המחלקה להנדסת
חשמל ואלקטרוניקה

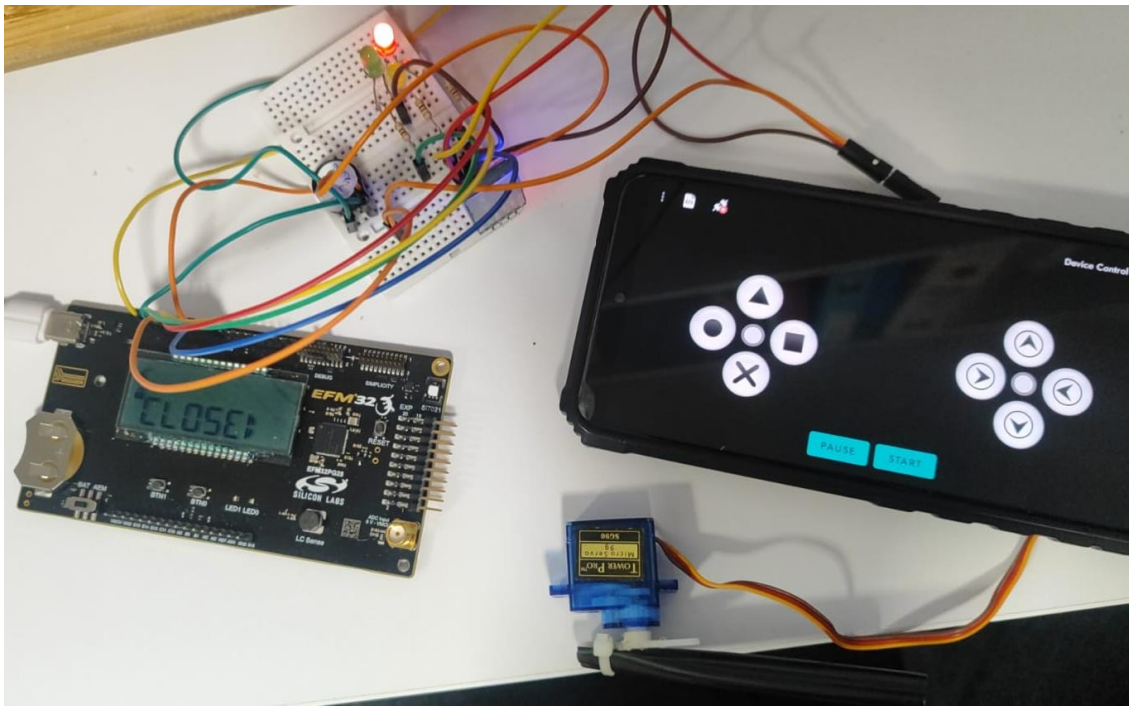
המכללה
האקדמית להנדסה
בראדה
בכרמיאל

מיקרו-בקרים,
31226

דו"ח מסכם – פרויקט סוף במיקרו-בקרים
שער כניסה ויציאה חשמלי

מוגש ע"י: יובל המר, 209158518

מוגש ע"י: עידו בן הרוש, 316439116



בהנחיית: ד"ר פאדל טריף

תאריך: 30/01/25

תוכן עניינים

מבוא:

בפרויקט זה ברצוננו לממש מערכת מבוקרת לכניסה ויציאה של הולכי רגל למבנה מסוים, תוך שימוש במיקרו בקר ARM, תקשורת טורית המבוססת על קישוריות Bluetooth, מנוע סרוו (Servo), ותצוגת LCD הבנויה בבקר עצמו.

השער, אשר ישמש ככניסה ויציאה, נפתח בעת קליטת פקודת כניסה ומציג את כמות האנשים אשר נכנסו דרך השער על גבי תצוגת ה-LCD.

על ידי לחיצה על כפתור בבקר המערכת תעבור ממצב של תצוגת סטטוס השער בנקודת הזמן הנוכחית אל מצב תצוגת כמות האנשים שנכנסו.

במצב של יציאה, המשתמש ילחץ על כפתור היציאה ובהתאם תתעדכן כמות האנשים הנוכחית שנכנסו דרך השער.

במידה וכמות הנכנסים בשער הגיעה לקיבולת המרבית אשר הוגדרה מראש (וניתנת לשינוי לפי דרישה), תוצג הודעה מתאימה למשתמש והשער יישאר נעול.

כמו כן, במידה ולא נכנסו אנשים דרך השער והמשתמש לחץ על כפתור היציאה, תוצג הודעה מתאימה במסך ה-LCD והשער לא ייפתח.

בנוסף, ובמטרה לשלוט בשער במצבי חירום, המערכת כוללת כפתור נוסף שבעת לחיצה עליו יפתח את השער באופן ידני ויישאר פתוח עד ללחיצה נוספת. בעת לחיצה על כפתור זה, יופעל זמזם (המדמה אזעקה) ומונה האנשים אשר נכנסו דרך השער יאופס, מה שמאפשר גישה יעילה ומבוקרת.

כמו כן, נשתמש בנורות LED בצבעים אדום, צהוב וירוק למתן אינדיקציה למצב הפעולה של השער בזמן אמת (אדום – סגור, צהוב – בתהליך פתיחה/סגירה, ירוק – פתוח)

השימוש בתקשורת מבוססת Bluetooth מבטיח שליטה ובקרה מדויקת של אנשים אשר רוצים לצאת או להיכנס דרך השער, בעוד שהמיקרו-בקר מנהל את פעולות השער, הכפתורים, תצוגת ה-LCD, נורות ה-LED והזמזם.

פרויקט זה מדגים פתרון מעשי לשליטה נגישה במערכת, המשלב רכיבי חומרה ותוכנה לחוויית משתמש חלקה.

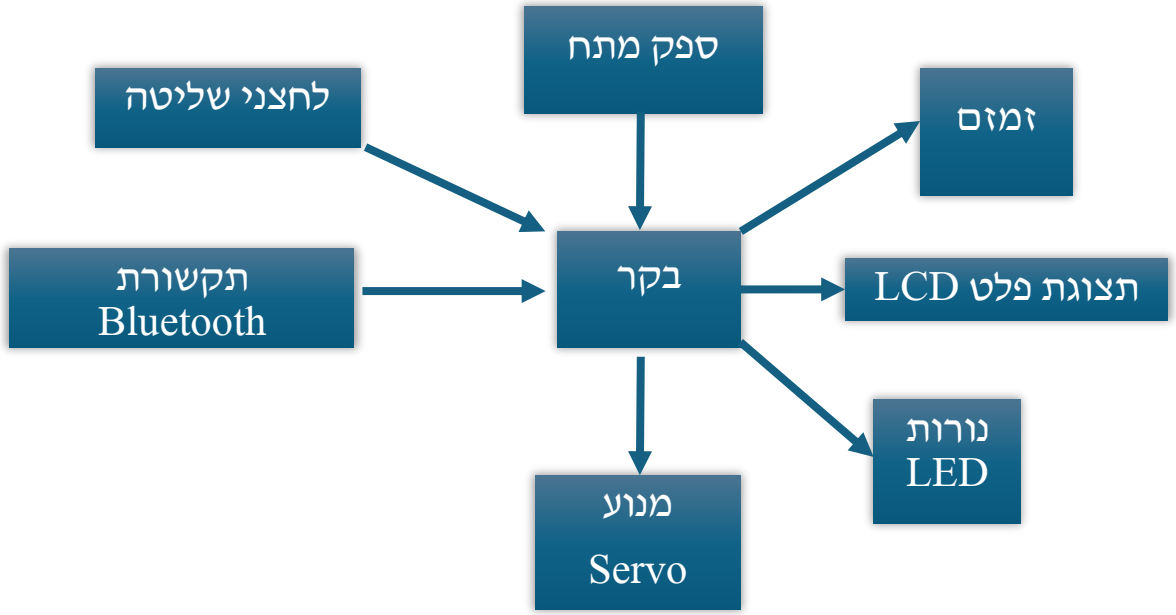
פירוט דרישות המערכת:

המערכת המנוהלת ע״י בקר ARM מסוג EFM32PG28 תהיה בעלת התכונות הבאות:

1. הפעלת סרוו (Servo) למען פתיחה וסגירה מבוקרת של שער הכניסה/יציאה.
2. בעזרת תקשורת Bluetooth, המשתמש יוכל לפתוח ולסגור את השער, בהתאם לכיוון הליכתו (יציאה או כניסה).
3. בעזרת תצוגת ה-LCD המובנית בבקר, נציג את מצב פעולת השער מבין המצבים הבאים:
   - סגור (closed)
   - פתוח (open)
   - בתהליך סגירה (closing)
   - בתהליך פתיחה (opening)
4. בתצוגת ה-LCD נוכל לראות גם את כמות האנשים אשר נכנסו דרך השער.
5. בעזרת שימוש בלחצן המובנה בבקר (PB1) נוכל לעבור ממצב של תצוגת מצב השער לתצוגת כמות האנשים שנכנסו דרך השער.
6. תינתן אפשרות לשלוט בפתיחה השער בעזרת לחצן נוסף המובנה בבקר (PB6).
7. בעת לחיצה על לחצן PB6, המערכת תיכנס למצב חירום בו השער ייפתח והזמזם יופעל. מצב החירום יסתיים רק לאחר לחיצה נוספת על הלחצן הנ״ל.
8. עבור הגעה לכמות מקסימלית אשר הוגדרה מראש (וברת שינוי), לא תינתן אפשרות למשתמש להיכנס דרך השער (כלומר המקום מלא).
9. במידה והמשתמש לחץ על כפתור היציאה אך אין אנשים שנכנסו דרך השער, תוצג הודעה מתאימה למשתמש והשער יישאר נעול (המקום ריק).
10. נורות LED בצבעים שונים יהוו אינדיקציה לסטטוס השער בזמן אמת.

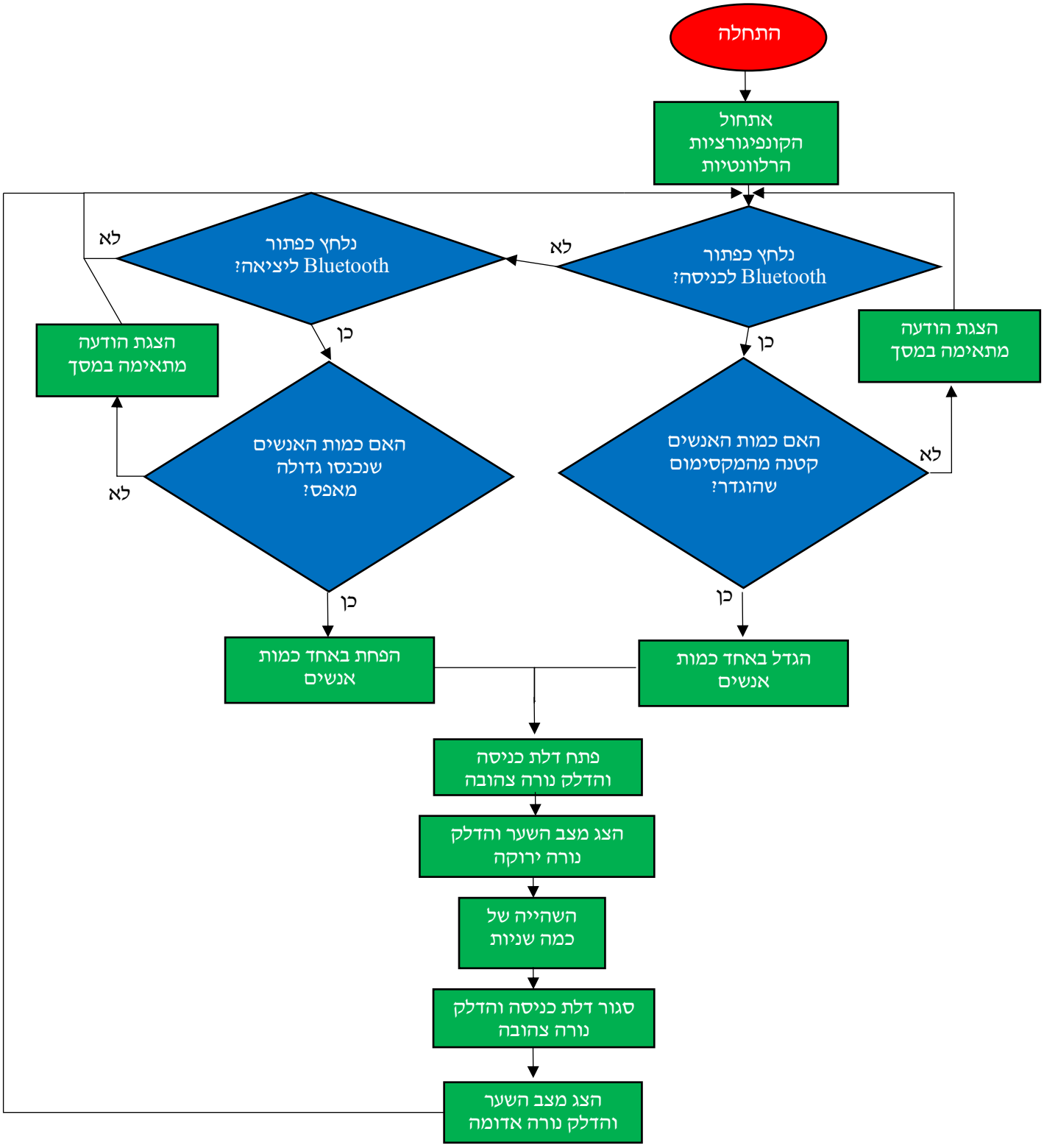
תיאור המערכת:

דיאגרמת בלוקים פונקציונלית:

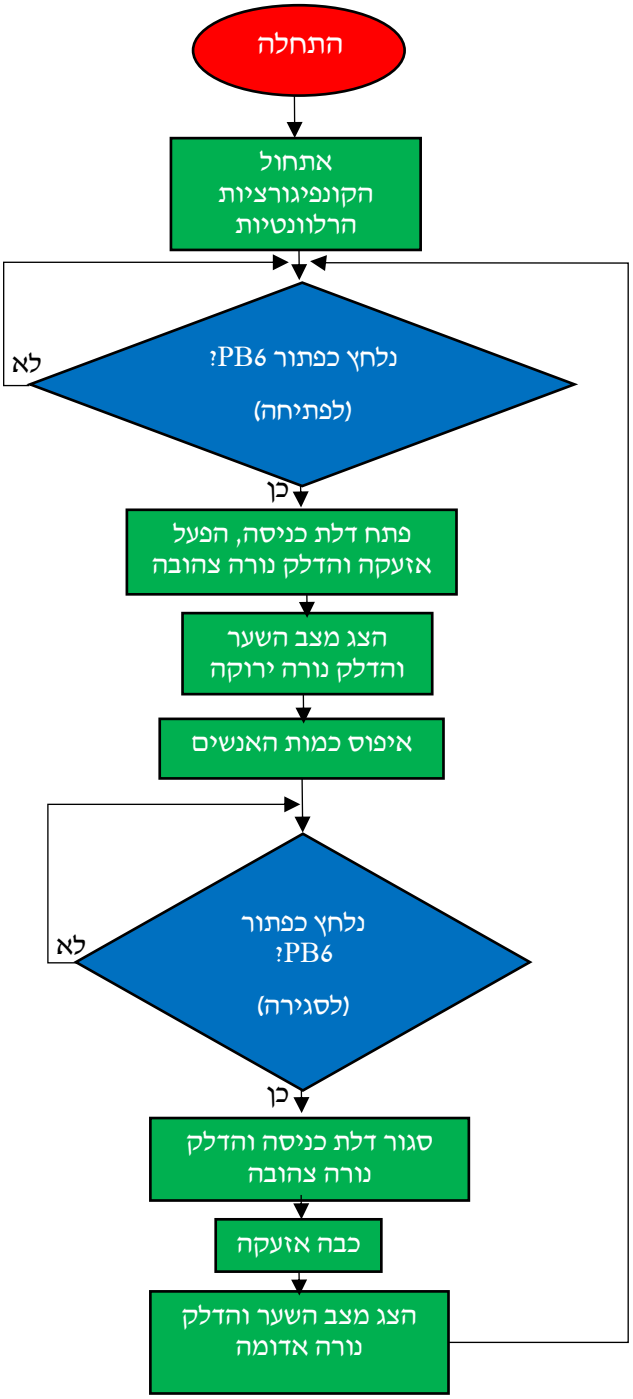


תרשים זרימה עקרוני:

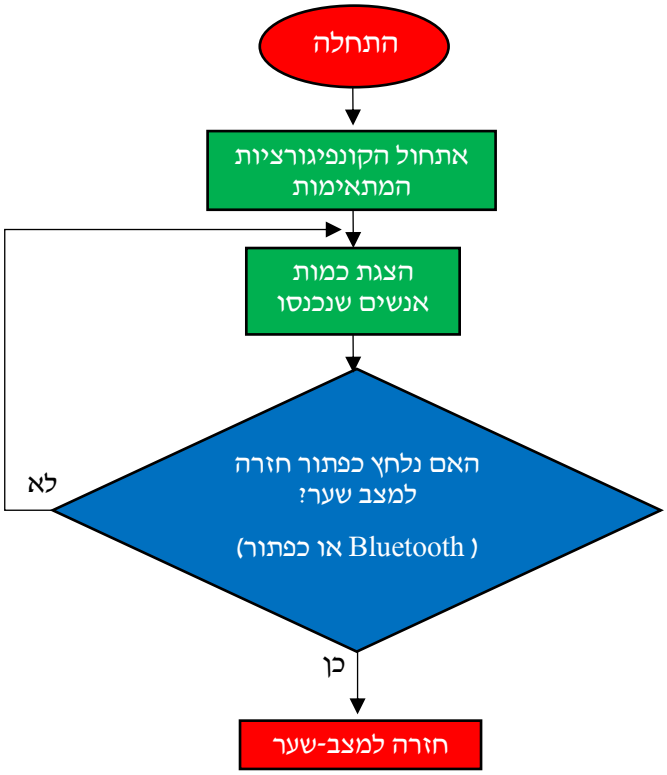
תקשורת Bluetooth - מצב שער:



שימוש בלחצנים - מצב שער:



שימוש בלחצנים - מצב תצוגה:



- ניתן לעבור ממצב תצוגה למצב שער ולהפך בעת המתנה לפקודה מתקשורת ה-Bluetooth או בעת המתנה ללחיצה על לחצן הפתיחה המובנה בבקר.

חלוקת משאבים:

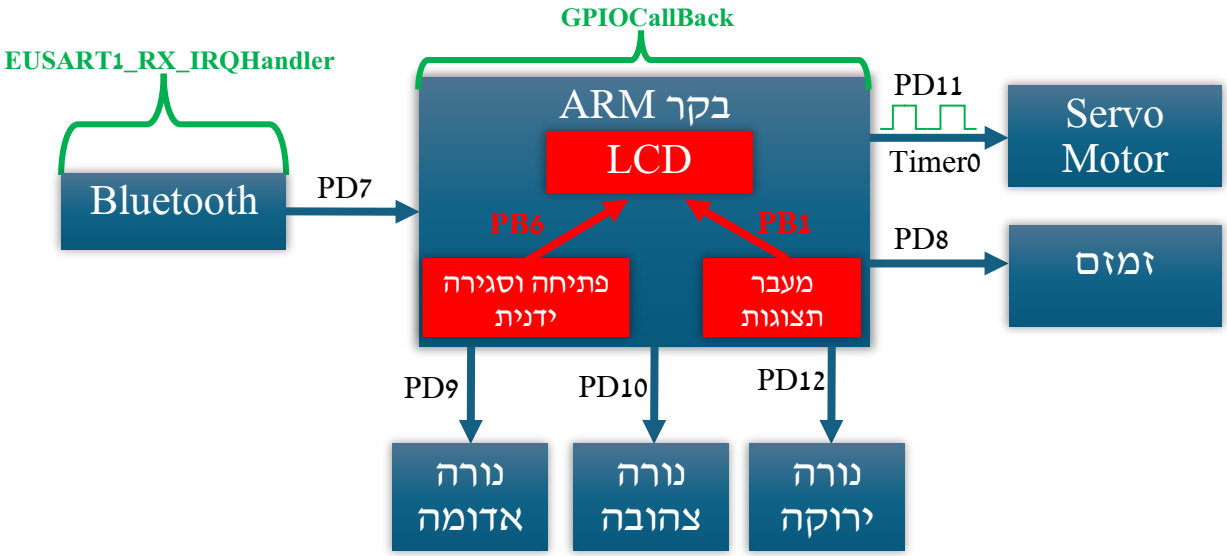
LCD ולחצנים:

	פורט הפין	קלט/פלט	אנלוגי/דיגיטלי	שימוש
GPIO	PD7	קלט	דיגיטלי	קריאת אות Bluetooth
	PD8	פלט		הפעלת זמזם
	PD9	פלט		הפעלת LED אדום
	PD10	פלט		הפעלת LED צהוב
	PD11	קלט		יצירת אות PWM למנוע Servo
	PD12	פלט		הפעלת LED ירוק
	PB1	קלט		מעבר בין תצוגות
	PB6	קלט		פתיחה וסגירת שער ידנית
LCD	LCD	פלט	-	תצוגת מצב השער וכמות הנכנסים

פסיקות וטיימרים:

	שם	ארגומנטים	שימוש
פסיקות	EUSART1_RX_IRQHandler	-	אחראית על שליטה בשער בעזרת Bluetooth
	GPIOCallBack	uint8_t pin	אחראית על פסיקה במידה ונלחץ לחצן בבקר (פתיחה וסגירה ידנית או מעבר בין תצוגות)
טיימר	Timer0 (Channel 0)	-	מטרתו ליצור אות PWM להזזת המנוע

תרשים מלבנים להצגת חלוקת משאבים:



קוד הפרויקט[1]:

ייבוא ספריות:

```
/* Library includes */
#include "em_device.h"      // Device-specific definitions
#include "em_chip.h"        // Chip initialization and configuration
#include "em_cmu.h"         // Clock Management Unit functions
#include "em_gpio.h"        // GPIO configuration and control
#include "em_timer.h"       // Timer configuration and control
#include "em_emu.h"         // Energy Management Unit for power modes
#include "em_eusart.h"      // UART communication functions
#include "sl_segmentlcd.h" // Segmented LCD control functions
#include "gpiointerrupt.h" // GPIO interrupt handling
#include "stdbool.h"        // Boolean type and logic
#include "sl_udelay.h"      // Microsecond delay utility
```

[1] במערכת זו לא מומשה הפונקציה app_process_action ולא נעשה שינוי בפונקציית ה-main של התוכנית.

הגדרות משתני Define והגדרות פינים:

```
/*****************************************************************************
 * System Configuration Constants
 *****************************************************************************/
/* PWM Configuration */
#define PWM_FREQ            1000    // PWM frequency in Hz
#define INITIAL_DUTY_CYCLE  60      // Initial PWM duty cycle (%)
#define TIMER1_MS           15      // 15ms delay for debouncing

/* GPIO Port and Pin Definitions */
#define BUTTONS_PORT        gpioPortB
#define SERVO_PORT          gpioPortD
#define UART_PORT           gpioPortD
#define BUZZER_PORT         gpioPortD
#define LED_PORT            gpioPortD
#define SERVO_PIN           11      // Pin for signal output for the servo motor
#define PERSON_SHOW         1       // Pin for persons display
#define CLOSE_BUTTON_PIN    6       // Pin for closing the gate
#define UART_RX_PIN         7       // UART receive pin

/* System Constants */
#define MAX_CAPACITY       2        // Maximum number of persons allowed
#define TRANSITION_DELAY   1000000 // Delay for state transitions (in microseconds)
#define DISPLAY_DELAY      20000000 // Delay for displaying messages (in microseconds)
#define UART_BAUD_RATE     9600     // UART communication speed

/* Servo positions */
#define SERVO_PERIOD   780000 // Timer top value
// 20ms period (50Hz) for 39MHz clock (0.02*39M)

// For 1ms pulse (0 degrees) = 39M * 0.77ms ≈ 30000 (Pulse Width = x/Clock Frequency)
#define SERVO_CLOSED   30000    // 0.77ms pulse for 0 degrees

// For 1.6ms pulse (90 degrees) = 39M * 1.6ms) ≈ 62000 (Pulse Width = x/Clock Frequency)
#define SERVO_OPEN     62000    // 1.6ms pulse for 90 degrees

/* Bluetooth Command Codes */
#define ENTER_GATE         'F'     // Command to increment person count
#define EXIT_GATE          'B'     // Command to decrement person count

/* Gate States */
#define STATE_CLOSED        6       // Gate fully closed state
#define STATE_OPEN          1       // Gate fully open state
#define NO_BUTTON_PRESSED   0xFF    // Indicates no button is currently pressed

/*LED & Buzzer Pins*/
#define RED 9
#define YELLOW 10
#define GREEN 12
#define BUZZER 8
#define ON 1
#define OFF 0
```

הגדרות משתנים סטטיים:

```
/*****************************************
 * Static Variables
 *****************************************
static volatile int state = STATE_CLOSED;
static volatile int PersonsInside = 0;
static volatile bool Emergency = false;
```

הכרזות על פונקציות אתחול ועזר:

```
/*******************************************************
 * Function Prototypes
 *******************************************************
static void GPIOCallback(uint8_t pin);
void displayTransitionState(const char* stateMessage);
int checkButtonState(void);
void GateHandling(void);
void MaxMinCapacity(char* s);
void CLOSED_LCD(void);
void OPEN_LCD(void);
void LCD_CONFIGURATIONS(void);
void BUTTONS_CONFIGURATIONS(void);
void BLUETOOTH_INIT(void);
void Servo_Init(void);
void openGate(void);
void closeGate(void);
void BuzzerOn(void);
void BuzzerOff(void);
void LED_Handling(int red, int yellow, int green);
```

מימוש הפונקציות:

פסיקת GPIOCallBack:

```
 * Callback Functions
 ******************************************************************************/
static void GPIOCallback(uint8_t pin) {
    // Static variable to track display mode for Button 1
    // true = show person count, false = show gate state
    static bool showNumber = true;
    // Ensure Gecko symbol is always visible on LCD
    sl_segment_lcd_symbol(SL_LCD_SYMBOL_GECKO, 1);

    // Handle Open Button (Button 1) press
    if (pin == PERSON_SHOW) {
        if (showNumber) {
            // Display Mode 1: Show current number of people inside
            sl_segment_lcd_number(PersonsInside);
            sl_segment_lcd_symbol(SL_LCD_SYMBOL_GECKO, 1);
        } else {
            // Display Mode 2: Show current gate state
            if (state == STATE_OPEN) {
                OPEN_LCD();    // Display "OPEN" if gate is open
            } else if (state == STATE_CLOSED) {
                CLOSED_LCD();  // Display "CLOSED" if gate is closed
            }
        }
        // Toggle between display modes for next button press
        showNumber = !showNumber;
    }
    // Handle Close Button press
    else if (pin == CLOSE_BUTTON_PIN) {
        // Reset person counter when gate operation is initiated
        PersonsInside = 0;
        Emergency = true; // we are at emergency situation
        if (state == STATE_CLOSED) {
            // If gate is closed, initiate opening sequence
            state = STATE_OPEN;
            BuzzerOn(); // turns on the buzzer
            displayTransitionState("OPENING");  // Show transition animation
            openGate();
            OPEN_LCD();                         // Update display to show "OPEN"
        } else if (state == STATE_OPEN) {
            // If gate is open, wait for close button press
            while (state == STATE_OPEN) {
                state = checkButtonState();    // Poll for button state change
            }
            // Initiate closing sequence
            state = STATE_CLOSED;
            BuzzerOff(); // turns off the buzzer
            closeGate();
            displayTransitionState("CLOSING"); // Show transition animation
```

הסבר פסיקת GPIOCallBack:

הפונקציה (פסיקה) הנ"ל אחראית על מעבר בין תצוגות ה-LCD (סטטוס השער וכמות האנשים שנכנסו) בעת לחיצה על אחת מהלחצנים המובנים בבקר.

בעת לחיצה על לחצן התצוגה (PB1) המערכת תציג את כמות האנשים שנכנסו דרך השער. לחיצה נוספת על לחצן זה תחזיר את ה-LCD למצב תצוגת השער.

בעת לחיצה על לחצן השער (PB6) המערכת תיכנס למצב חירום ותפעיל אזעקה.

בעת כניסה למצב חירום, השער ייפתח ויישאר במצב זה עד אשר תתקבל לחיצה נוספת על הלחצן הנ"ל.

בנוסף, כמות האנשים שהיו בתוך השער תתאפס (כיאה למצב חירום – כולם יצאו מן השער).

מצב החירום יסתיים כאשר תתקבל לחיצה נוספת על הלחצן הנ"ל.

פונקציית displayTransitionState:

```
void displayTransitionState(const char* stateMessage) {
    LED_Handling(OFF,ON,OFF);                    // Turns on the yellow LED
    sl_segment_lcd_write(stateMessage);          // Write the transition message to LCD
    sl_segment_lcd_symbol(SL_LCD_SYMBOL_GECKO, 1); // Ensure Gecko symbol remains visible
    sl_udelay_wait(TRANSITION_DELAY);            // Hold the message for defined delay period
```

הסבר פונקציית displayTransitionState:

הפונקציה אחראית על הצגת מעבר כניסה/יציאה בתצוגת ה-LCD.

הפונקציה מקבלת מחרוזת להצגה ב-LCD (משתנה בין OPENING ל-CLOSING, תלוי במצב), אותה תציג זמן מוגדר וקבוע (שימוש בפונקציה מובנית sl_udelay_wait) ולאחר מכן המערכת תמשיך בפעולתה. בעת הצגת המחרוזת המערכת תפעיל נורת LED צהובה המעידה על שינוי מצב משער פתוח לסגור או להפך.

פונקציית checkButtonState:

```
int checkButtonState(void) {
    // Check if person show button is pressed (logic low indicates press)
    if (GPIO_PinInGet(BUTTONS_PORT, PERSON_SHOW) == 0) {
        return STATE_OPEN;
    }
    // Check if close button is pressed (logic low indicates press)
    if (GPIO_PinInGet(BUTTONS_PORT, CLOSE_BUTTON_PIN) == 0) {
        return STATE_CLOSED;

    }
    // No button is currently pressed
    return NO_BUTTON_PRESSED;
}
```

הסבר פונקציית checkButtonState:

הפונקציה אחראית על מתן אינדיקציה איזה לחצן נלחץ בבקר.

הפונקציה נקראת בפסיקת ה-GPIOCallBack בתוך לולאה אינסופית עד אשר נלחץ לחצן השער (PB6), בעת לחיצה על הלחצן הנ"ל המערכת תצא ממצב החירום אליו נכנסה קודם לכן.

פונקציית GateHandling:

```
void GateHandling(void) {
    // Begin opening sequence
    displayTransitionState("OPENING");    // Show opening transition message
    openGate();
    OPEN_LCD();                           // Display gate open state
    sl_udelay_wait(TRANSITION_DELAY);     // Wait for specified delay

    // Begin closing sequence
    displayTransitionState("CLOSING");    // Show closing transition message
    closeGate();
    CLOSED_LCD();                         // Display gate closed state
}
```

הסבר פונקציית GateHandling:

הפונקציה אחראית על העברת השער בין המצבים שהוגדרו קודם לכן (פתוח, סגור, בתהליך פתיחה/סגירה). לאחר שהשער נפתח, המערכת מחכה זמן מוגדר מראש לפני שנסגרת באופן אוטומטי. נדגיש כי הפונקציה משתמשת בפונקציות אחרות (יפורטו בהמשך) להפעלת המנוע המדמה את השער.

פונקציית Servo_Init:

```
 * PWM Servo Configuration
 ******************************************************************************/
void Servo_Init(void) {
    // Enable clocks
    CMU_ClockEnable(cmuClock_GPIO, true);
    CMU_ClockEnable(cmuClock_TIMER0, true);

    // Configure servo pin as push-pull output
    GPIO_PinModeSet(SERVO_PORT, SERVO_PIN, gpioModePushPull, 0);

    // Configure Timer for PWM
    TIMER_Init_TypeDef timerInit = TIMER_INIT_DEFAULT;
    timerInit.prescale = timerPrescale1;
    timerInit.enable = false;

    TIMER_InitCC_TypeDef timerCCInit = TIMER_INITCC_DEFAULT;
    timerCCInit.mode = timerCCModePWM;

    // Initialize timer
    TIMER_Init(TIMER0, &timerInit);
    TIMER_InitCC(TIMER0, 0, &timerCCInit);

    // Set PWM period (50Hz)
    TIMER_TopSet(TIMER0, SERVO_PERIOD);

    // Configure PWM routing to GPIO pin
    GPIO->TIMERROUTE[0].ROUTEEN = GPIO_TIMER_ROUTEEN_CC0PEN;
    GPIO->TIMERROUTE[0].CC0ROUTE = (SERVO_PORT << _GPIO_TIMER_CC0ROUTE_PORT_SHIFT)
                                 | (SERVO_PIN << _GPIO_TIMER_CC0ROUTE_PIN_SHIFT);

    // Set initial position (closed)
    TIMER_CompareSet(TIMER0, 0, SERVO_CLOSED);

    // Enable timer
    TIMER_Enable(TIMER0, true);
}
```

הסבר פונקציית Servo_Init:

פונקציה זו אחראית על אתחול מנוע הסרוו בו אנו משתמשים ויצירת אות PWM תוך שימוש בטיימר 0.

הפונקציה תחילה מאפשרת את שעון הטיימר ואת ה-GPIO. לאחר מכן הפונקציה מבצעת אתחול של פין PD11 כפין יציאת אות PWM בו נשתמש להפעלת המנוע.

בנוסף, הפונקציה מאתחלת את השער למצב סגור.

פונקציית MaxMinCapacity:

```
void MaxMinCapacity(char* s) {
    sl_segment_lcd_write(s);                        // Write the capacity message
    sl_segment_lcd_symbol(SL_LCD_SYMBOL_GECKO, 1); // Display Gecko symbol
    sl_segment_lcd_symbol(SL_LCD_SYMBOL_PAD0, 1);  // Enable PAD0 indicator
    sl_segment_lcd_symbol(SL_LCD_SYMBOL_PAD1, 1);  // Enable PAD1 indicator
    sl_udelay_wait(DISPLAY_DELAY);                 // Hold message for defined period
    CLOSED_LCD();                                  // Return to closed state display
}
```

הסבר פונקציית MaxMinCapacity:

פונקציה זו אחראית להציג למשתמש על גבי מסך ה-LCD כי לא ניתן להכניס יותר אנשים דרך השער מכיוון שהגענו למקסימום הקיבולת או שלא ניתן לצאת מן השער כי לא נכנסו אנשים.
השער יישאר סגור לאחר הצגת ההודעות הנ"ל.

פונקציית CLOSED_LCD:

```
void CLOSED_LCD(void) {
    LED_Handling(ON,OFF,OFF); // turns on the red LED
    sl_segment_lcd_write("CLOSED");              // Display "CLOSED" text
    sl_segment_lcd_symbol(SL_LCD_SYMBOL_GECKO, 1); // Show Gecko symbol
    sl_segment_lcd_symbol(SL_LCD_SYMBOL_PAD0, 1);  // Enable PAD0 for locked indication
    sl_segment_lcd_symbol(SL_LCD_SYMBOL_PAD1, 1);  // Enable PAD1 for locked indication
}
```

הסבר פונקציית CLOSED_LCD:

פונקציה זו אחראית על הצגת המילה "CLOSED" על גבי תצוגת ה-LCD, בנוסף להצגת מנעול וסמל לטאה על התצוגה. בנוסף, הפונקציה תקרא לפונקציה LED_Handling להצגת אור אדום המסמל שער סגור.

פונקציית OPEN_LCD:

```
void OPEN_LCD(void) {
    LED_Handling(OFF,OFF,ON); // turns on the green LED
    sl_segment_lcd_write("OPEN");                  // Display "OPEN" text
    sl_segment_lcd_symbol(SL_LCD_SYMBOL_GECKO, 1); // Show Gecko symbol only
}
```

הסבר פונקציית OPEN _LCD:

פונקציה זו אחראית על הצגת המילה "OPEN" על גבי תצוגת ה-LCD, בנוסף להסרת סמל המנעול.
בנוסף, הפונקציה תקרא לפונקציה LED_Handling להצגת אור ירוק המסמל שער פתוח.

פונקציית BuzzerOn:

```
void BuzzerOn(void) // Turns on the buzzer
{
  GPIO_PinModeSet(BUZZER_PORT, BUZZER, gpioModePushPull, 1);
}
```

הסבר פונקציית BuzzerOn:

הפונקציה אחראית מתן מתח חיובי לפין PD8 לצורך הפעלת הזמזם המדמה אזעקה בעת כניסה למצב חירום.

פונקציית BuzzerOff:

```
void BuzzerOff(void) // Turns off the buzzer
{
  GPIO_PinModeSet(BUZZER_PORT, BUZZER, gpioModePushPull, 0);
}
```

הסבר פונקציית BuzzerOff:

הפונקציה אחראית על כיבוי הזמזם באמצעות הפסקת המתח שניתן לו.

פונקציית LED_Handling:

```
void LED_Handling(int red, int yellow, int green) // Turns on specific LED
{
  GPIO_PinModeSet(LED_PORT, RED, gpioModePushPull, red);
  GPIO_PinModeSet(LED_PORT, YELLOW, gpioModePushPull, yellow);
  GPIO_PinModeSet(LED_PORT, GREEN, gpioModePushPull, green);
}
```

הסבר פונקציית LED_Handling:

הפונקציה אחראית על מתן מתח לפין הרצוי במטרה להפעיל את נורת ה-LED המתאימה לפעולת השער. פונקציה זו מקבלת שלושה פרמטרים (שערכיהם 0 או 1) המהווים אינדיקציה איזה נורה יש להפעיל. נציין כי לפי האלגוריתם הכולל של המערכת, בכל פעם תידלק נורה אחת בלבד, בהתאם למצב פעולת המערכת.

פונקציית LCD_CONFIGURATIONS:

```
void LCD_CONFIGURATIONS(void) {
    sl_segment_lcd_init(false);  // Initialize LCD in normal power mode
    CLOSED_LCD();               // Set initial display state to closed
}
```

הסבר פונקציית LCD_CONFIGURATIONS:

הפונקציה אחראית על אתחול תצוגת ה-LCD המובנית בבקר במצב שימוש רגיל באנרגיה.
בנוסף, הפונקציה מאתחלת את התצוגה למצב של שער סגור.

פונקציית BUTTONS_CONFIGURATIONS:

```
void BUTTONS_CONFIGURATIONS(void) {
    // Enable required peripheral clocks
    CMU_ClockEnable(cmuClock_GPIO, true);  // Enable GPIO clock
    CMU_ClockEnable(cmuClock_LCD, true);   // Enable LCD clock

    // Configure button pins as inputs with pull-up resistors
    // Pull-up means pin reads high when button is not pressed
    GPIO_PinModeSet(BUTTONS_PORT, PERSON_SHOW, gpioModeInputPull, 1);      // Person show button
    GPIO_PinModeSet(BUTTONS_PORT, CLOSE_BUTTON_PIN, gpioModeInputPull, 1); // Close button

    // Initialize GPIO interrupt handling
    GPIOINT_Init();  // Initialize GPIO interrupt module

    // Configure external interrupts for both buttons
    // Parameters: port, pin, interrupt number, rising edge, falling edge, enable
    GPIO_ExtIntConfig(BUTTONS_PORT, PERSON_SHOW, PERSON_SHOW, false, true, true);            // Person show button
    GPIO_ExtIntConfig(BUTTONS_PORT, CLOSE_BUTTON_PIN, CLOSE_BUTTON_PIN, false, true, true); // Close button

    // Register callback function for button interrupts
    GPIOINT_CallbackRegister(PERSON_SHOW, GPIOCallback);      // Register person show button callback
    GPIOINT_CallbackRegister(CLOSE_BUTTON_PIN, GPIOCallback); // Register close button callback
}
```

הסבר פונקציית BUTTONS_CONFIGURATIONS:

פונקציה זו אחראית על אתחול וקונפיגורציה מתאימה של הלחצנים המובנים בבקר.

באתחולים ניתן למצוא בין היתר הקצאת הפינים הרלוונטיים, מתן שעון ל-GPIO והגדרת פסיקות עבור לחצנים אלו. בנוסף ניתן לראות שיוך הפונקציה CallBack[2] (שראינו קודם לכן) עבור הלחצנים הנ"ל.

פונקציית BLUETOOTH_INIT:

```
void BLUETOOTH_INIT(void) {
    // Enable required peripheral clocks
    CMU_ClockEnable(cmuClock_GPIO, true);      // Enable GPIO clock
    CMU_ClockEnable(cmuClock_EUSART1, true);   // Enable UART clock

    // Configure UART receive pin
    GPIO_PinModeSet(UART_PORT, UART_RX_PIN, gpioModeInput, 0);

    // Initialize UART with default high-frequency configuration
    EUSART_UartInit_TypeDef init = EUSART_UART_INIT_DEFAULT_HF;
    init.baudrate = UART_BAUD_RATE;        // Set communication speed
    init.oversampling = eusartOVS4;        // 4x oversampling for better reliability

    // Configure UART pin routing
    GPIO->EUSARTROUTE[1].RXROUTE = (UART_PORT << _GPIO_EUSART_RXROUTE_PORT_SHIFT)
                                 | (UART_RX_PIN << _GPIO_EUSART_RXROUTE_PIN_SHIFT);
    GPIO->EUSARTROUTE[1].ROUTEEN = GPIO_EUSART_ROUTEEN_RXPEN;  // Enable receive pin routing

    // Initialize UART with configured parameters
    EUSART_UartInitHf(EUSART1, &init);

    // Configure and enable UART interrupts
    NVIC_ClearPendingIRQ(EUSART1_RX_IRQn);    // Clear any pending interrupts
    NVIC_EnableIRQ(EUSART1_RX_IRQn);          // Enable UART receive interrupts
    EUSART_IntEnable(EUSART1, EUSART_IEN_RXFL); // Enable UART receive interrupt flag
}
```

[2] פונקציה המועברת כפרמטר לפונקציה אחרת ומבוצעת מאוחר יותר בתגובה לאירוע או תנאי מסוים.

הסבר פונקציית BLUETOOTH_INIT:

פונקציה זו אחראית על אתחול וקונפיגורציה מתאימה של התקשורת הטורית (UART) מבוססת Bluetooth למען שליטה בפעולת השער.

בין האתחולים, ניתן למצוא בין היתר מתן שעון ל-GPIO ול-EUSART, הקצאה וניתוב פין PD7 כפין המקבל נתונים ממודל ה-Bluetooth, אתחול קצב השידור לערך קבוע מוגדר מראש ואתחול פסיקת תקשורת בווקטור הפסיקות (NVIC).

פסיקת EUSART1_RX_IRQHandler:

```c
void EUSART1_RX_IRQHandler(void) {

    // Read the received data from Bluetooth
    uint8_t receivedData = EUSART_Rx(EUSART1);


    if (!Emergency) // Available only if not in emergency situation
      {
        switch(receivedData) {     // Handle received data based on its value

            case ENTER_GATE: // Command to enter the gate
                // Check if the current number of persons inside is less than the maximum capacity
                if(PersonsInside < MAX_CAPACITY) {
                    PersonsInside++; // Increment the count of persons inside
                    GateHandling(); // Handle gate operations
                } else {
                    MaxMinCapacity("FULL-UP"); // Notify that the gate is full
                }
                break;

            case EXIT_GATE: // Command to exit the gate
                // Check if there are persons inside to exit
                if(PersonsInside > 0) {
                    PersonsInside--; // Decrement the count of persons inside
                    GateHandling(); // Handle gate operations
                } else {
                    MaxMinCapacity("EMPTY"); // Notify that there are no persons inside
                }
                break;
        }
      }
    // Clear the interrupt flag for receive events
    EUSART_IntClear(EUSART1, EUSART_IF_RXFL);
}
```

הסבר פסיקת EUSART1_RX_IRQHandler:

פסיקה זו אחראית על הפעלת השער במידה וניתנה פקודה דרך תקשורת ה-Bluetooth וכאשר המערכת **אינה** נמצאת במצב חירום.

המערכת מקבלת נתונים (תו 'F' לכניסה או תו 'B' ליציאה), במידה והתקבל תו 'F' הדבר אומר כי אדם רוצה להיכנס דרך השער. הפונקציה בודקת האם הגענו למקסימום הקיבולת שהוגדרה בראש התוכנית. במידה ולא, מעדכנת את כמות האנשים שנכנסו ופותחת את השער. אחרת, תוצג הודעה שהמקום מלא ולא ניתן להיכנס.

במידה והתקבל תו 'B', הפונקציה תבדוק האם המקום ריק. במידה וכן, תוצג הודעה כי המקום ריק ולא ניתן לצאת ממנו לפני שנכנס לפחות אדם אחד. במידה והמקום אינו ריק, השער ייפתח וכמות האנשים תרד באחד. לבסוף, ננקה את דגל הפסיקה.

פונקציית openGate:

```
void openGate(void) {
  // Gradually set the servo to the open position (90 degrees)

  // Incrementally adjust the servo pulse width from half-open to fully open
  for(int i = SERVO_OPEN/2; i <= SERVO_OPEN; i++)
    {
      TIMER_CompareSet(TIMER0, 0, i); // Update PWM duty cycle for the servo
      sl_udelay_wait(5);              // Small delay for smoother movement
    }
  sl_udelay_wait(200000);  // Delay for making gate remains fully open
}
```

הסבר פונקציית openGate:

הפונקציה אחראית על הזזה אקטיבית של השער ממצב סגור לפתוח. בעזרת הלולאה, אנו משנים את אורך פולס ה-PWM למנוע הסרוו בשביל פתיחה איטית ומבוקרת יותר של השער. לאחר שהשער נפתח, הפונקציה מבצעת השהייה של כ-2 שניות נוספות במטרה לתת למשתמש לעבור דרך השער.

פונקציית closeGate:

```
void closeGate(void) {
    // Set servo to closed position (0 degrees)
    TIMER_CompareSet(TIMER0, 0, SERVO_CLOSED);
}
```

הסבר פונקציית closeGate:

הפונקציה אחראית על הזזה אקטיבית של השער ממצב פתוח לסגור.

פונקציית app_init:

```
void app_init(void) {

    // Configure the LCD display for the application
    LCD_CONFIGURATIONS();

    // Configure button inputs
    BUTTONS_CONFIGURATIONS();

    // Initialize Bluetooth module
    BLUETOOTH_INIT();

    // Initialize Servo motor
    Servo_Init();
}
```

הסבר פונקציית app_init:

אחראית על קריאה כלל פונקציות האתחול שתוארו קודם לכן. נציין כי פונקציה זו, כמו גם פונקציות האתחול האחרות, נקראות פעם אחת בלבד בתחילת הרצת התוכנית.

חישובים והסברים:

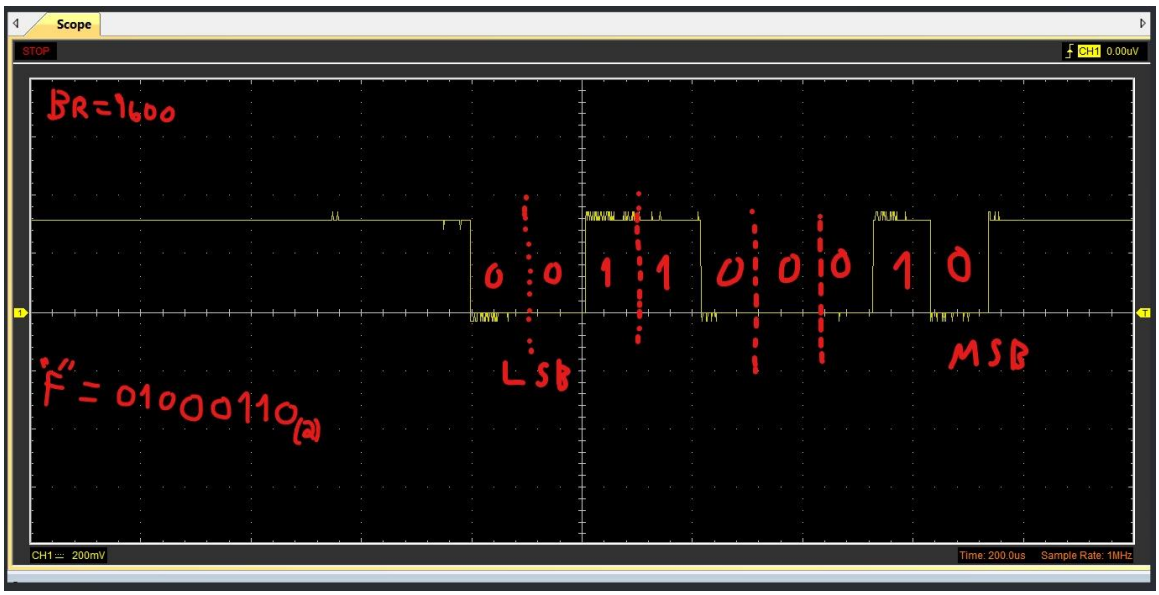
תקשורת EUART:

כזכור, השתמשנו במודל בלוטות' למימוש בתקשורת טורית מסוג EUART (Enhanced UART).

כלומר, הבקר קיבל פקודות דרך הבלוטות' לטובת פתיחה של השער.

נרצה למדוד ולוודא כי הערכים המתקבלים בתקשורת הבלוטות' אכן נכונים.

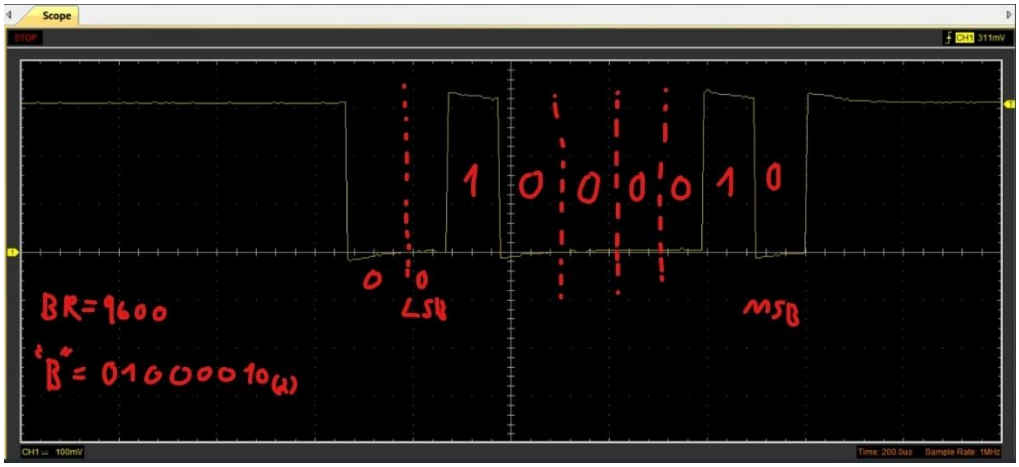
להלן האות שהתקבל בעת לחיצה על לחצן 'F' כפי שנמדד באוסילוסקופ:



קידוד האות 'F' בייצוג בינארי הינו 01000110. ניתן לראות כי קיבלנו תחילה ביט אפס מוביל (השמאלי ביותר) המעיד על התחלת שידור (ללא נתונים).

לאחר מכן ניתן לראות את הקידוד של האות 'F' בייצוגה הבינארי החל מסיבית ה-LSB עד לסיבית ה-MSB באופן רציף, תוך מרווח של $\frac{1}{BR}$ כאשר $BR = 9600$ ומסמל את כמות הסיביות שנשלחו לשנייה.

להלן האות שהתקבל בעת לחיצה על לחצן 'B' כפי שנמדד באוסילוסקופ:



קידוד האות 'B' בייצוג בינארי הינו 01000010. ניתן לראות כי גם עבור פקודה זו קיבלנו תחילה ביט אפס מוביל (השמאלי ביותר) המעיד על התחלת שידור (ללא נתונים).

לאחר מכן ניתן לראות את הקידוד של האות 'B' בייצוגה הבינארי החל מסיבית ה-LSB עד לסיבית ה-MSB באופן רציף, תוך שמירה על המרווח שצוין קודם לכן $\left(\frac{1}{BR}\right)$.

נסיק מכך כי התקשורת הטורית בין מודול הבלוטות' לבין הבקר עובדת כמצופה.

אות PWM ו-TIMER:

מכיוון שאנו משתמשים במנוע לתזוזת השער, עלינו ליצור אות PWM למנוע בכדי לקבוע פעולה נכונה ומדויקת של השער.

בכדי להשתמש באופן מדויק במנוע הסרוו, עלינו ליצור אות PWM ברוחב 20ms (50Hz).

לשם כך, נגדיר את ערך ה-TOP של הטיימר להיות תוצאת החישוב הבא:

$\underbrace{39MHz}_{Clock\ Frequency} \cdot \underbrace{20ms}_{Pulse\ Width} = 780000_{DEC} = BE6E0_{HEX}$ כלומר, הטיימר יספור עד ערך של 780000.

להלן ערך הרגיסטר TOP כפי שמופיע בתוכנת Simplicity 5:

Register	Address	Value
TOP	[31:0]	0xBE6E0

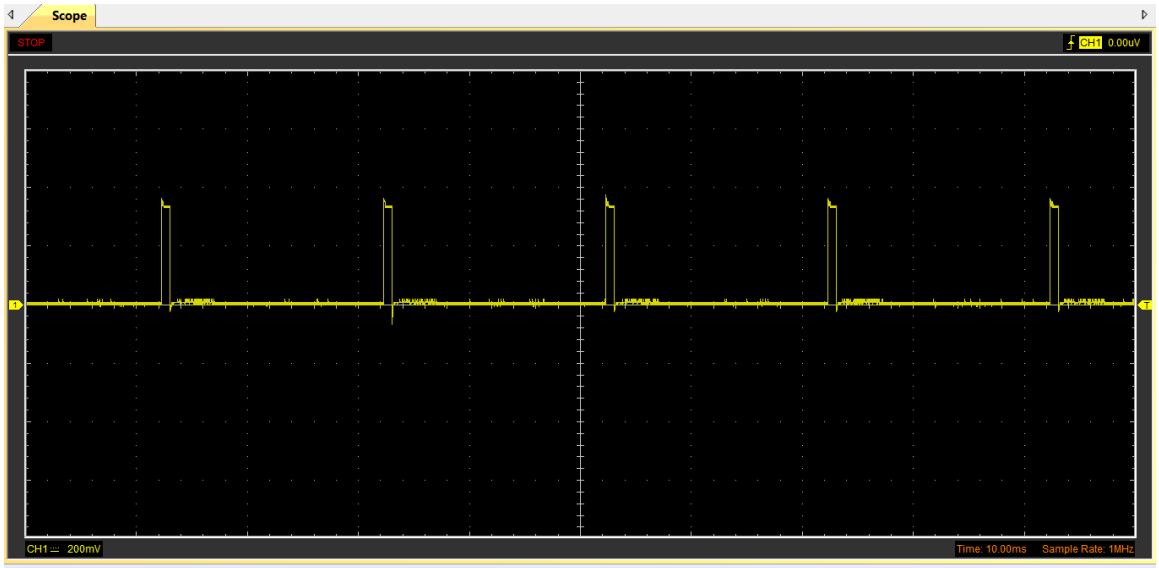
עבור סגירת השער, נרצה כי זרוע המנוע תהיה בזווית של כ-0 מעלות.

לאחר ניסוי וטעיה, ראינו כי עלינו להשתמש בפולס ברוחב 0.77ms (תוך שינוי מיקום הידית המחוברת למנוע) כך שהערך אליו תגיע ספירת המונה המובנה בטיימר יהיה הערך 30000 בקירוב, לפי החישוב הבא: $\underbrace{39MHz}_{Clock\ Frequency} \cdot \underbrace{0.77ms}_{Pulse\ Width} \approx 30000_{DEC} = 7530_{HEX}$.

להלן ערך הרגיסטר CC0_OC[3] כפי שמופיע בתוכנת Simplicity 5:

Register	Address	Value
CC0_OC	0x50048068	0x00007530

[3] רגיסטר המשמש להשוואת ערכים ואירועים הקשורים לאות PWM בטיימר 0



Output

CH1	
Period	20.0mS
Frequency	50.00Hz
+ Duty Cycle	3.8%

להלן אות PWM עבור שער סגור:

ניתן לראות כי קיבלנו אות ברוחב 20ms ועם D.C. של 3.8% בקירוב. כלומר, רוחב הפולס החיובי הינו 0.77ms כמצופה, וזאת לפי החישוב הבא: $20ms \cdot \frac{3.8}{100} \approx 0.77ms$

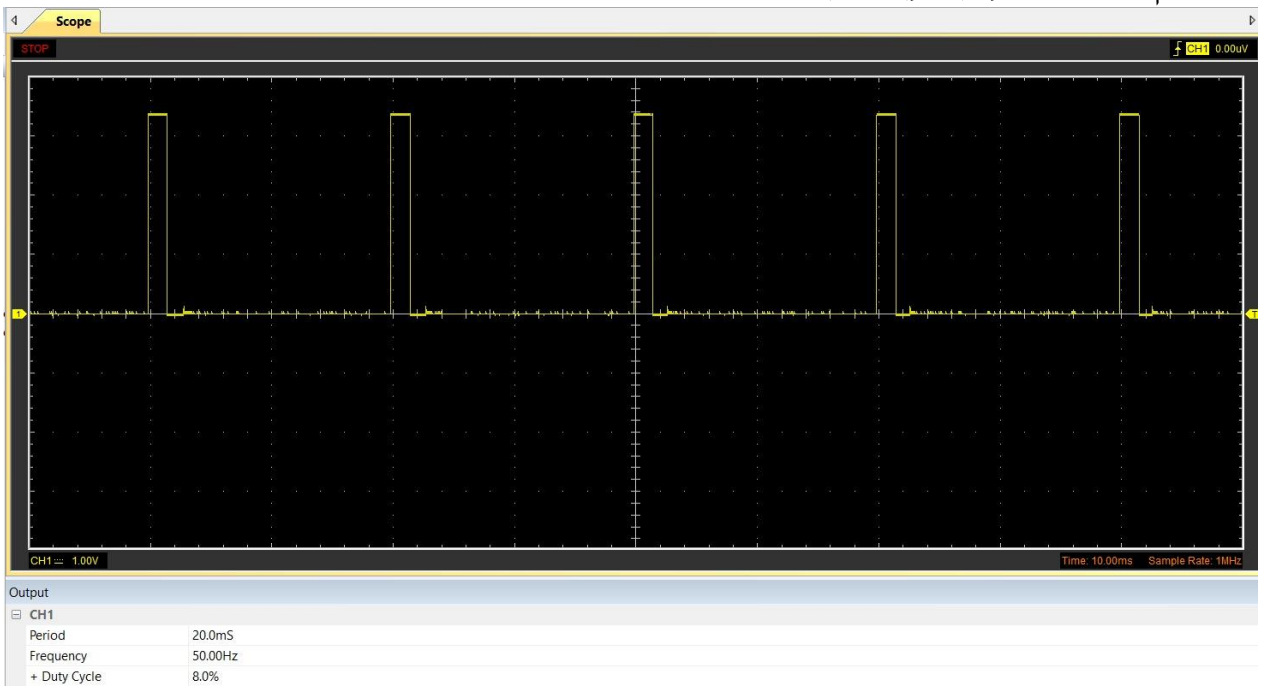
עבור פתיחת השער, נרצה כי זרוע המנוע תהיה בזווית של כ-90 מעלות. לשם כך, עלינו ליצור את האות כך שה-D.C. (Duty Cycle) יהיה ברוחב של 2ms בקירוב (לפי דפי הנתונים).

לאחר ניסוי וטעיה, ראינו כי עלינו להשתמש בפולס ברוחב 1.6ms (תוך שינוי מיקום הידית המחוברת למנוע) כך שהערך אליו תגיע ספירת המונה המובנה בטיימר יהיה הערך 62000 בקירוב, לפי החישוב הבא: $\underbrace{39MHz}_{Clock\ Frequency} \cdot \underbrace{1.6ms}_{Pulse\ Width} \approx 62000_{DEC} = F230_{HEX}$.

להלן ערך הרגיסטר CC0_OC כפי שמופיע בתוכנת Simplicity 5:

Register	Address	Value
CC0_OC	0x50048068	0x0000F230

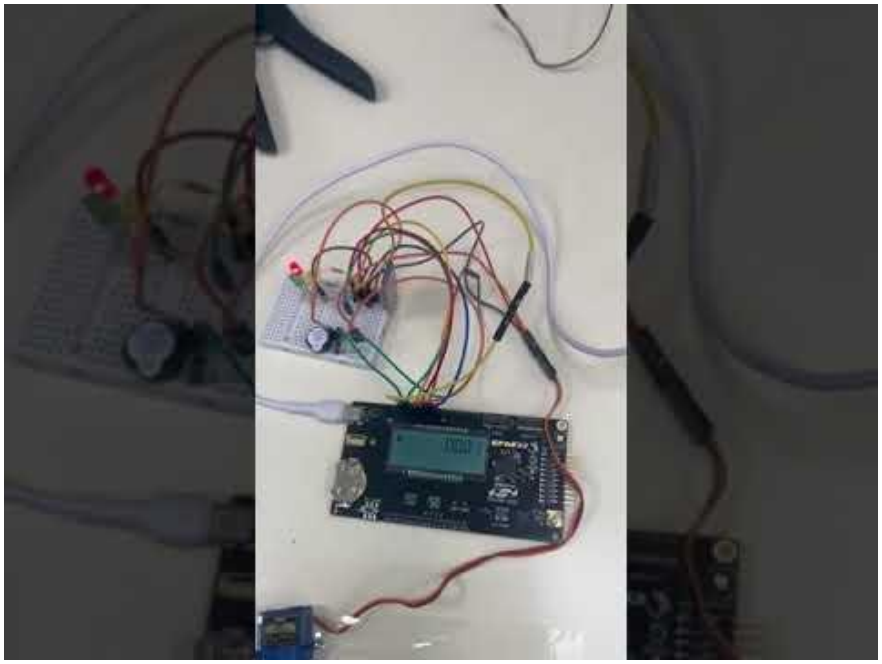
להלן אות PWM עבור שער פתוח:



ניתן לראות כי קיבלנו אות ברוחב 20ms ועם D.C. של 8% בקירוב. כלומר, רוחב הפולס החיובי הינו 1.6ms כמצופה, וזאת לפי החישוב הבא: $20ms \cdot \frac{8}{100} = 1.6ms$

נסיק מכך כי אות הפעלת המנוע PWM עובד כמצופה בשני מצבי פעולת המערכת.

סרטון הדגמת פעולת המערכת:



בעיות ופתרונן:

1. כאשר ראינו את האות המתקבל דרך התקשורת הטורית בעזרת מודול בלוטות', לא ידענו כיצד לפרש אותו ומה הוא מסמל. לאחר חקירה והתייעצות עם מנחה הפרויקט ועם סטודנטים נוספים, הוסבר לנו כיצד יש לקרוא את האות המתקבל (בצורה בינארית כאשר LSB היא הסיבית הכי ימנית)
2. בפרויקט זה התנסינו לראשונה בשימוש במנוע סרוו השונה משימוש במנוע DC. חוסר הידע הנדרש בהפעלת מנוע זה גרם לטעויות בבחירת הערכים הרצויים לתפעול נכון של המערכת. ניסוי וטעיה, תוך קריאת דף הנתונים של מנוע הסרוו לעומק ופנייה למקורות מידע נוספים, הגענו לערכים הרצויים לתפעול נכון של השער.
3. לא ידענו כיצד להשתמש בתצוגת ה-LCD בבקר ARM החדש לנו. על מנת להתגבר על הבעיה, חזרנו על ההרצאות והחומר התיאורטי הנלמד ואף חיפשנו באתר החברה Silicon Labs עבור פונקציות מתאימות לשימוש בפרויקט זה.

ביבליוגרפיה:

1. Silicon Labs LCD API
2. EFM32PG28 MCU Reference Manual
3. Servo Motor Data Sheet
4. כלל החומר הנמצא באתר הקורס

נספחים:

פונקציה sl_udelay_wait:

בפרויקט זה נעשה שימוש בפונקציה **המובנית** sl_udelay_wait אשר מקבלת מספר כארגומנט ומבצעת השהייה כזמן הארגומנט שניתן לה במיקרו שניות. להלן מימוש הפונקציה:

```
void sl_udelay_wait(unsigned us)
{
  uint32_t freq_khz;
  uint32_t ns_period;
  uint32_t cycles;
  uint32_t loops;

  freq_khz = SystemCoreClockGet() / 1000U;
  if (freq_khz == 0) {
    EFM_ASSERT(false);
    return;
  }

  ns_period = 1000000U / freq_khz;
  if (ns_period == 0) {
    EFM_ASSERT(false);
    return;
  }

  cycles = us * 1000U / ns_period;
  loops = cycles / HW_LOOP_CYCLE;
  if (loops > 0U) {
    sli_delay_loop(loops);
  }
}
```

הסבר פונקציית sl_udelay_wait:

פונקציה זו משתמשת ב-busy loop כדי לעכב את ביצוע הקוד במספר מסוים של מיקרו-שניות. הפונקציה לא משתמשת בהתקנים חיצוניים למיניהם לתזמון ההשהיה, אלא היא משתמשת בתדר שעון הליבה (Core Clock) כדי לחשב את ההשהיה. הדיוק של לולאת השהיה זו יושפע בין היתר מפסיקות.

לקריאה נוספת ניתן לקרוא באתר של Silicon Labs - קישור

הפונקציה הנ"ל עושה שימוש בפונקציה sli_delay_loop אשר עליה לא נמצא מידע כלל. ניתן להניח כי הפונקציה אחראית על ביצוע לולאות ההשהיה בעזרת שיטתbusy wait. ייתכן כי מימוש פונקציה זו נעשה בשפת אסמבלי.