אורט בראודה
המכללה האקדמית להנדסה

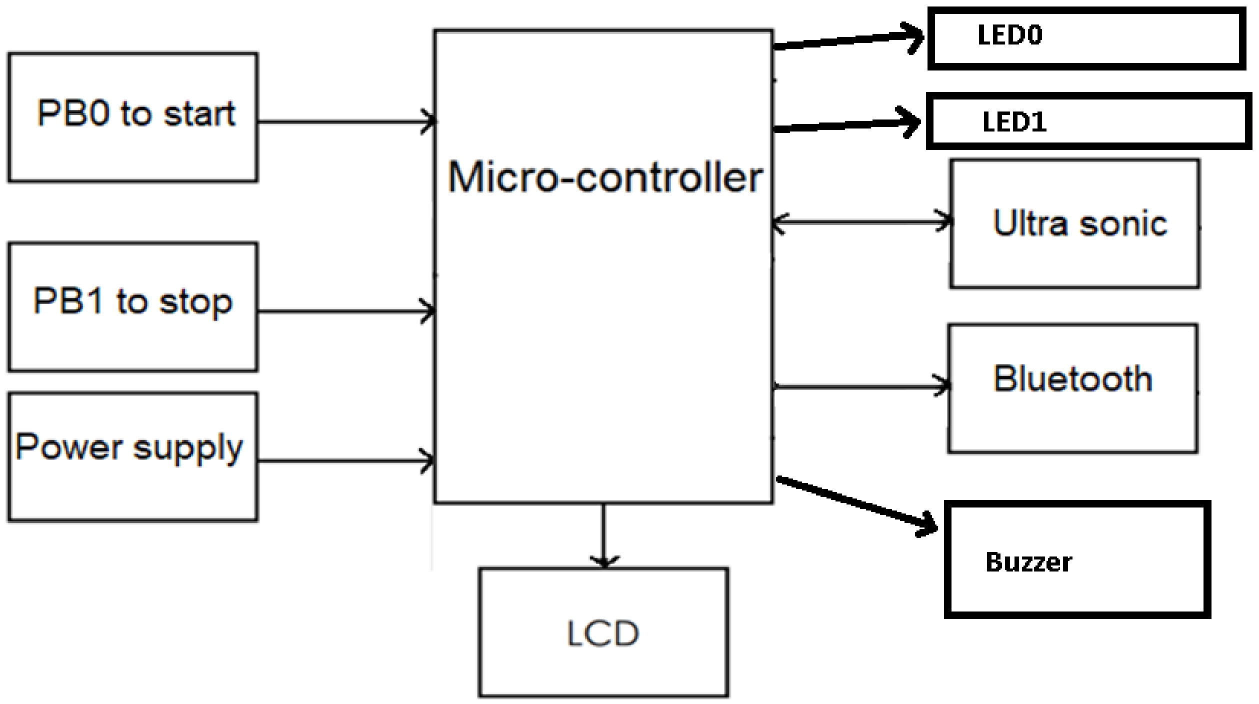
המחלקה להנדסת חשמל ואלקטרוניקה

פרויקט סוף מיקרו בקרים

מחמוד חגה	318396355
נאיל חסון	318386364

תוכן עניינים:

תרשים מלבנים למערכת:



We use Timer0 and Timer1 for trigger, Timer1 reach overflow each 0.5 second, Timer0 reach overflow each 10us. Pin D2 is output for the trigger.

Timer2 cc0 for echo, we use pin A12 for input the echo from the ultrasonic sensor.

Pins:

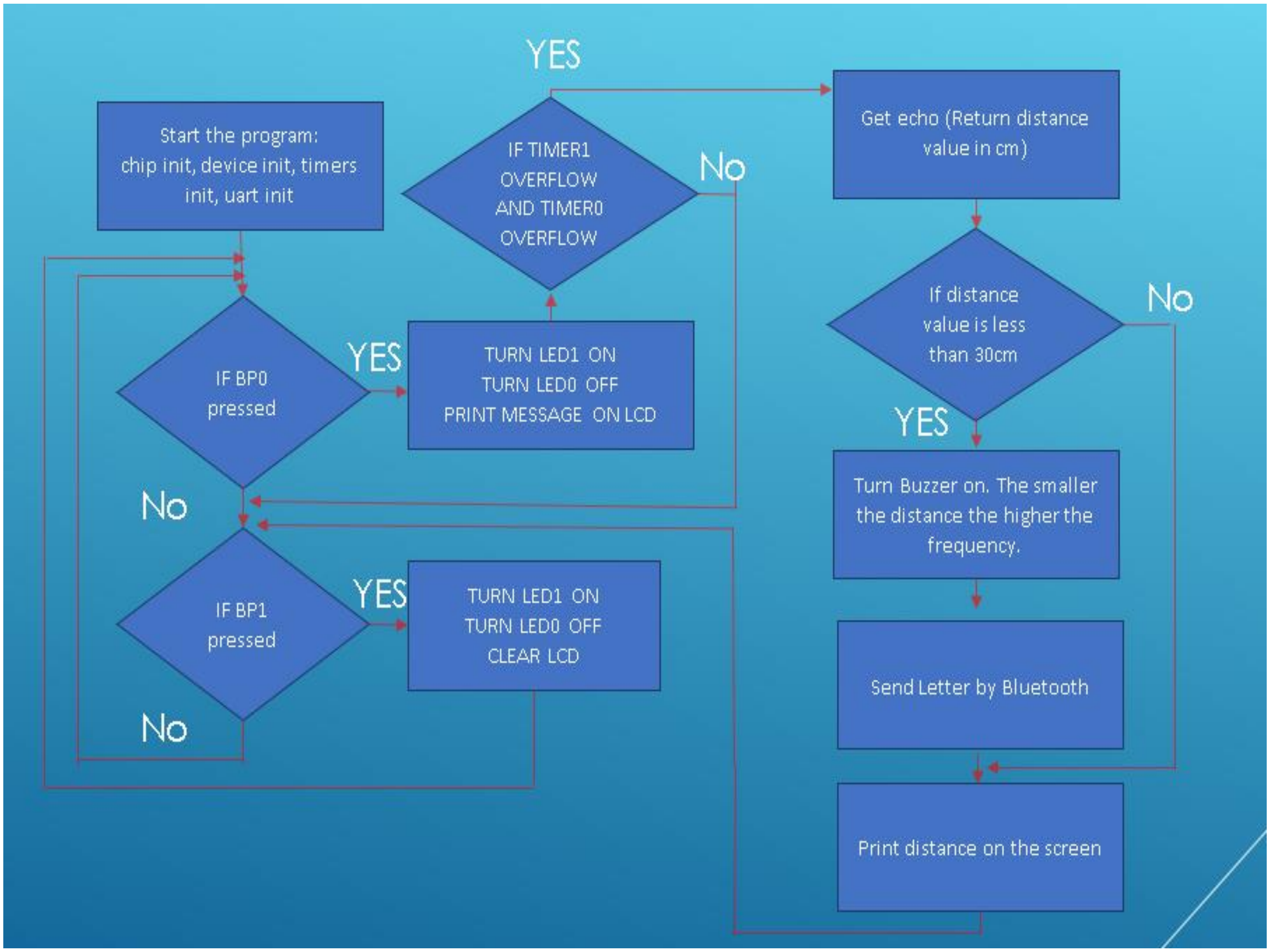
Pin E1 as output for Buzzer.

Pin D7 as output for TRx.

Pin A12 as input from echo.

Pin D2 as output for Trigger

תרשים זרמיה לפונקציות:


Start the program: chip init, device init, timers init, uart init
IF BP0 pressed
YES
TURN LED1 ON
TURN LED0 OFF
PRINT MESSAGE ON LCD
No
IF BP1 pressed
YES
TURN LED1 ON
TURN LED0 OFF
CLEAR LCD
No
IF TIMER1 OVERFLOW AND TIMER0 OVERFLOW
YES
No
Get echo (Return distance value in cm)
If distance value is less than 30cm
YES
No
Turn Buzzer on. The smaller the distance the higher the frequency.
Send Letter by Bluetooth
Print distance on the screen

הסבר שיקולי התכנות:

איך מיצריים TRIGGER:

We use Timer0 and Timer1 for trigger, Timer1 reach overflow each 0.5 second, Timer0 reach overflow each 10us. We wait for Timer1 overflow after happen we clear overflow, set to pin D2 and start timer0, when timer0 overflow happen, we clear pin D2, clear overflow flag and stop Timer0, and wait again for Timer1 overflow.

איך מודדים echo :

First, we send pulse Trigger to the ultrasonic sensor, and we get back pulse echo from the sensor.
We start Timer2 (to work in rising edge) then wait for rising edge from the echo, after it happen we save it, then we define Timer2 to work in falling edge then wait for falling edge from the echo. At the end we stop the Timer 2.

At the end we calculate echo ticks, by sub when happen rising edge from when happen falling edge.

If (echo_end<echo_start) : echo_ticks=0xffff-echo_start+echo_end;

Else : echo_ticks=echo_end-echo_start;

איך מחשבים מחלק מתח לרגל echo :

Here we worked with 3.3v, so there is no need for voltage divider.

חישוב ערכי TIMERS :

Fclock =14MHZ (for all TIMERS).

TIMER0:

We choose register frequency to be 50KHZ (pulse for 10us).

We choose Prescale to be 1.

We calculate the value of Top:

TOP = (Fclock) / (2*Fregister*Prescale) = 140

TIMER1:

We choose register frequency to be 2HZ (each 0.5s).

We choose Prescale to be 64.

We calculate the value of Top:

TOP = (Fclock) / (2*Fregister*Prescale) = 54687

TIMER2:

We choose register frequency to be 2HZ (each 0.5s).

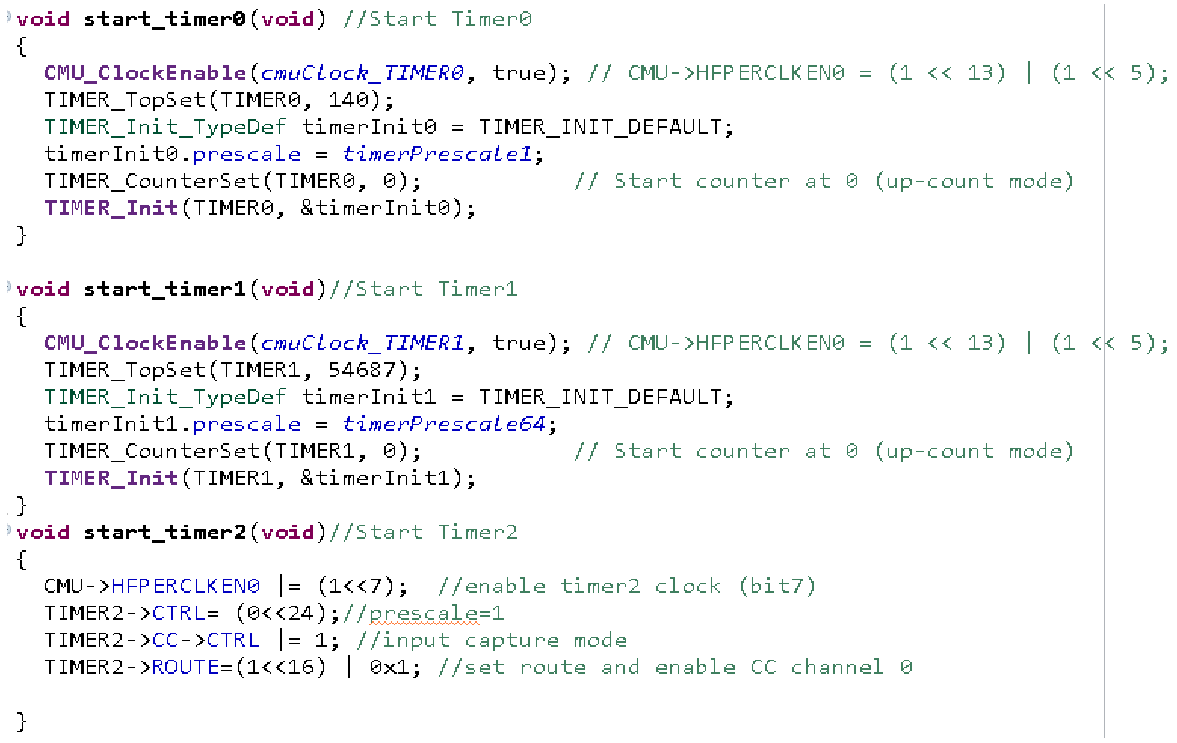
We choose Prescale to be 64.

We calculate the value of Top:

TOP = (Fclock) / (2*Fregister*Prescale) = 54687

צילומי מסך של אתחולים רלוונטיים:

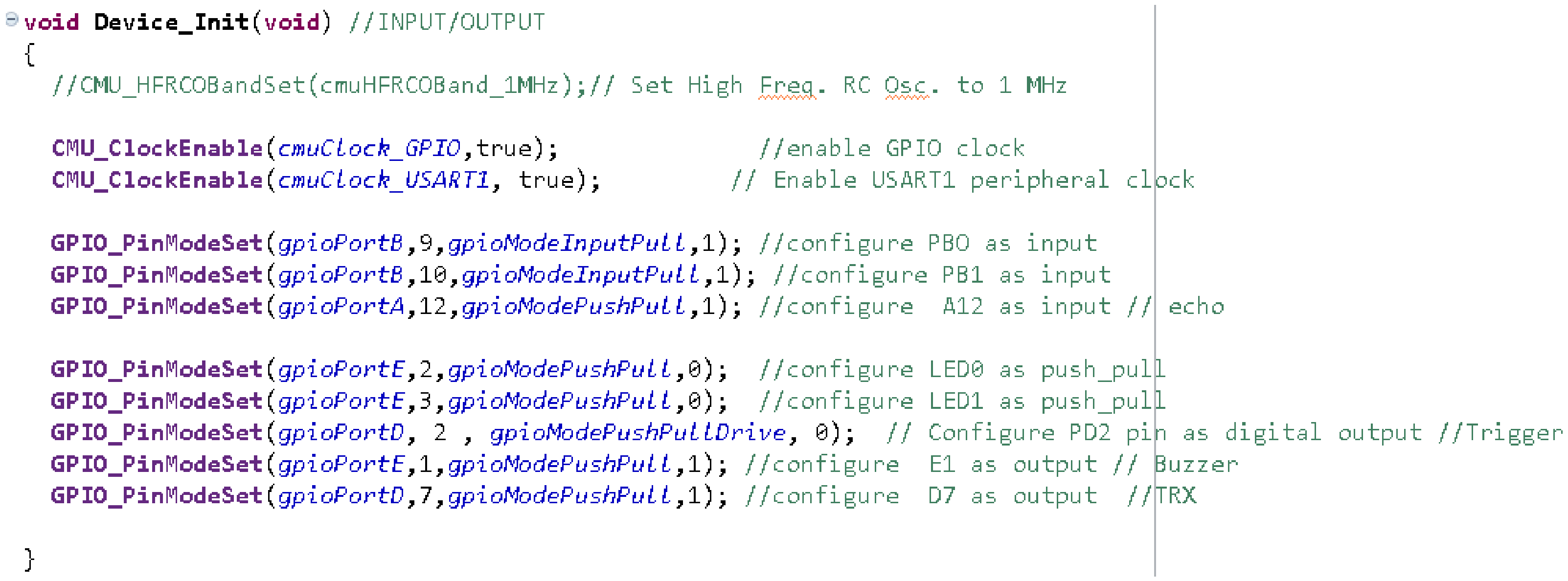
TIMERS:



```
void start_timer0(void) //Start Timer0
{
   CMU_ClockEnable(cmuClock_TIMER0, true); // CMU->HFPERCLKEN0 = (1 << 13) | (1 << 5);
   TIMER_TopSet(TIMER0, 140);
   TIMER_Init_TypeDef timerInit0 = TIMER_INIT_DEFAULT;
   timerInit0.prescale = timerPrescale1;
   TIMER_CounterSet(TIMER0, 0);            // Start counter at 0 (up-count mode)
   TIMER_Init(TIMER0, &timerInit0);
}

void start_timer1(void)//Start Timer1
{
   CMU_ClockEnable(cmuClock_TIMER1, true); // CMU->HFPERCLKEN0 = (1 << 13) | (1 << 5);
   TIMER_TopSet(TIMER1, 54687);
   TIMER_Init_TypeDef timerInit1 = TIMER_INIT_DEFAULT;
   timerInit1.prescale = timerPrescale64;
   TIMER_CounterSet(TIMER1, 0);            // Start counter at 0 (up-count mode)
   TIMER_Init(TIMER1, &timerInit1);
}
void start_timer2(void)//Start Timer2
{
   CMU->HFPERCLKEN0 |= (1<<7);  //enable timer2 clock (bit7)
   TIMER2->CTRL= (0<<24);//prescale=1
   TIMER2->CC->CTRL |= 1; //input capture mode
   TIMER2->ROUTE=(1<<16) | 0x1; //set route and enable CC channel 0

}
```

TIMER0,1 explained in previous pages, Timer2 we enable it and put it in input capture mode, and we do route to pin A12 chanel0.

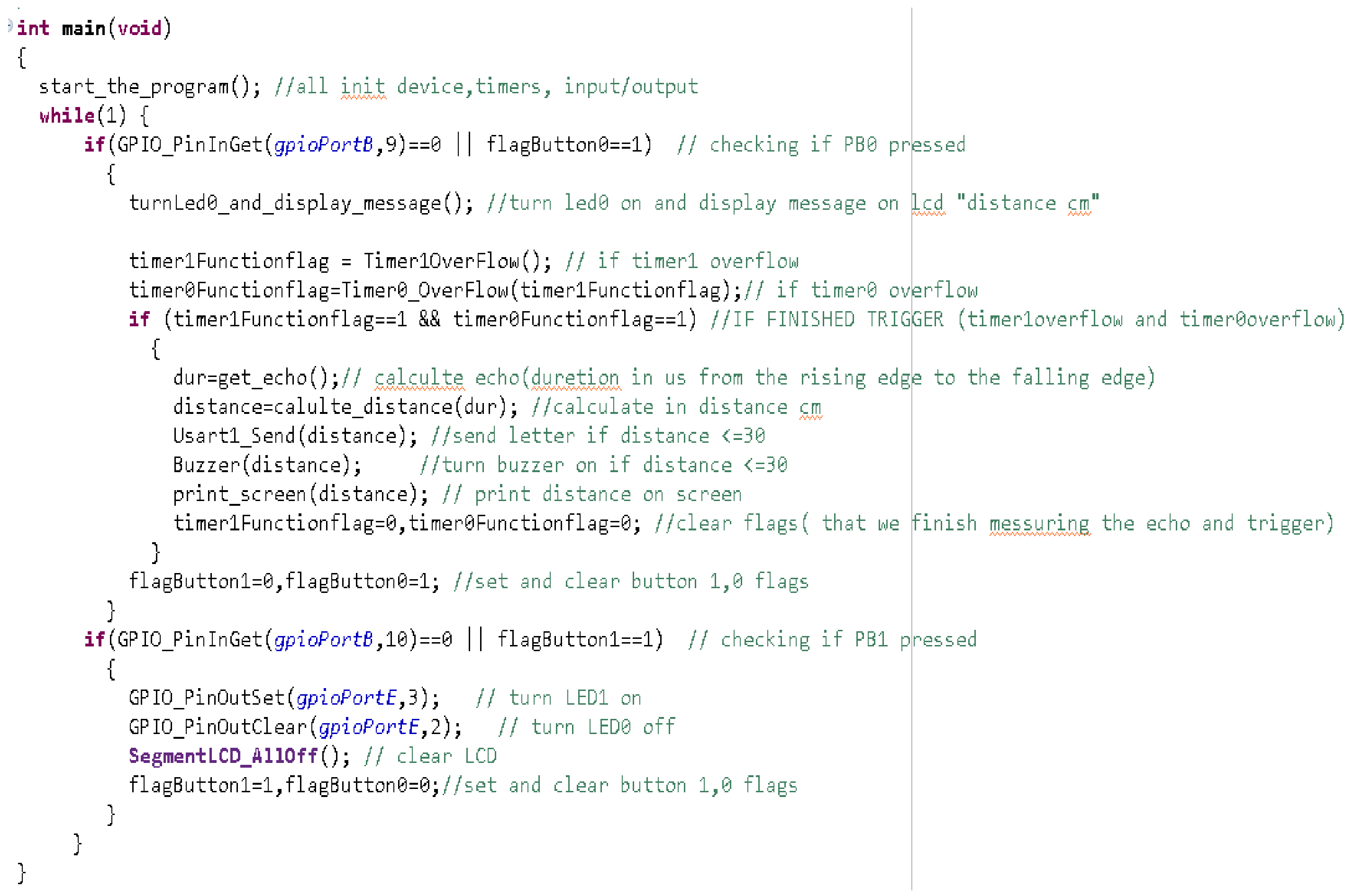
Device int:



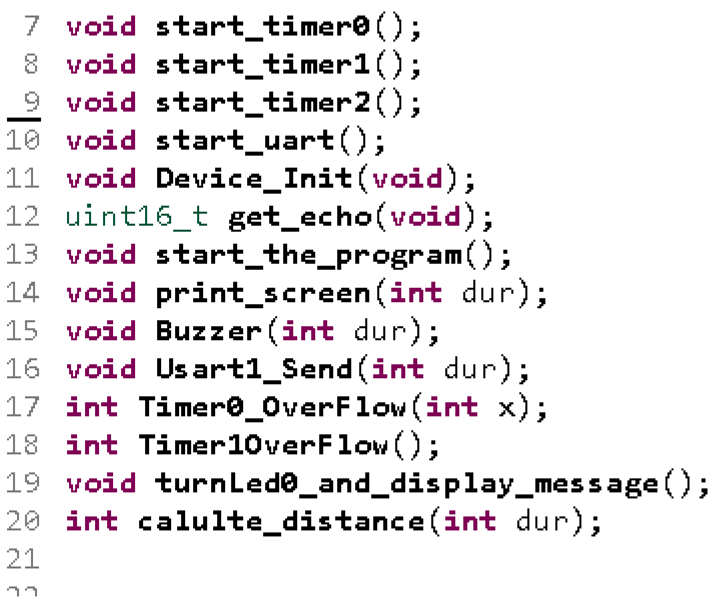
```
void Device_Init(void) //INPUT/OUTPUT
{
  //CMU_HFRCOBandSet(cmuHFRCOBand_1MHz);// Set High Freq. RC Osc. to 1 MHz

  CMU_ClockEnable(cmuClock_GPIO,true);              //enable GPIO clock
  CMU_ClockEnable(cmuClock_USART1, true);         // Enable USART1 peripheral clock

  GPIO_PinModeSet(gpioPortB,9,gpioModeInputPull,1); //configure PB0 as input
  GPIO_PinModeSet(gpioPortB,10,gpioModeInputPull,1); //configure PB1 as input
  GPIO_PinModeSet(gpioPortA,12,gpioModePushPull,1); //configure  A12 as input // echo

  GPIO_PinModeSet(gpioPortE,2,gpioModePushPull,0);  //configure LED0 as push_pull
  GPIO_PinModeSet(gpioPortE,3,gpioModePushPull,0);  //configure LED1 as push_pull
  GPIO_PinModeSet(gpioPortD, 2 , gpioModePushPullDrive, 0);  // Configure PD2 pin as digital output //Trigger
  GPIO_PinModeSet(gpioPortE,1,gpioModePushPull,1); //configure  E1 as output // Buzzer
  GPIO_PinModeSet(gpioPortD,7,gpioModePushPull,1); //configure  D7 as output  //TRX

}
```

צילומי מסך של קוד:

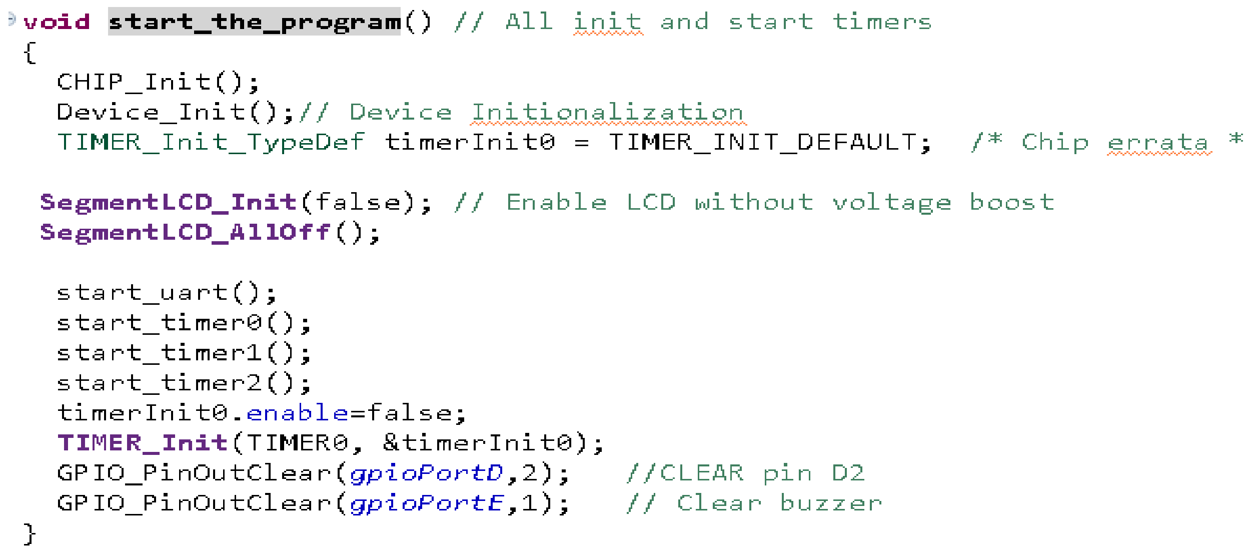
Main:



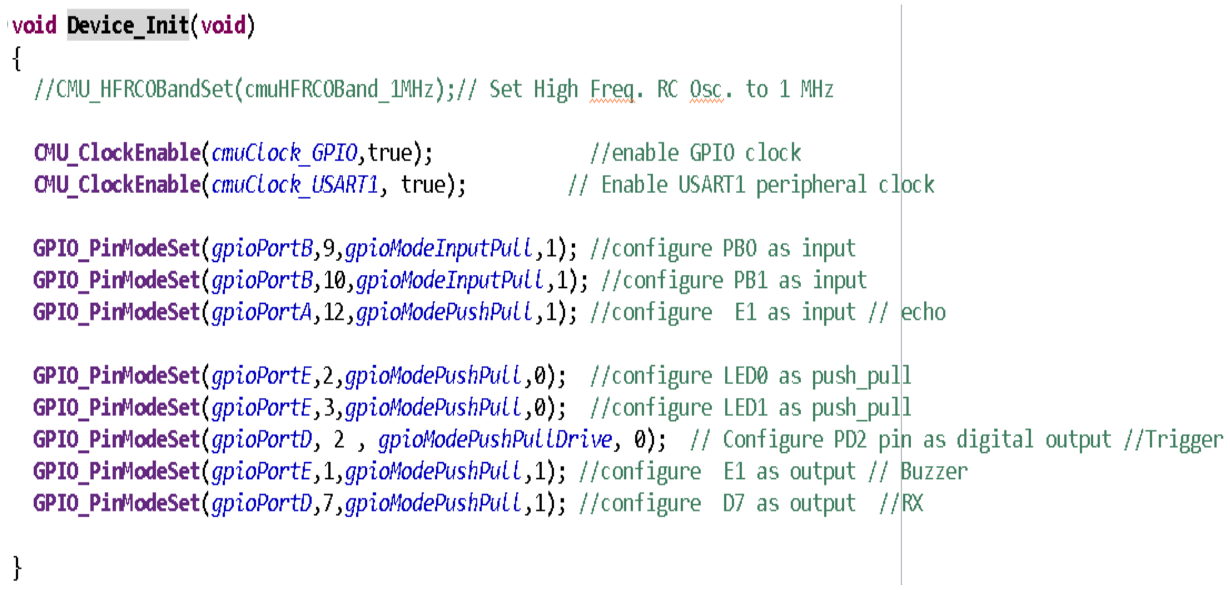
```c
int main(void)
{
  start_the_program(); //all init device,timers, input/output
  while(1) {
      if(GPIO_PinInGet(gpioPortB,9)==0 || flagButton0==1)  // checking if PB0 pressed
        {
          turnLed0_and_display_message(); //turn led0 on and display message on lcd "distance cm"

          timer1Functionflag = Timer1OverFlow(); // if timer1 overflow
          timer0Functionflag=Timer0_OverFlow(timer1Functionflag);// if timer0 overflow
          if (timer1Functionflag==1 && timer0Functionflag==1) //IF FINISHED TRIGGER (timer1overflow and timer0overflow)
            {
              dur=get_echo();// calculte echo(duretion in us from the rising edge to the falling edge)
              distance=calulte_distance(dur); //calculate in distance cm
              Usart1_Send(distance); //send letter if distance <=30
              Buzzer(distance);     //turn buzzer on if distance <=30
              print_screen(distance); // print distance on screen
              timer1Functionflag=0,timer0Functionflag=0; //clear flags( that we finish messuring the echo and trigger)
            }
          flagButton1=0,flagButton0=1; //set and clear button 1,0 flags
        }
      if(GPIO_PinInGet(gpioPortB,10)==0 || flagButton1==1)  // checking if PB1 pressed
        {
          GPIO_PinOutSet(gpioPortE,3);   // turn LED1 on
          GPIO_PinOutClear(gpioPortE,2);   // turn LED0 off
          SegmentLCD_AllOff(); // clear LCD
          flagButton1=1,flagButton0=0;//set and clear button 1,0 flags
        }
     }
}
```



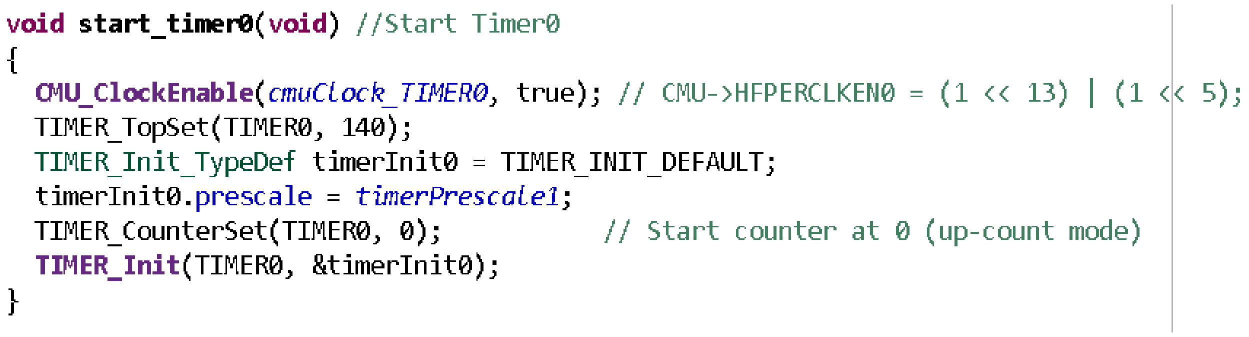
```c
void start_timer0();
void start_timer1();
void start_timer2();
void start_uart();
void Device_Init(void);
uint16_t get_echo(void);
void start_the_program();
void print_screen(int dur);
void Buzzer(int dur);
void Usart1_Send(int dur);
int Timer0_OverFlow(int x);
int Timer1OverFlow();
void turnLed0_and_display_message();
int calulte_distance(int dur);
```



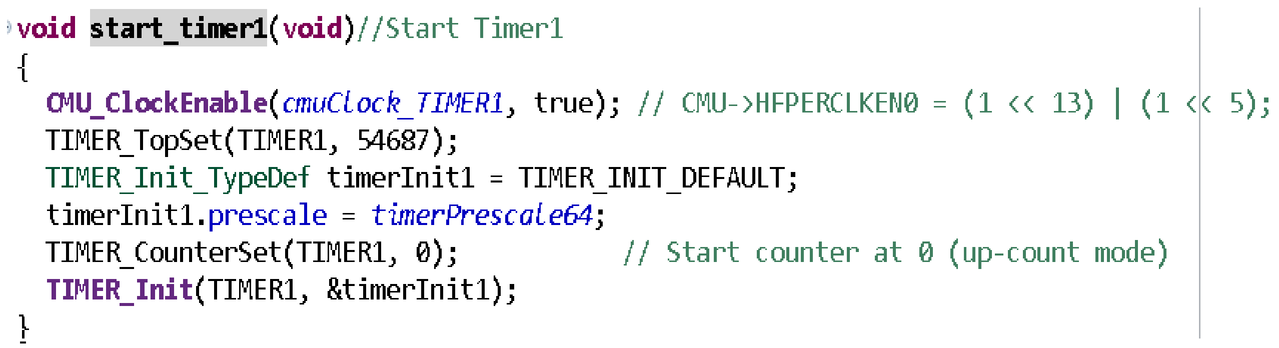
```
void start_the_program() // All init and start timers
{
  CHIP_Init();
  Device_Init();// Device Initionalization
  TIMER_Init_TypeDef timerInit0 = TIMER_INIT_DEFAULT;  /* Chip errata *

 SegmentLCD_Init(false); // Enable LCD without voltage boost
 SegmentLCD_AllOff();

  start_uart();
  start_timer0();
  start_timer1();
  start_timer2();
  timerInit0.enable=false;
  TIMER_Init(TIMER0, &timerInit0);
  GPIO_PinOutClear(gpioPortD,2);   //CLEAR pin D2
  GPIO_PinOutClear(gpioPortE,1);   // Clear buzzer
}
```



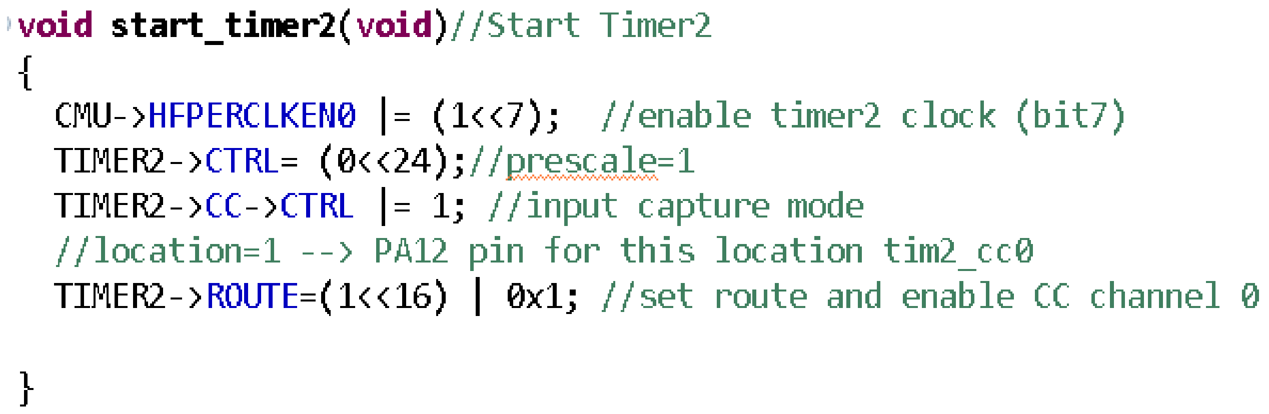
```
void Device_Init(void)
{
  //CMU_HFRCOBandSet(cmuHFRCOBand_1MHz);// Set High Freq. RC Osc. to 1 MHz

  CMU_ClockEnable(cmuClock_GPIO,true);             //enable GPIO clock
  CMU_ClockEnable(cmuClock_USART1, true);         // Enable USART1 peripheral clock

  GPIO_PinModeSet(gpioPortB,9,gpioModeInputPull,1); //configure PB0 as input
  GPIO_PinModeSet(gpioPortB,10,gpioModeInputPull,1); //configure PB1 as input
  GPIO_PinModeSet(gpioPortA,12,gpioModePushPull,1); //configure  E1 as input // echo

  GPIO_PinModeSet(gpioPortE,2,gpioModePushPull,0);  //configure LED0 as push_pull
  GPIO_PinModeSet(gpioPortE,3,gpioModePushPull,0);  //configure LED1 as push_pull
  GPIO_PinModeSet(gpioPortD, 2 , gpioModePushPullDrive, 0);  // Configure PD2 pin as digital output //Trigger
  GPIO_PinModeSet(gpioPortE,1,gpioModePushPull,1); //configure  E1 as output // Buzzer
  GPIO_PinModeSet(gpioPortD,7,gpioModePushPull,1); //configure  D7 as output  //RX

}
```



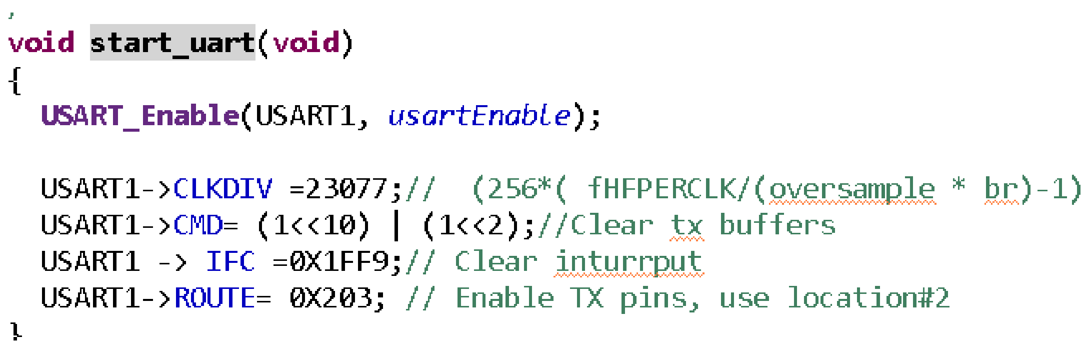
```
void start_timer0(void) //Start Timer0
{
  CMU_ClockEnable(cmuClock_TIMER0, true); // CMU->HFPERCLKEN0 = (1 << 13) | (1 << 5);
  TIMER_TopSet(TIMER0, 140);
  TIMER_Init_TypeDef timerInit0 = TIMER_INIT_DEFAULT;
  timerInit0.prescale = timerPrescale1;
  TIMER_CounterSet(TIMER0, 0);            // Start counter at 0 (up-count mode)
  TIMER_Init(TIMER0, &timerInit0);
}
```



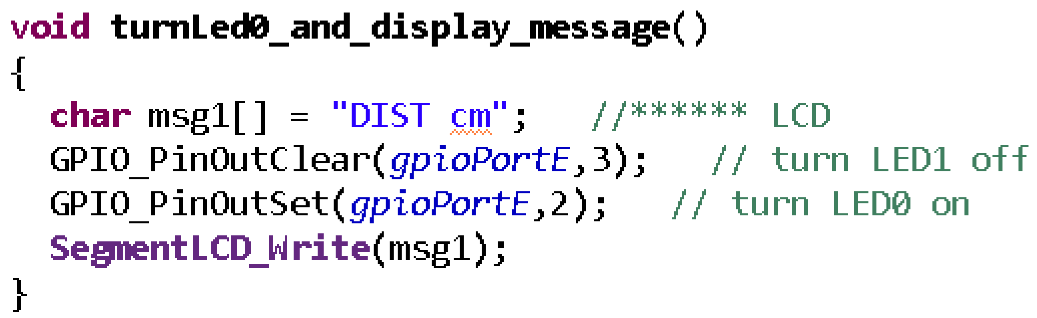
```
void start_timer1(void)//Start Timer1
{
  CMU_ClockEnable(cmuClock_TIMER1, true); // CMU->HFPERCLKEN0 = (1 << 13) | (1 << 5);
  TIMER_TopSet(TIMER1, 54687);
  TIMER_Init_TypeDef timerInit1 = TIMER_INIT_DEFAULT;
  timerInit1.prescale = timerPrescale64;
  TIMER_CounterSet(TIMER1, 0);            // Start counter at 0 (up-count mode)
  TIMER_Init(TIMER1, &timerInit1);
}
```



```
void start_timer2(void)//Start Timer2
{
  CMU->HFPERCLKEN0 |= (1<<7);  //enable timer2 clock (bit7)
  TIMER2->CTRL= (0<<24);//prescale=1
  TIMER2->CC->CTRL |= 1; //input capture mode
  //location=1 --> PA12 pin for this location tim2_cc0
  TIMER2->ROUTE=(1<<16) | 0x1; //set route and enable CC channel 0

}
```



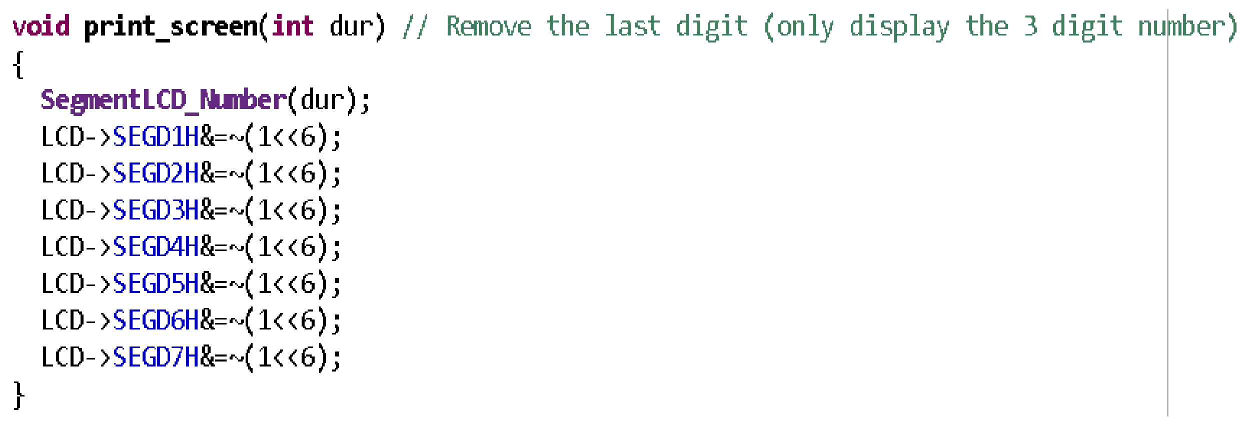
```
void start_uart(void)
{
  USART_Enable(USART1, usartEnable);

  USART1->CLKDIV =23077;//  (256*( fHFPERCLK/(oversample * br)-1)
  USART1->CMD= (1<<10) | (1<<2);//Clear tx buffers
  USART1 -> IFC =0X1FF9;// Clear inturrput
  USART1->ROUTE= 0X203; // Enable TX pins, use location#2
}
```



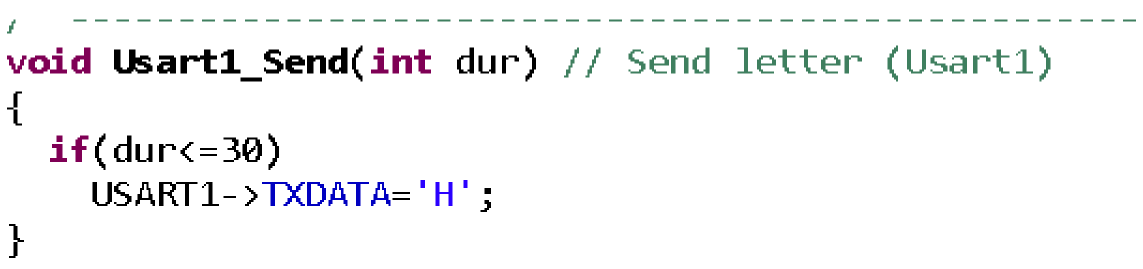
```
void turnLed0_and_display_message()
{
  char msg1[] = "DIST cm";    //****** LCD
  GPIO_PinOutClear(gpioPortE,3);    // turn LED1 off
  GPIO_PinOutSet(gpioPortE,2);    // turn LED0 on
  SegmentLCD_Write(msg1);
}
```

```
int Timer1OverFlow() // if timer1 overflow
{
  if(((TIMER1->IF) & (1<<0))==1) //TIMER 1 OVER FLOW
           {
             TIMER_IntClear(TIMER1,1); // Clear overflow flag
             timerInit0.enable=true;
             TIMER_Init(TIMER0, &timerInit0);
             GPIO_PinOutSet(gpioPortD,2);    //set pin D2
             flagtimer11=1;
           }
  return flagtimer11;
}
/* --------------------------------------------------------------
/* --------------------------------------------------------------
/* --------------------------------------------------------------
int Timer0_OverFlow(int flagtimer11) // if timer0 overflow
{
  if(((TIMER0->IF) & (1<<0))==1)
       {
         GPIO_PinOutClear(gpioPortD,2);    //CLEAR pin D2
         TIMER_IntClear(TIMER0,1); // Clear overflow flag
         timerInit0.enable=false;
         TIMER_Init(TIMER0, &timerInit0);
         if (flagtimer11==1)
           {
             flagtimer00=1;
           }
       }
  return flagtimer00;
}
```

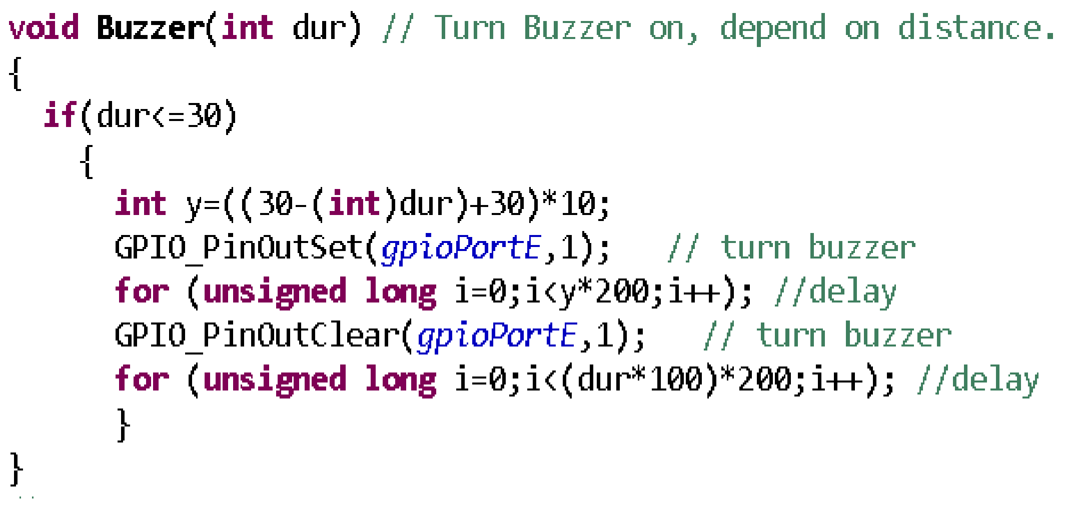
```
uint16_t get_echo(void)
{
  uint16_t echo_start,echo_end,echo_ticks;

//   TIMER2->CC->CCV = 0; //reset timer0 value
   TIMER2->CNT = 0; //reset timer0 value
   TIMER2->CC->CTRL &= ~(1<<24); //wait for rising edge
   TIMER2->CMD = 1; //start timer
   while ((TIMER2->IF & (1<<4)) ==0);  //wait till CC0 flag is set
   echo_start = TIMER2->CC->CCV;

   TIMER2->IFC=1<<4; //clear cc0 int flag
   TIMER2->CC->CTRL |= 1<<24; //wait for falling edge
   while ((TIMER2->IF & (1<<4)) ==0);  //wait till CC0 flag is set
   echo_end = TIMER2->CC->CCV;

   TIMER2->IFC=1<<4; //clear cc0 int flag
   TIMER2->CMD = 2; //stop timer

   //calculate pulse length in timer ticks
   if (echo_end<echo_start)
     echo_ticks=0xffff-echo_start+echo_end;
   else
     echo_ticks=echo_end-echo_start;

   return (uint16_t)echo_ticks;

}

int calulte_distance(int echo_ticks)
{
  float echo_duration, dist_cm;
  //calculate in us
  echo_duration=(float)echo_ticks/14;
  dist_cm=echo_duration/58;

  return (uint16_t)dist_cm;
}
```



```
void print_screen(int dur) // Remove the last digit (only display the 3 digit number)
{
  SegmentLCD_Number(dur);
  LCD->SEGD1H&=~(1<<6);
  LCD->SEGD2H&=~(1<<6);
  LCD->SEGD3H&=~(1<<6);
  LCD->SEGD4H&=~(1<<6);
  LCD->SEGD5H&=~(1<<6);
  LCD->SEGD6H&=~(1<<6);
  LCD->SEGD7H&=~(1<<6);
}
```



```
/   ---------------------------------------------
void Usart1_Send(int dur) // Send letter (Usart1)
{
  if(dur<=30)
    USART1->TXDATA='H';
}
```



```
void Buzzer(int dur) // Turn Buzzer on, depend on distance.
{
  if(dur<=30)
    {
      int y=((30-(int)dur)+30)*10;
      GPIO_PinOutSet(gpioPortE,1);    // turn buzzer
      for (unsigned long i=0;i<y*200;i++); //delay
      GPIO_PinOutClear(gpioPortE,1);    // turn buzzer
      for (unsigned long i=0;i<(dur*100)*200;i++); //delay
      }
}
```

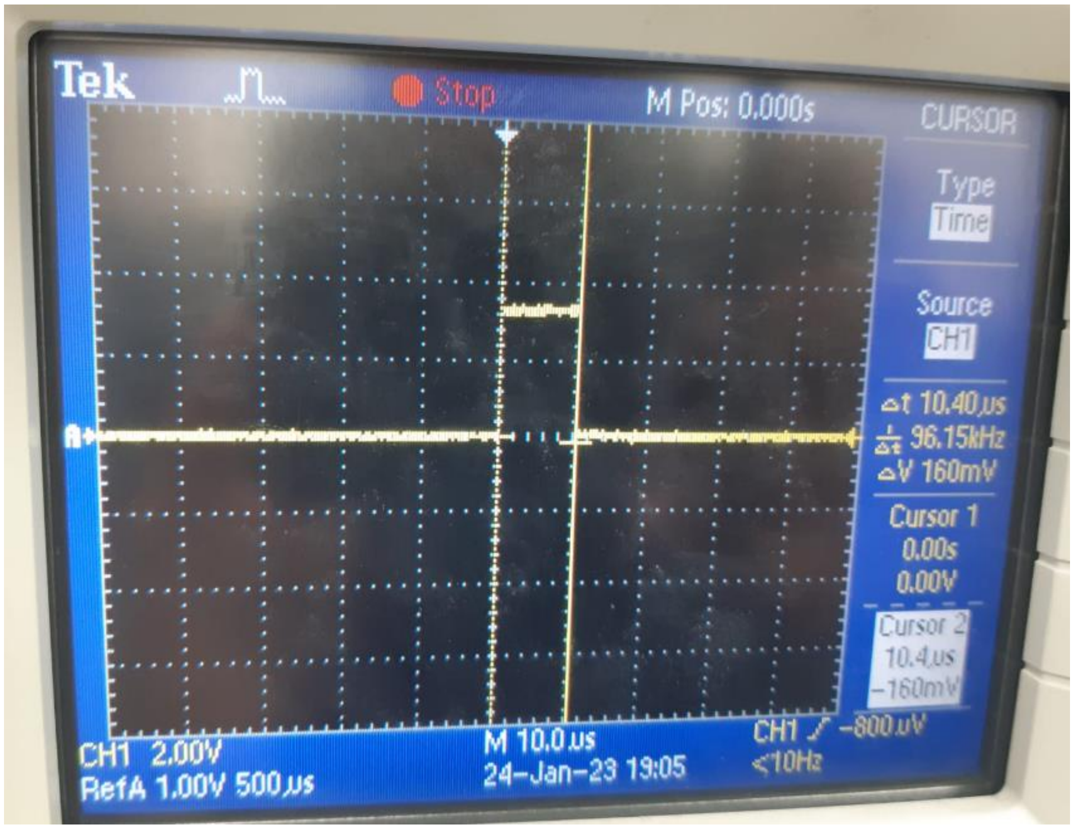
צילום של תו המצוקה שנקלט בטלפון הנייד דרך ערוץ BT:

We choose to send the letter ‘H’, we can see it in the phone in green color.

```
[illegible] Disconnected
[illegible] Connecting to HC-06
[illegible] Connected
[illegible] [illegible]
```

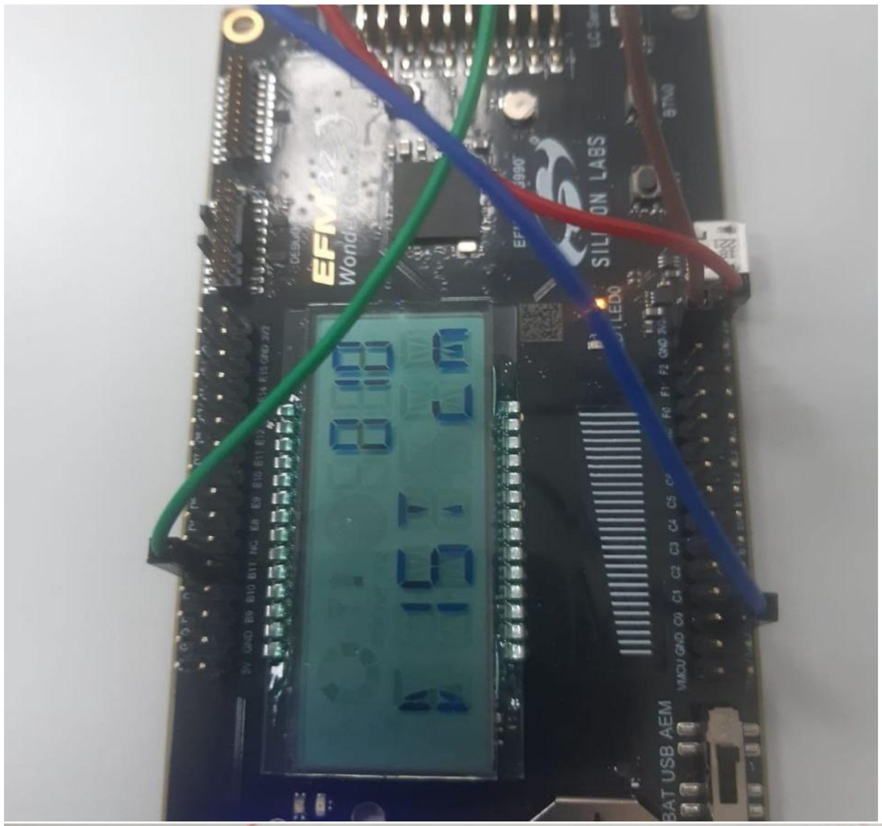
צילומי מסך של הסקופ:

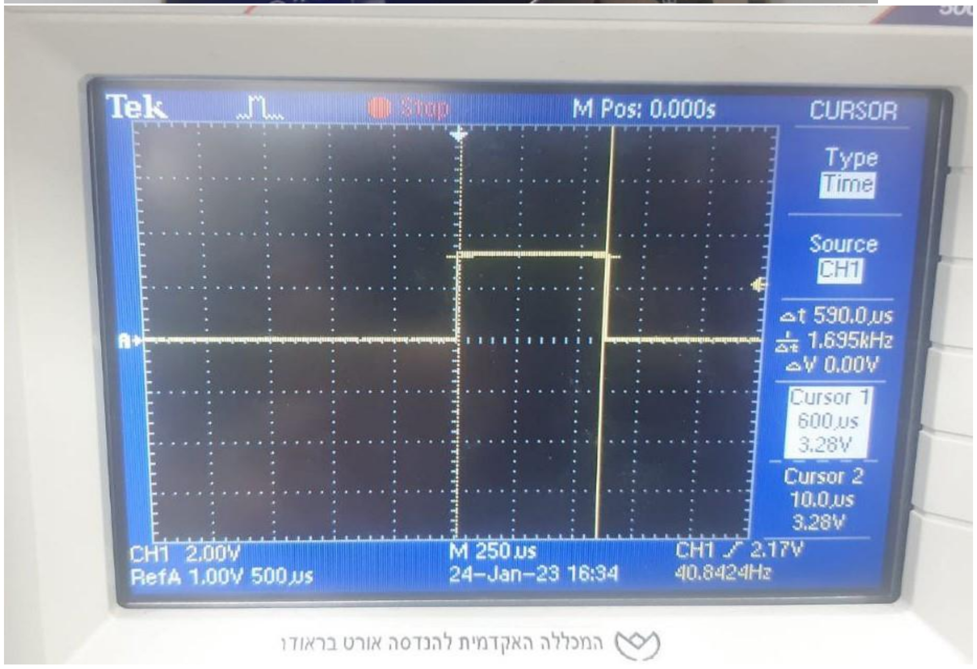
Trigger pulse:



We can see this pulse is 10.4us

Echo pulses:

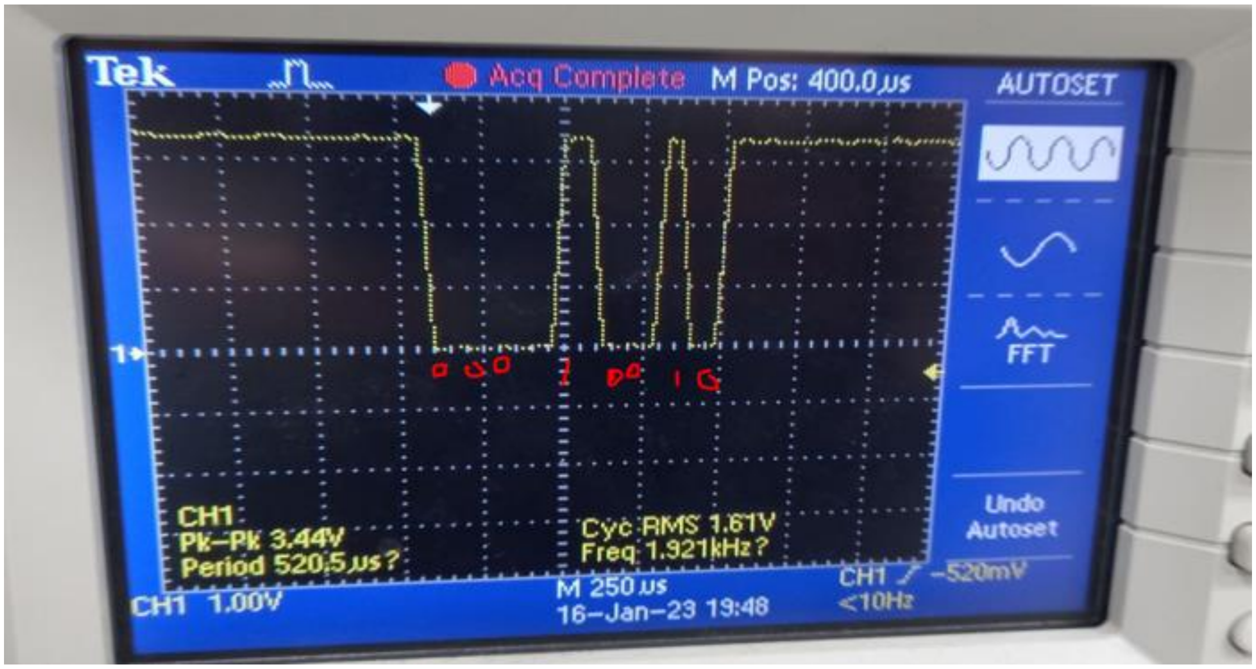




We can see that the distance is 10 cm in the screen, we got a pulse of 590us (Delta).

590/58=10.17cm

USART pulse:



שידרנו אות H

‘H’ in ascii = 48H = 01001000 B

תיאור בעיות:

תצוגת אות הTRIGGER בסקופ:

בהתחלה קבלנו אות רעש בסקופ, ניסנו לשנות בקוד שלנו, אבל היינו בטוחים שזה נכון, בסופו של דבר שיננו את דרגת המדידה (SCALE) בסקופ כך שלא יופיע אות רעש.

קוד echo לא עובד עם TRIGGER:

הייתה לנו בעיה בתוכנית שהקוד נתקע בecho , בהתחלה לא הבנו מאיזו סיבה, אחר כך גילנו שהתוכנית נכנסת ל echo לפני שתסיים Trigger, אז סמנו דגלים מתאימים כך שנדע שהסתיים ה TRIGGER