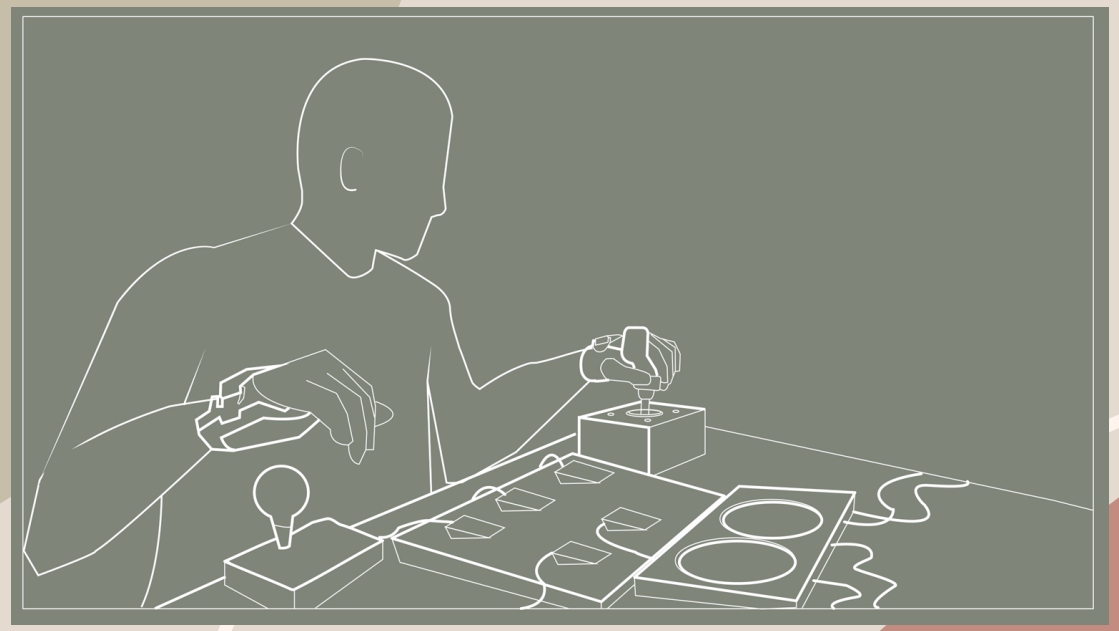
מכון טכנולוגי חולון

Accessible controller



אפיון ותכנון מערכות משובצות מחשב | סמסטר א' 2024

מגישים: חיים עוזר – 316063569 | שון פזרקר – 208426890 | מרצה: ד"ר אביחי אהרון

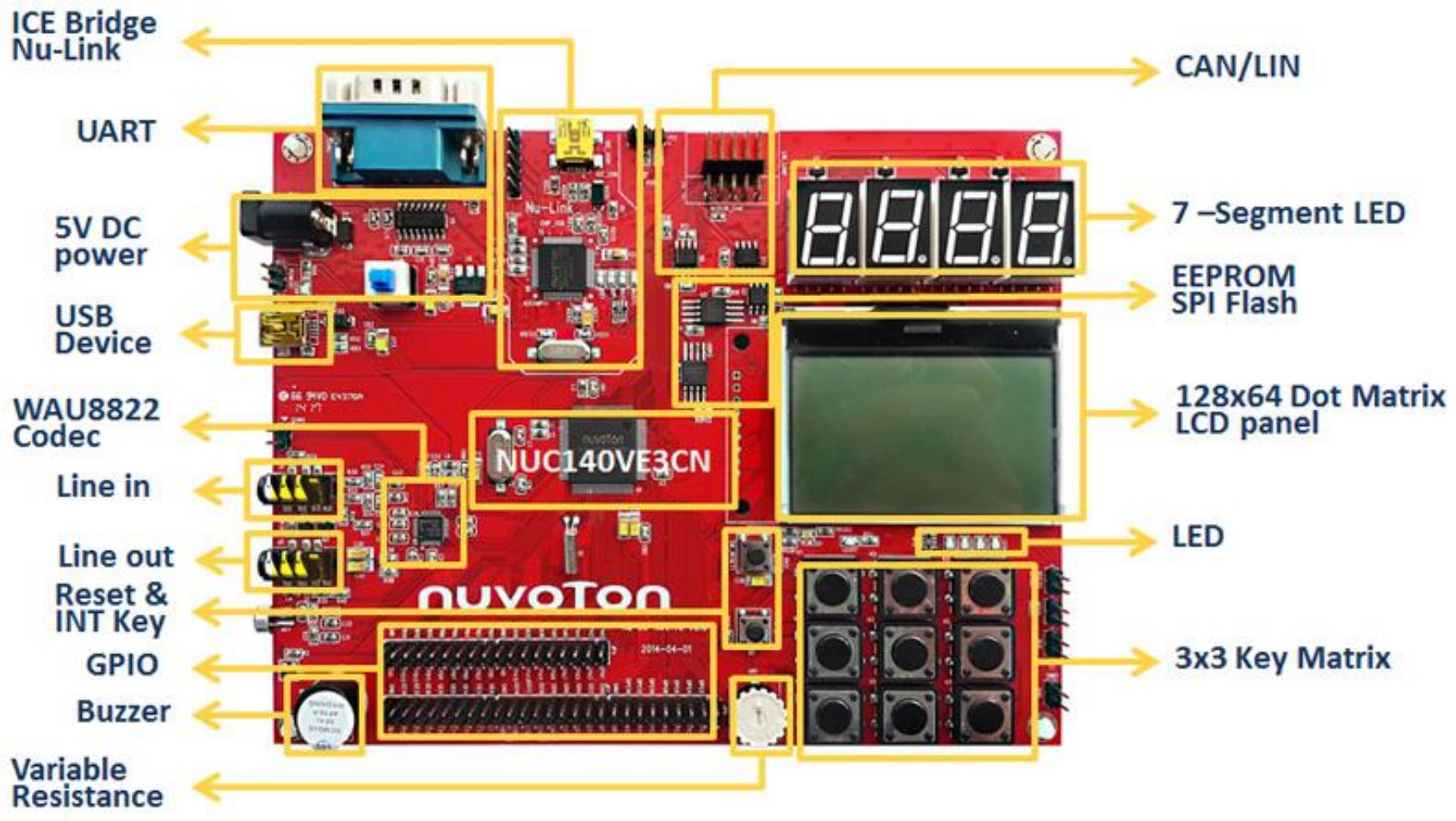
מבוא לפרויקט:

ממשקים ומכשירים קונבנציונליים רבים אינם מתוכננים מתוך מחשבה על הצרכים המגוונים של אנשים עם מוגבלויות.

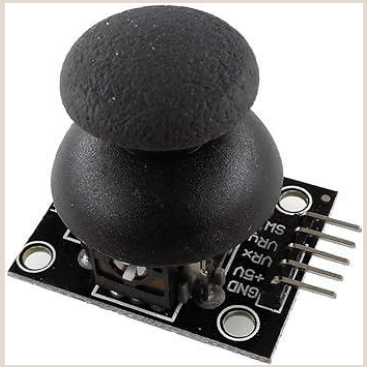
המוטיבציה מאחורי פרויקט הג'ויסטיק לנכים נובעת ממחויבות עמוקה להכלה ולהעצמה. אנשים עם מוגבלויות מתמודדים לעתים קרובות עם אפשרויות מוגבלות בכל הנוגע לגישה ושליטה במכשירים אלקטרוניים, במיוחד אלה עם לקויות ניידות. ג'ויסטיקים מסורתיים עשויים שלא להתאים לצרכים הייחודיים שלהם או שהם עשויים להיות יקרים בצורה בלתי רגילה.

על ידי מינוף היכולות של חיישנים קוליים, נוריות, לחצנים וכרטיס NUVOTON הרב-תכליתי, פרויקט זה מבקש ליצור פתרון נגיש וחסכוני. המטרה היא לפתח ממשק ג'ויסטיק שהוא אינטואיטיבי לשימוש, ניתן להתאמה אישית להעדפות אישיות ומגיב לאתגרי ניידות מגוונים. יוזמה זו מונעת מתוך רצון לשפר את איכות החיים של אנשים עם מוגבלות על ידי מתן עצמאות ושליטה רבה יותר בסביבתם.

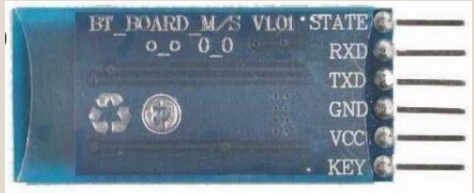
רכיבים בהם השתמשנו:



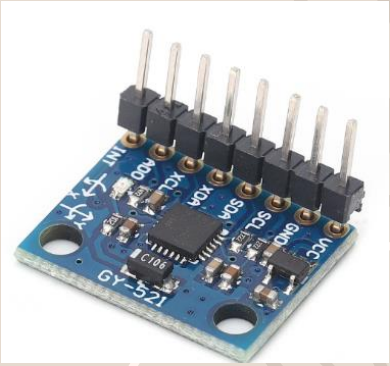
SR - 04
Ultrasonic sensor



HW - 504
Joystick Module



HC-05
Bluetooth Module

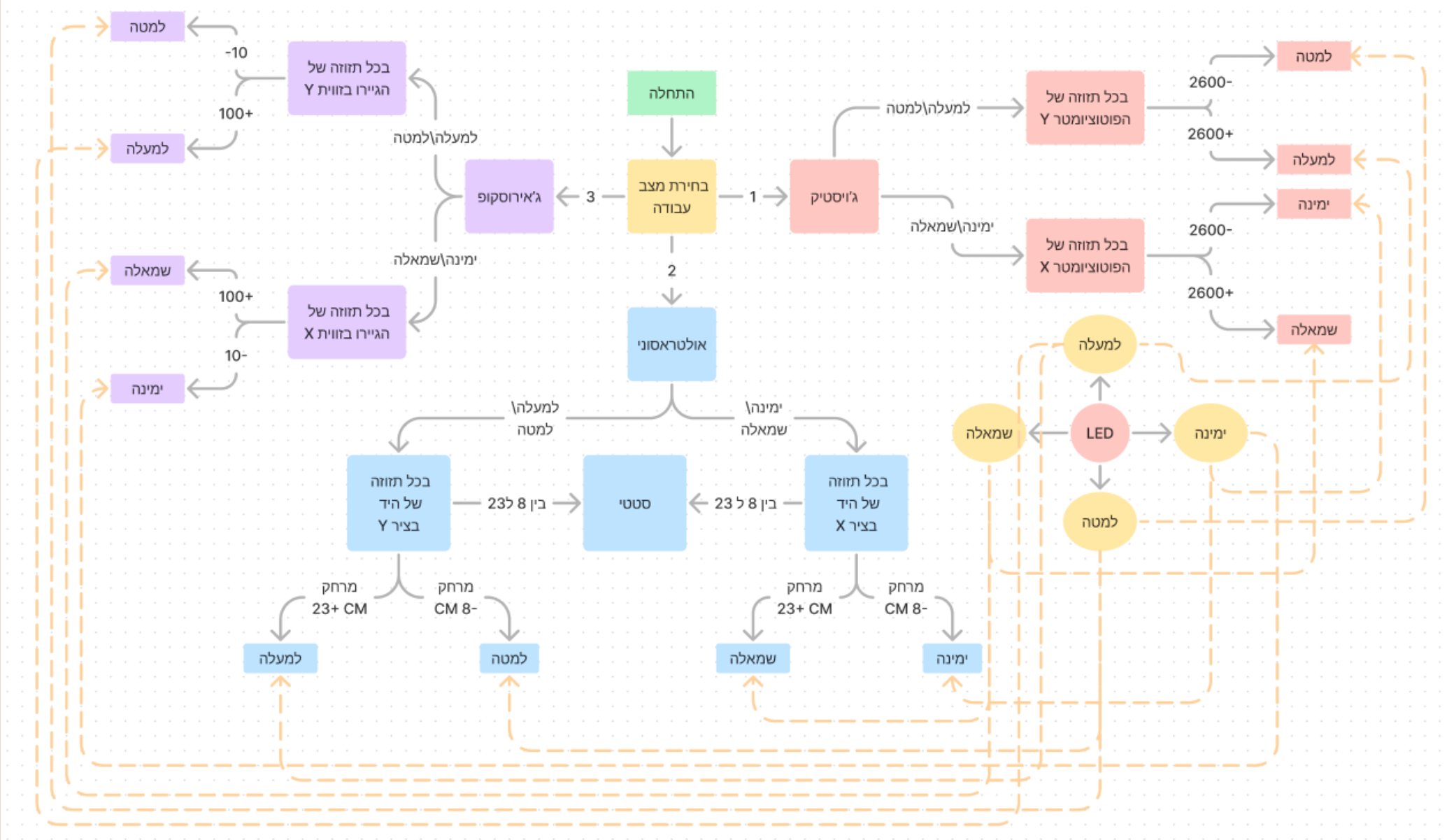


GY-521
Gyroscope Module

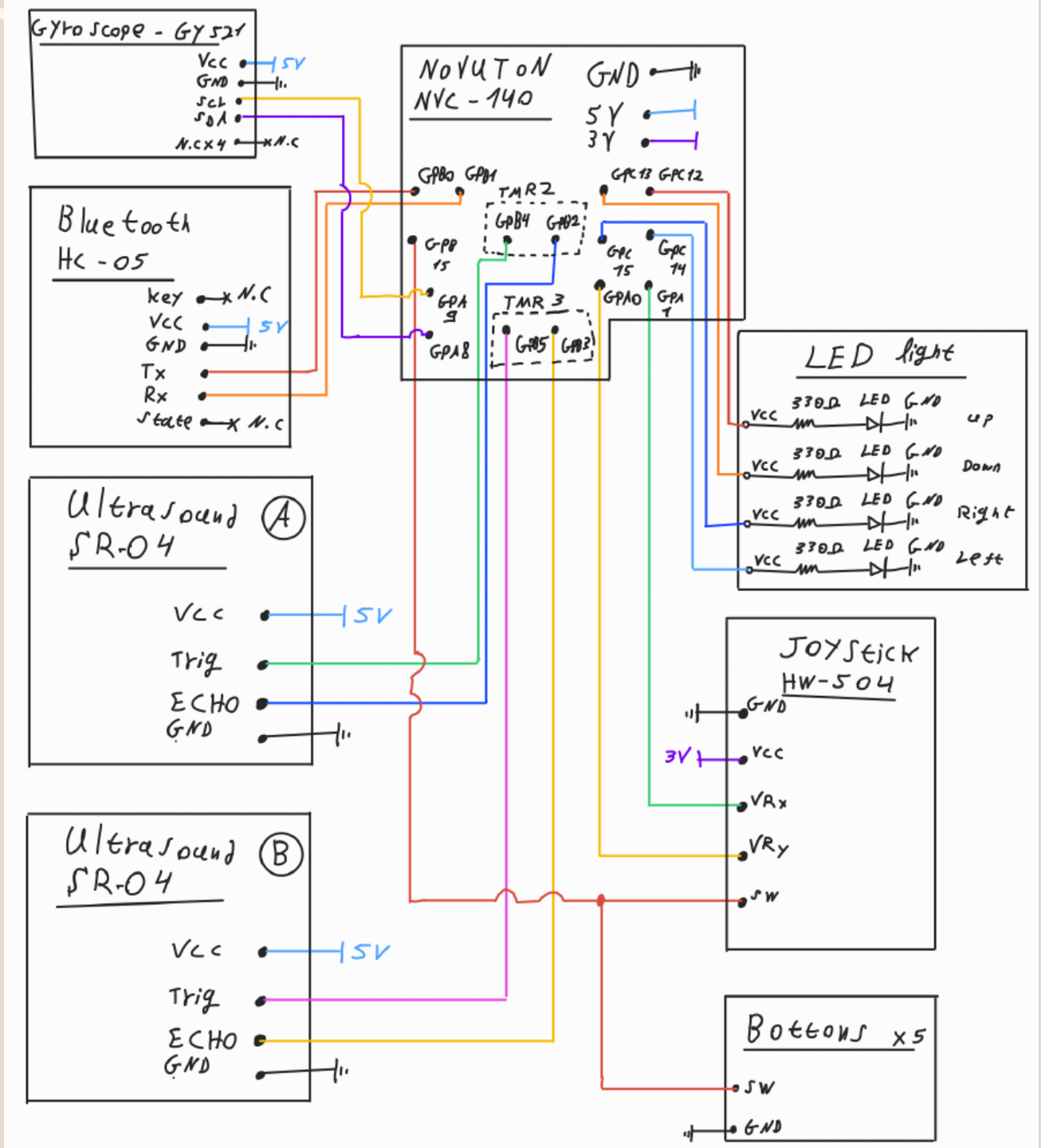
Botton

LED

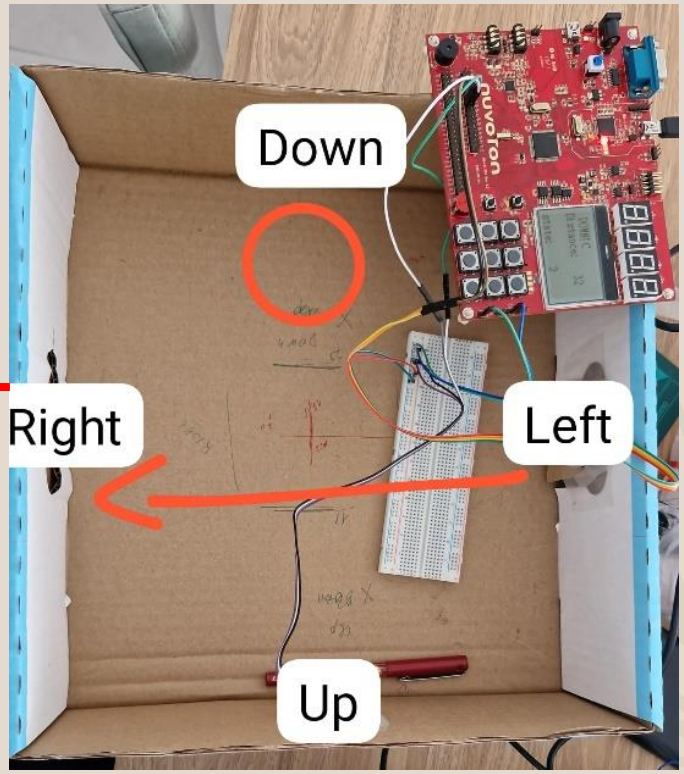
תרשים עקרוני של המערכת

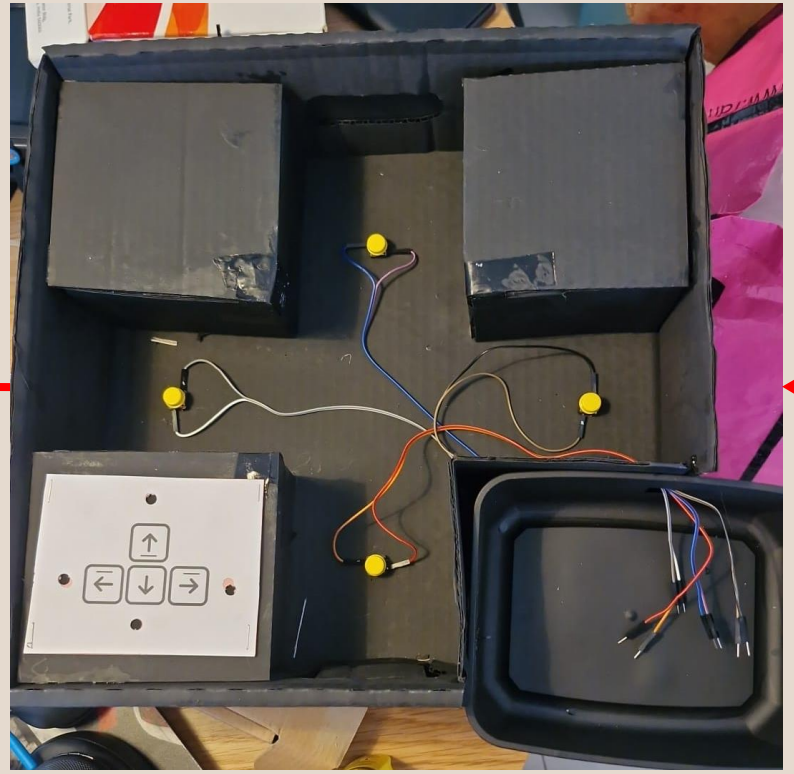


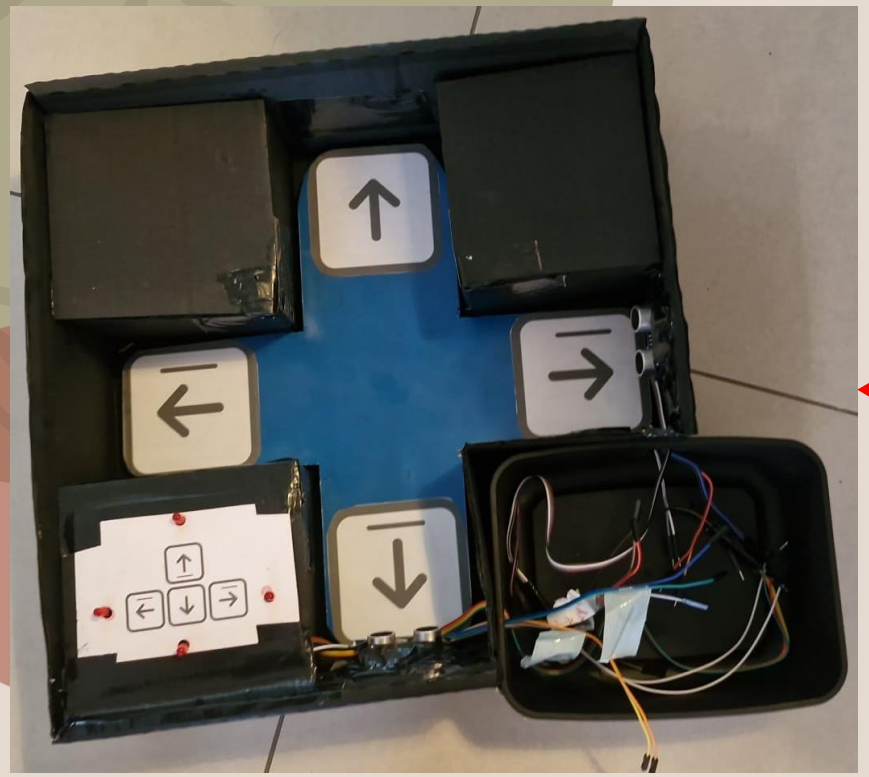
סכמת חיבורים

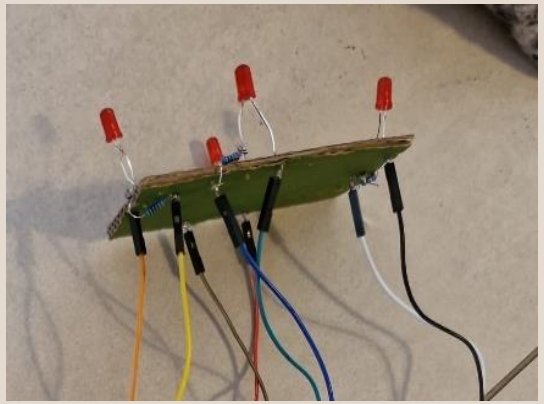


תמונות מתהליך הבניה וההרכבה









הסבר על הקוד

הפונקציה ()main מאתחלת משתנים לאחסון מרחקי חיישן קולי distance_cm_x -ldistance_cm_y

ערכי ג'ויסטיק Vx-lVy, מצב מתג SW ומשתנים נחוצים אחרים.

גם מגדיר ציוד היקפי שונה הכולל נוריות LED, טיימרים ללכידת נתונים קוליים, לוח מקשים ותקשורת UART.

```
int32_t main()
{
        uint32_t distance_cm_x, distance_cm_y;
        uint16_t Vx, Vy;
        uint8_t SW;
        char TEXT[16];
        char *direction;
        int8_t key_button, state = 1;
        int8_t spacekey;
        int8_t LEDUP, LEDDOWN, LEDLEFT, LEDRIGHT;
        STR_UART_T sParam;
        OpenKeyPad();
        Init_LED();
        Init_TMR2(); // initialize Timer2 Capture
        Init_TMR3(); // initialize Timer3 Capture
        Init_GPIO_SR04();
        UNLOCKREG();
        DrvSYS_Open(48000000);
        SYSCLK->PWRCON.XTL12M_EN = 1; // enable external clock (12MHz)
        SYSCLK->CLKSEL0.HCLK_S = 0;     // select external clock (12MHz)
        LOCKREG();
        DrvGPIO_InitFunction(E_FUNC_UART0); // Set UART pins
        /* UART Setting */
        sParam.u32BaudRate = 9600;
        sParam.u8cDataBits = DRVUART_DATABITS_8;
        sParam.u8cStopBits = DRVUART_STOPBITS_1;
        sParam.u8cParity = DRVUART_PARITY_NONE;
        sParam.u8cRxTriggerLevel = DRVUART_FIFO_1BYTES;
        /* Set UART Configuration */
        if (DrvUART_Open(UART_PORT0, &sParam) != E_SUCCESS);
        // DrvUART_EnableInt(UART_PORT0, DRVUART_RDAINT);
```

הסבר על הקוד

קטע זה מאתחל את כניסות הג'ויסטיק והמתג, כמו גם את לוח ה-LCD.

הוא קורא באופן רציף את מצב המתג וכפתורי הג'ויסטיק, ומעדכן את מצב המערכת בהתאם.

אם לוחצים על כפתור מסוים, הוא משנה את מצב המערכת כך שיתאים למצבים או לפעולות שונות:

- לחיצה על 1 = שימוש בג'ויסטיק
- לחיצה על 2 = שימוש באולטרה-סוני
- לחיצה על 3 = In game

```
        //  Joystick init
        DrvADC_Open(ADC_SINGLE_END, ADC_SINGLE_CYCLE_OP, 0x03,
INTERNAL_HCLK, 1); // ADC1 & ADC0
        DrvGPIO_Open(E_GPB, 15, E_IO_INPUT);
                                                        // SW
        Initial_panel(); // initialize LCD panel
        clr_all_panel(); // clear LCD panel

        while (1)
        {
                SW = DrvGPIO_GetBit(E_GPB, 15);

                key_button = Scankey();
                if (key_button == 1)
                {
                        state = 1;
                }
                if (key_button == 2)
                {
                        state = 2;
                }
                if (key_button == 3)
                {
                        state = 0;
                }
                if (state == 1)
```

הסבר על הקוד

בלוק קוד זה מטפל בקלט הג'ויסטיק כאשר המצב מוגדר ל-1.

הוא יוזם המרת ADC,

ממתין עד להשלמתו, ואז מאחזר את ערכי ה- Xוה- Y של הג'ויסטיק מאוגרי תוצאות ה-ADC.

ערכים אלה משמשים לקביעת כיוון הג'ויסטיק, אשר מוצג לאחר מכן על מסך LCD יחד עם ערכי Xו-Y.

```
if (state == 1)
{
    DrvADC_StartConvert();
    while (DrvADC_IsConversionDone() == FALSE)
        ; // wait till conversion is done
    Vx = ADC->ADDR[0].RSLT & 0xFFF;
    Vy = ADC->ADDR[1].RSLT & 0xFFF;
    direction = getDirectionJoystick(Vx, Vy);

    sprintf(TEXT, " %s", direction);
    print_lcd(0, TEXT);
    sprintf(TEXT, "Vx:%4d Vy:%4d", Vx, Vy);
    print_lcd(1, TEXT);
}
```

הסבר על הקוד

כאשר המצב שווה ל-2, החיישנים האולטראסוניים מופעלים, והשהייה של 40 מילישניות מוכנסת כדי לאפשר להד להפעיל פסיקה.

אם מזוהה הד (המציין אובייקט בסמיכות), המרחק מחושב על סמך רוחב ההד, והכיוון נקבע באמצעות הפונקציה getDirectionUltrasonic().

מידע הכיוון והמרחק מוצג על מסך LCD, ואם הכיוון השתנה מאז המדידה האחרונה, הוא נשלח באמצעות UART.

בנוסף, אם מצב כל מתג (גו'יסטיק או 5 לחצנים כללי)הוא 0, הודעה המציינת "רווח" נשלחת דרך UART.

```
if (state == 2)
{
        SR04_Trigger();                 // Trigger Ultrasound Sensor for 10us
        DrvSYS_Delay(40000); // Wait 40ms for Echo to trigger interrupt
        if (SR04A_Echo_Flag == TRUE || SR04B_Echo_Flag == TRUE)
        {
                SR04A_Echo_Flag = FALSE;
                SR04B_Echo_Flag = FALSE;
                distance_cm_x = SR04A_Echo_Width * (340 / 2) / 1000 / 10;
                distance_cm_y = SR04B_Echo_Width * (340 / 2) / 1000 / 10;
                direction = getDirectionUltrasonic(distance_cm_x, distance_cm_y);
        }
        sprintf(TEXT, " %s", direction); // print ADC0 value into text
        print_lcd(0, TEXT);
        sprintf(TEXT, "x: %4d y: %4d", distance_cm_x, distance_cm_y);
        print_lcd(1, TEXT);
}
sprintf(TEXT, "SW: %4d", SW);
print_lcd(2, TEXT);
sprintf(TEXT, "state: %4d", state); // print switch input
print_lcd(3, TEXT);
sprintf(TEXT, " %s", direction);
Turn_Leds(direction);
if (strcmp(direction, prevDirection) != 0)
{
        DrvUART_Write(UART_PORT0, TEXT, strlen(TEXT));
        strcpy(prevDirection, direction);
}
if (SW == 0)
{
        DrvUART_Write(UART_PORT0, " SPACE", 6); // adjustable delay for vision
}
```

הסבר על הקוד

קוד זה קורא נתוני מד תאוצה וג'ירוסקופ מחישן, מחשב זוויות הטיה לאורך צירי X, Y-וZ-, מדפיס זוויות אלו לצג LCD וקובע את הכיוון בהתבסס על כיוון החיישן.

```
if (state == 3)
        {
                tmpL = I2C_Read(MPU6050_ACCEL_XOUT_L); // read Accelerometer X_Low  value
                tmpH = I2C_Read(MPU6050_ACCEL_XOUT_H); // read Accelerometer X_High value
                tmp = (tmpH << 8) + tmpL;
                accX = (float)tmp / 32768 * 2;
                tmpL = I2C_Read(MPU6050_ACCEL_YOUT_L); // read Accelerometer Y_Low  value
                tmpH = I2C_Read(MPU6050_ACCEL_YOUT_H); // read Accelerometer Y_High value
                tmp = (tmpH << 8) + tmpL;
                accY = (float)tmp / 32768 * 2;
                tmpL = I2C_Read(MPU6050_ACCEL_ZOUT_L); // read Accelerometer Z_Low  value
                tmpH = I2C_Read(MPU6050_ACCEL_ZOUT_H); // read Accelerometer Z_High value
                tmp = (tmpH << 8) + tmpL;
                accZ = (float)tmp / 32768 * 2;
                tmpL = I2C_Read(MPU6050_GYRO_XOUT_L); // read Gyroscope X_Low  value
                tmpH = I2C_Read(MPU6050_GYRO_XOUT_H); // read Gyroscope X_High value
                tmp = (tmpH << 8) + tmpL;
                tmpL = I2C_Read(MPU6050_GYRO_YOUT_L); // read Gyroscope Y_Low  value
                tmpH = I2C_Read(MPU6050_GYRO_YOUT_H); // read Gyroscope Y_High value
                tmp = (tmpH << 8) + tmpL;
                tmpL = I2C_Read(MPU6050_GYRO_ZOUT_L); // read Gyroscope Z_Low  value
                tmpH = I2C_Read(MPU6050_GYRO_ZOUT_H); // read Gyroscope Z_High value
                tmp = (tmpH << 8) + tmpL;
                // calculate tilt angle (*57.295 = degree of angle)
                Axr = 57.295 * acos(accX / sqrt(pow(accX, 2) + pow(accY, 2) + pow(accZ, 2)));
                Ayr = 57.295 * acos(accY / sqrt(pow(accX, 2) + pow(accY, 2) + pow(accZ, 2)));
                Azr = 57.295 * acos(accZ / sqrt(pow(accX, 2) + pow(accY, 2) + pow(accZ, 2)));
                // print to LCD
                sprintf(TEXT, "Axr: %f", Axr);
                print_lcd(1, TEXT);
                sprintf(TEXT, "Ayr: %f", Ayr);
                print_lcd(2, TEXT);
                sprintf(TEXT, "Azr: %f", Azr);
                print_lcd(3, TEXT);
                direction = getDirectionGyro(Axr, Ayr, Azr);
        }
```

thank you