA0CN0_CR = 0; // stop PCA0
24     PCA0CN0_CCF0 = 0;
25     if(PCA0CN0_CF)
26         PCA0CN0_CF=0;
27
28     // PCA0 clock period is 0.5usec then it
29     // we know that the max echo time is ab
30     if(REG_PCA_FALL<REG_PCA_RISE)
31         delta_reg= 0xffff - REG_PCA_RISE +
32     else
33         delta_reg= REG_PCA_FALL- REG_PCA_RI
34
35     // PCA0 clock period is 0.5usec

```

> 1010 Channel 4 / WDT			Channel 4 / WDT Registers
1010 PCA 0			PCA 0 Registers
1010 PCA0	0x21AB		PCA Counter/Timer Word
1010 PCA0H	0x21		PCA Counter/Timer High Byte
1010 PCA0L	0xAB		PCA Counter/Timer Low Byte
1010 PCA0CN0	0x41		PCA Control 0
1010 CF	0x0 - NOT_SET (The PCA counter/t...		PCA Counter/Timer Overflow Flag
1010 CR	0x1 - RUN (Start the PCA Counter/...		PCA Counter/Timer Run Control
1010 CCF4	0x0 - NOT_SET (A match or captur...		PCA Module 4 Capture/Compare ...
1010 CCF3	0x0 - NOT_SET (A match or captur...		PCA Module 3 Capture/Compare ...
1010 CCF2	0x0 - NOT_SET (A match or captur...		PCA Module 2 Capture/Compare ...
1010 CCF1	0x0 - NOT_SET (A match or captur...		PCA Module 1 Capture/Compare ...
1010 CCF0	0x1 - SET (A match or capture occ...		PCA Module 0 Capture/Compare ...
1010 PCA0MD	0x0		PCA Mode

```

    PCA0CN0_CCF0=0;
    //get ready to catch echo signal fall
    PCA0CPM0=PCA0CPM0_CAPN_ENABLED;
    while(PCA0CN0_CCF0==0); //waiting echo f
221
222     REG_PCA_FALL = PCA0CPH0*0x100 + PCA0CPL
223     PCA0CN0_CR = 0; // stop PCA0
224     PCA0CN0_CCF0 = 0;
225     if(PCA0CN0_CF)
226         PCA0CN0_CF=0;
227
228     // PCA0 clock period is 0.5usec then it
229     // we know that the max echo time is ab
230     if(REG_PCA_FALL<REG_PCA_RISE)
231         delta_reg= 0xffff - REG_PCA_RISE +
232     else
233         delta_reg= REG_PCA_FALL- REG_PCA_RI
234
235     // PCA0 clock period is 0.5usec
236     delta_reg *= 0.5; //echo_time[usec]
237     return delta_reg;
238 }
239

```

Expression	Type	Value	Address
REG_PCA_RISE	signed int	1140	RAM:0x2e
REG_PCA_FALL	signed int	8618	RAM:0x4
delta_reg	signed int	7478	RAM:0x2
+ Add new expression			

ערך זה מוכפל בזמן המחזר של השעון המודד. (חץ מיקרו שנייה)
ומוחזר הזמן במיקרו שנייות של פולס האקן.

LCK Watchdog Timer

▼ PCA Counter/Timer Configuration

PCA Clock Frequency	2.000 MHz
PCA Clock Period	500.000 ns
Select PCA Counter/Timer Pulse	SYSCLK / 12
PCA Counter/Timer	0 (0x0)
PCA Counter/Timer Idle Control	Normal
Enable PCA Counter/Timer Overflow Interrupt	Disabled

1010 PCA0MD	0x0	PCA Mode
1010 CIDL	0x0 - NORMAL (PCA continues to ...	PCA Counter/Timer Idle Control
1010 WDTE	0x0 - DISABLED (Disable Watchd...)	Watchdog Timer Enable
1010 WDLCK	0x0 - UNLOCKED (Watchdog Time...)	Watchdog Timer Lock
1010 CPS	0x0 - SYSCLK_DIV_12 (System cloc...)	PCA Counter/Timer Pulse Select
1010 ECF	0x0 - OVF_INT_DISABLED (Disable...)	PCA Counter/Timer Overflow Inter...
1010 PMU		PMU Registers
1010 P0		P0 Registers

היפוך הזמן :

חוור לתוכן עניינים

```

03 overflow_counter=0;
04 TCON_TR1=1; // start timer 1
05
06 while(overflow_counter<=BAUD_RATE)//wait
07 {
08     if(ton_state>=1 && ton_state<=4){
09         if(buzz_cnt == (BAUD_RATE/scale)){
10             toggle_buzz();
11             buzz_cnt=0;
12         }
13     }
14     while(TCON_TF1==0); // wait for timer
15     TCON_TF1 = 0; // clear overflow flag
16     overflow_counter++;
17     buzz_cnt++;
18 }
19
20 TCON_TR1=0; // stop timer 1
21 delay_end=1;
22 }
23
24 void distance_int_to_text(int distance_mesa
25 {
26     int temp;
27     int temp_inv;
28     int cnt;
29     ...
30 }
```

> D101 Channel 2		Channel 2 Registers
> D101 Channel 3		Channel 3 Registers
> D101 Channel 4 / WDT		Channel 4 / WDT Registers
> D101 PCA 0		PCA 0 Registers
> D101 PMU		PMU Registers
> D101 P0		P0 Registers
▼ D101 P1		P1 Registers
▼ D101 P1	0x90	Port 1 Pin Latch
D101 B7	0x1 - HIGH (P1.7 is high. Set P1.7 t...	Port 1 Bit 7 Latch
D101 B6	0x0 - LOW (P1.6 is low. Set P1.6 to...	Port 1 Bit 6 Latch
D101 B5	0x0 - LOW (P1.5 is low. Set P1.5 to...	Port 1 Bit 5 Latch
D101 B4	0x1 - HIGH (P1.4 is high. Set P1.4 t...	Port 1 Bit 4 Latch
D101 B3	0x0 - LOW (P1.3 is low. Set P1.3 to...	Port 1 Bit 3 Latch
D101 B2	0x0 - LOW (P1.2 is low. Set P1.2 to...	Port 1 Bit 2 Latch
D101 B1	0x0 - LOW (P1.1 is low. Set P1.1 to...	Port 1 Bit 1 Latch
D101 B0	0x0 - LOW (P1.0 is low. Set P1.0 to...	Port 1 Bit 0 Latch
> D101 P1MDIN	0xDF	Port 1 Input Mode
> D101 P1MDOUT	0x51	Port 1 Output Mode
> D101 P1SKIP	0xFE	Port 1 Skip
> D101 P2		P2 Registers
> D101 P3		P3 Registers
> D101 P4		P4 Registers
▼ Reset Sources		Reset Sources Registers

לחיצה על לחץ אפוי מובילת לפסיקה 0 :

```

10 // CSK_INTERRUPT // 
11 //-----
12 // INT0 ISR Content goes here. Remember to clear flag bits:
13 // TCON::IE0 (External Interrupt 0)
14 //-----
15 SI_INTERRUPT (INT0_ISR, INT0_IRQn)
16 {
17     TCON_IE0 = 0;
18     button0_pressed = 1;
19     button1_pressed = 0;
20     in_process = 1; // to indicate start periodic process
21 }
22
23 //-----
24 // INT1_ISR
25 //-----
26 // INT1 ISR Content goes here. Remember to clear flag bits:
27 // TCON::IE1 (External Interrupt 1)
28
29
30
31
32
33
34
35
36
37
38 }
```

> TIMER Setup		TIMER Setup Registers
> CKCON0	0x0	Clock Control 0
> CKCON1	0x4	Clock Control 1
▼ TCON	0x2	Timer 0/1 Control
TF1	0x0 - NOT_SET (Timer 1 did not o...	Timer 1 Overflow Flag
TR1	0x0 - STOP (Stop Timer 1.)	Timer 1 Run Control
TF0	0x0 - NOT_SET (Timer 0 did not o...	Timer 0 Overflow Flag
TR0	0x0 - STOP (Stop Timer 0.)	Timer 0 Run Control
IE1	0x0 - NOT_SET (Edge/level not de...	External Interrupt 1
IT1	0x0 - LEVEL (INT1 is level trigger...	Interrupt 1 Type Select
IE0	0x1 - SET (Edge/level detected)	External Interrupt 0
IT0	0x0 - LEVEL (INT0 is level trigger...	Interrupt 0 Type Select
> TMOD	0x22	Timer 0/1 Mode
> UART 1		UART 1 Registers
> UART 0		UART 0 Registers

הורדת הדגל :

```

15 SI_INTERRUPT (INT0_ISR, INT0_IRQn)
16 {
17     TCON_IE0 = 0;
18     button0_pressed = 1;
19     button1_pressed = 0;
20     in_process = 1; // to indicate start periodic process
21 }
22
23 //-----
24 // INT1_ISR
25 //-----
26 // INT1 ISR Content goes here. Remember to clear flag bits:
27 // TCON::IE1 (External Interrupt 1)
28
29 SI_INTERRUPT (INT1_ISR, INT1_IRQn)// button1 - stop periodic mesure
30 {
31     TCON_IE1 = 0;
32     button1_pressed = 1;
33     button0_pressed = 0;
34     in_process = 0; // to indicate end periodic process
35 }
36
37
38 }
```

> CKCON1	0x4	Clock Control 1
▼ TCON	0x48	Timer 0/1 Control
TF1	0x0 - NOT_SET (Timer 1 did not o...	Timer 1 Overflow Flag
TR1	0x1 - RUN (Start Timer 1 running.)	Timer 1 Run Control
TF0	0x0 - NOT_SET (Timer 0 did not o...	Timer 0 Overflow Flag
TR0	0x0 - STOP (Stop Timer 0.)	Timer 0 Run Control
IE1	0x1 - SET (Edge/level detected)	External Interrupt 1
IT1	0x0 - LEVEL (INT1 is level trigger...	Interrupt 1 Type Select
IE0	0x0 - NOT_SET (Edge/level not de...	External Interrupt 0
IT0	0x0 - LEVEL (INT0 is level trigger...	Interrupt 0 Type Select
> TMOD	0x22	Timer 0/1 Mode
> UART 1		UART 1 Registers
> UART 0		UART 0 Registers
> USB Library		USB Library Registers
> USB		USB Registers
> Supply Monitor		Supply Monitor Registers
> Voltage Reference		Voltage Reference Registers

הורדת הדגל :

חוור לתוכן עניינים

```
(INT1_INTERRUPT, INT1_ISR, INT1_IRQn)
16 {
17     TCON_IE0 = 0;
18     button0_pressed = 1;
19     button1_pressed = 0;
20     in_process = 1; // to indicate start periodic process
21 }
22
23@-----
24 // INT1_ISR
25 //-----
26 // INT1 ISR Content goes here. Remember to clear flag bits:
27 // TCON::IE1 (External Interrupt 1)
28 //-----
29@SI_INTERRUPT (INT1_ISR, INT1_IRQn)// button1 - stop pridic mesure
30 {
31     TCON_IE1 = 0;
32     button1_pressed = 1;
33     button0_pressed = 0;
34     in_process = 0; // to indicate end periodic process
35 }
36
37
```

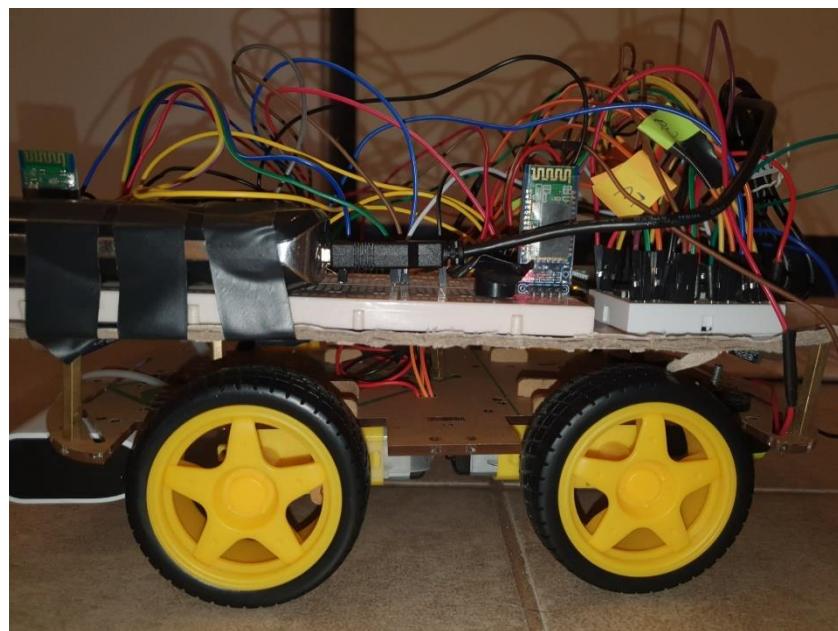
		CLOCK CONTROL
TCON	0x40	Timer 0/1 Control
TF1	0x0 - NOT_SET (Timer 1 did not o...	Timer 1 Overflow Flag
TR1	0x1 - RUN (Start Timer 1 running.)	Timer 1 Run Control
TF0	0x0 - NOT_SET (Timer 0 did not o...	Timer 0 Overflow Flag
TR0	0x0 - STOP (Stop Timer 0.)	Timer 0 Run Control
IE1	0x0 - NOT_SET (Edge/level not de...	External Interrupt 1
IT1	0x0 - LEVEL (INT1 is level triggere...	Interrupt 1 Type Select
IE0	0x0 - NOT_SET (Edge/level not de...	External Interrupt 0
IT0	0x0 - LEVEL (INT0 is level triggere...	Interrupt 0 Type Select
TMOD	0x22	Timer 0/1 Mode
UART 1		UART 1 Registers
UART 0		UART 0 Registers
USB Library		USB Library Registers
USB		USB Registers
Supply Monitor		Supply Monitor Registers
Voltage Reference		Voltage Reference Register
	...	
Name : TMOD		
Hex:0x22		

תהליך העבודה ותיקור תקלות שהתעוררו במהלך הפroyskt

1. קושי באינטגרציה של המערכת - המערכת לא עבדה כמו שציפינו בכך שלא התחמקה ממכוולים. פתרון - חילקנו את הקוד בצורה מסודרת לפונקציות כך שניתן לדבג את הקוד בצורה ייעילה וכך מצאנו את הבעיה.
2. ריבוי חוטים הקשה על חיבור כל המערכת. על מנת להתמודד עם הבעיה ביצענו חיבור קודם בתוך כל תשתית מערכת, ולבסוף חיבור בין תתי מערכות.



3. היה علينا לבצע הלחמה של חוטים. למදנו כיצד מלחמים חוטים והשאנו ציוד מחבר על מנת לבצע את הלחמות כנדרש.
4. היה לנו חוסר במקומות בהצבת הרכיבים על גביו המוצר. הוספנו מפלס נוסף שנutan לנו עוד מקום.



5. הדרייבר לא עבד עם אות PWM שסיפקנו לו. פתרון - עברנו על datasheet של הדרייבר ותיקנו את הפרמטרים שמסופקים לו בהתאם למה שנדרש - תדר וגובה המתח.

6. ניסינו להשתמש באותו התקן בלוטוס בעברור קליטה לARM ושידור מ8051 , אבל הדבר יצר התנגדויות בערוצ ולבן הוספנו התקן נוסף כדי למנוע תקלת זו . היה ניתן גם לפתור בעיה זו על ידי שליחה נקודתית ולא סדרתית של SOS וכן לאפשר לערוצ להירגע ולהיות פנוויל קליטה .