НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. ІГОРЯ СІКОРСЬКОГО»

Кафедра приладів і систем орієнтації та навігації

КУРСОВА РОБОТА

з дисципліни "МІКРОПРОЦЕСОРИ ТА МІКРОПРОЦЕСОРНА ТЕХНІКА"

на тему: Котушка для спінінгу з автоматичною підсічкою та витягування

Студента III курсу групи ПГ-01 напряму підготовки 151 - приладобудування спеціальності 151 – автоматизація та комп’ютерно-інтегровані технології приладів і систем орієнтації та навігації

Харковського О.О.

Керівник Вонсевич К.П.

Кількість балів: _____Оцінка: БСТБ

Київ - 2023 рік

Завдання

Завданням даного курсового проєкту є створення спінінгу з мотором - котушкою яка за допомогою датчиків буде реагувати на смикання ліски, коли сила смикання досягне якогось заданого порогу одразу вмикається мотор і починає витягувати рибу, при цьому на дисплей виводиться інформація з датчиків: яка довжина ліски залишилась і інші. Також система оснащена індикаторами які будуть показувати стан системи.

Вступ

Курс «Мікроконтролери та мікропроцесорна техніка » є дуже важливим для нашої майбутньої спеціальності та працевлаштування. Цей проект допоможе сформувати всі вивчені матеріали по курсу та дасть краще розуміння у їх застосуванні. У ході роботи виконуються такі важливі процеси як побудова схема самого мікропроцесора, його програмування та подальше вдосконалення. Дуже важливим аспектом є те, що тему для курсового проекту кожна група обирає самостійно і це дає змогу втілити ідеї кожного студента та створити дійсно індивідуальний проект.

Дане завдання представляє розробку автоматичної котушки для спінінгу з обв’язкою з датчиків та мотора на базі мікроконтролера. Призначений для користування досвідчених рибалок.

Зміст

1. Структурна схема мікропроцесорної системи

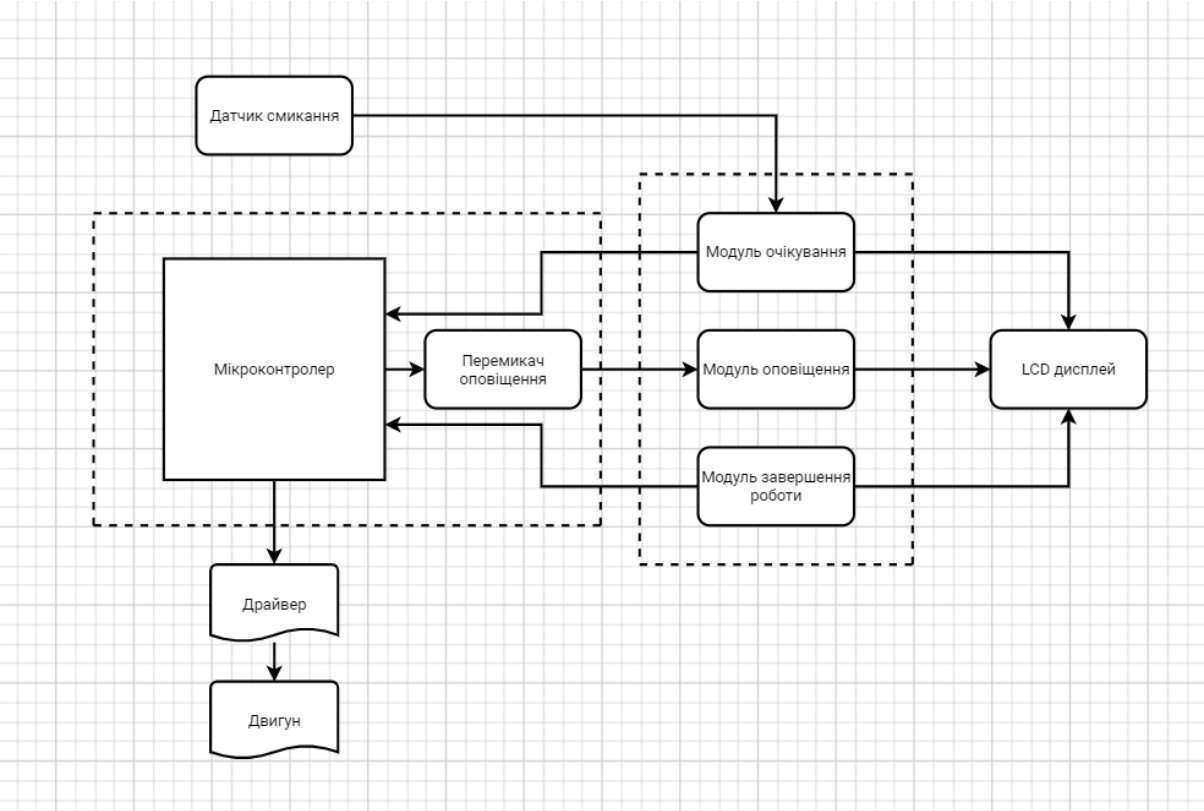


Рис.1.1 Структурна схема

Датчик смикання працює незалежно від мікроконтролера. Сигнал на драйвер подає виключно мікроконтролер. Сигнал з дачитика смикання подається в мікроконтролер, який вмикає модуль оповіщення та керує двигуном. Модуль завершення роботи працює незалежно від мікроконтролера, якщо сигнал з нього приходить до МК, то він подає сигнал на драйвер та завершує роботу.

2. Електрична схема

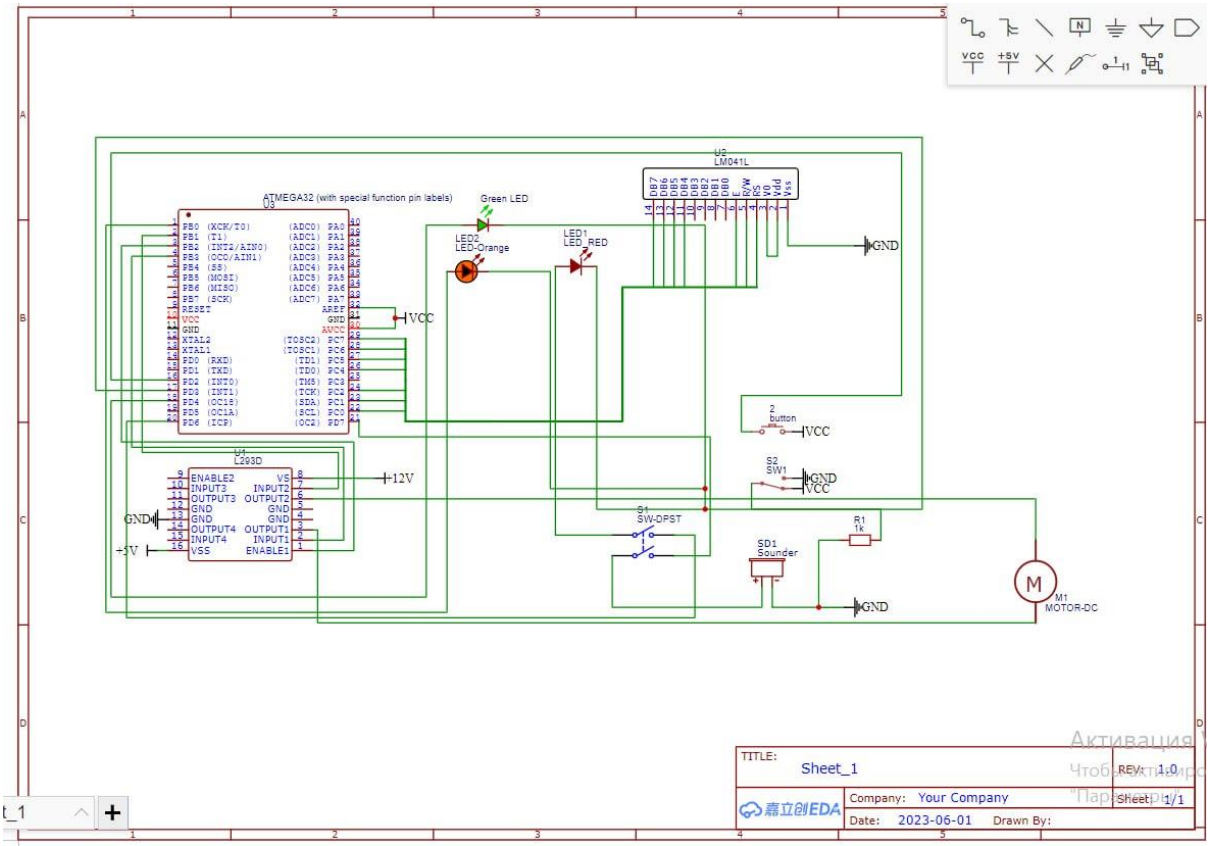


Рис.2.1 Електрична схема

3. Специфікація до електричної схеми

ID	Name	Designator	Footprint	Quantity
1	Green LED	1	LED-GREEN-D4	1
2	button	2	BUTTON	1
3	LED_RED	LED1	LED-RED	1
4	LED-Orange	LED2	LED-TH_BD4.2-P2.54-FD	1
5	MOTOR-DC	M1	2PIN	1
6	ATmega32	MM1	ATMEGA32	1
7	1k	R1	R0603	1
8	SW-DPST	S1	DPST-4-NO	1
9	SW1	S2	SNAP-SWITCHES-1122	1
10	Sounder	SD1	SOUNDER12MM	1
11	L293D	U1	DIP-16_L20.0-W6.4-P2.54-LS7.6-BL	1
12	LM041L	U2	LCD	1

4. Опис схемних рішень, які були використані в системі

Мотор-котушка: Для витягування риби була використана мотор-котушка. Цей компонент складається з електромотора і котушки, на яку намотано рибальську ліску. Мотор керується мікроконтролером і активується, коли сила смикання досягає заданого порогу. Рух мотор-котушки забезпечує витягування риби.

Датчики: Для вимірювання сили смикання та інших параметрів були використані датчики. Конкретний тип датчиків може бути різним, залежно від вимог проєкту. Наприклад, можуть бути використані датчики напруги, датчики тиску або датчики деформації. В нашому випадку, для спрощення конструкції, це підпружинені контакти між якими вставляється ліска. Вони підключаються до мікроконтролера і передають дані про силу смикання та інші параметри для подальшої обробки.

Мікроконтролер: головним керуючим елементом системи є мікроконтролер. Він відповідає за обробку даних з датчиків, прийняття рішень щодо активації мотор-котушки і управління іншими функціями системи. Мікроконтролер програмується за допомогою середовища CodeVision AVR і виконує відповідні алгоритми, що забезпечують правильну роботу системи.

Дисплей: Для виведення інформації з датчиків та стану системи використовується дисплей. Цей компонент може бути різних типів, таких як LCD-дисплей або LED-індикатори. На дисплеї відображаються дані про довжину ліски, стан системи та інші параметри, які допомагають оператору контролювати процес риболовлі.

Вся система побудована на основі взаємодії цих компонентів. Датчики передають дані про силу смикання та інші параметри мікроконтролеру, який аналізує ці дані і визначає потрібні дії. Згідно прийнятих рішень, мікроконтролер активує мотор-котушку для витягування риби і виводить інформацію на дисплей. Таким чином, система забезпечує автоматичне витягування риби і надає оператору зручний інтерфейс для контролю та моніторингу процесу риболовлі.

5. Блок-схема алгоритму роботи системи

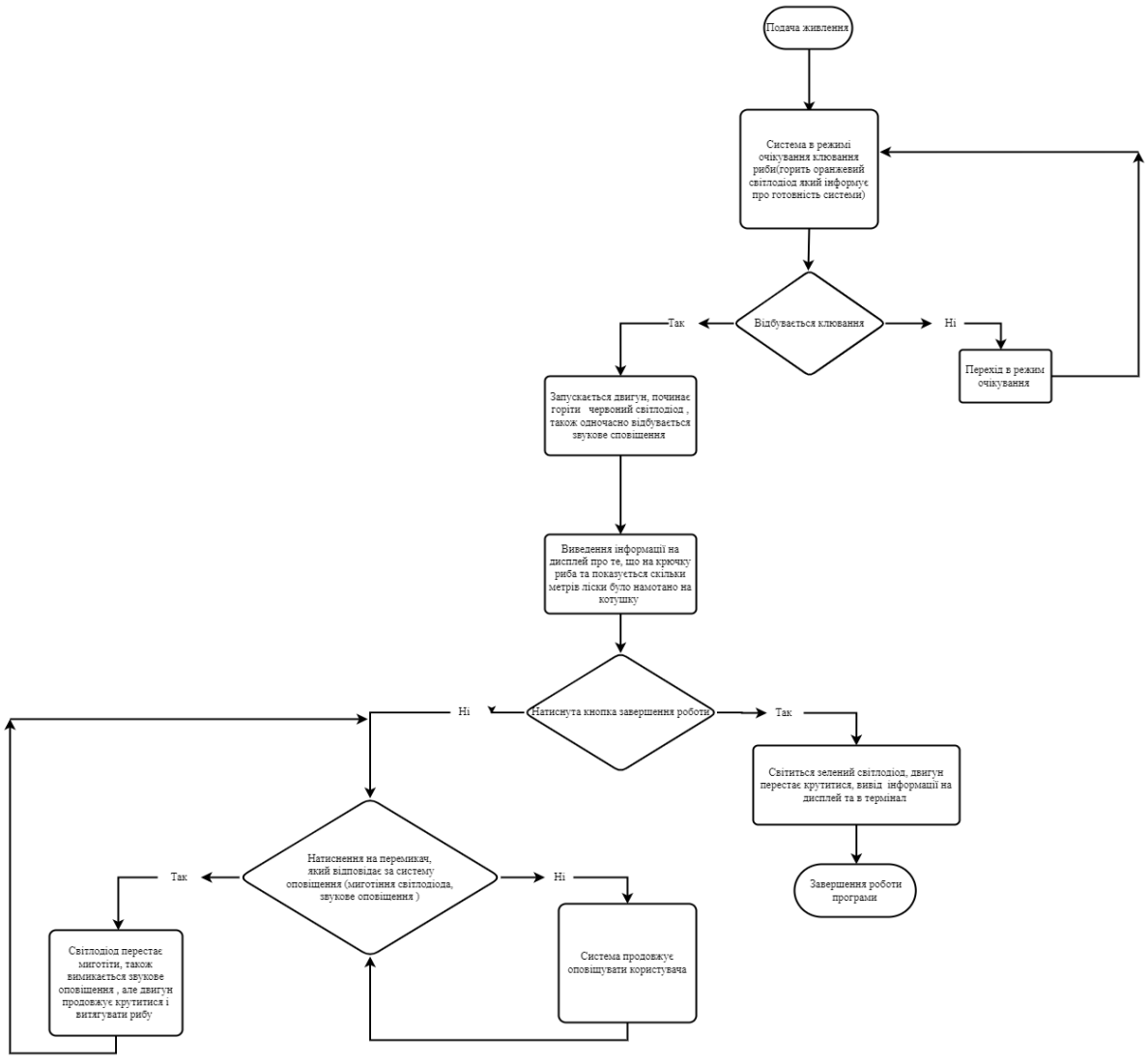


Рис.5.1 Блок-схема

6. Програмний код МК

```
#include <mega32.h>

#include <alcd.h>

#include <sdcard.h>

#include <stdio.h>

#include <delay.h>


int t,L, Lh,L1,T,Th,Tl;

float pi = 3.14158;

float d = 0.007;// діаметр вала двигуна 7 мм

int w = 25; // оберти двигуна в секунду


void start_motor(){

PORTB.1 = 1;//керування мотором

PORTB.2 = 1;

PORTD.6 = 1;//лампа яка показує що система запущена

PORTD.4 = 0;//лампа яка показує що система перейшла в ручний режим/вимкнулась
```

PORTB.0 = 0; //лампа очікування смикання

OCR2 = 0x9F;

TCCR2 = 0x11;

TIMSK=(1<<OCIE1A);//увімкнути переривання

}

void stop_motor(){

PORTB.1 = 0;//керування мотором

PORTB.2 = 0;

PORTD.6 = 0;//лампа яка показує що система запущена

PORTD.4 = 1;//лампа яка показує що система перейшла в ручний режим/вимкнулась

PORTB.0 = 0;//лампа очікування смикання

OCR2 = 0x00;

TCCR2 = 0x00;

TIMSK=(0<<OCIE1A);//вимкнути переривання

}

void cord_left(){ //функція підрахунку довжини замотаної на катушку ліски

```
 T = t/55;

 Th = T/10;

 Tl = T - Th*10;


L = (w*d*pi*T) ;

Lh = L/10;

L1 = L - Lh* 10;

lcd_gotoxy(1,2) ;

lcd_putsf("Cord is pulled:")   ;

lcd_gotoxy(3,3);

lcd_putchar(Lh+0x30);

lcd_gotoxy(4,3);

lcd_putchar(L1+0x30);

lcd_gotoxy(6,3) ;

lcd_putsf("meters") ;

lcd_gotoxy(0,1);

lcd_putsf("                ");

lcd_gotoxy(1,0);

lcd_putsf("Fish on a hook!");
```

```
}


void end_pulling(){  //функція завершення роботи системи
lcd_gotoxy(0,0);
lcd_putsf("   Motor stop   ");
lcd_gotoxy(1,2);
lcd_putsf("Check the hook ");
lcd_gotoxy(0,3);
lcd_putsf("            ") ;
lcd_gotoxy(0,1);
lcd_putsf("         ") ;
}
interrupt [EXT_INT0] void ext_int0_isr(void)
{
stop_motor();}
interrupt [EXT_INT1] void ext_int1_isr(void)
{
```

```
start_motor();}


interrupt [TIM1_COMPA] void timer1_compa_isr(void)

{

disk_timerproc();

t++;

}




void main(void)

{

// Port A initialization

// Function: Bit7=Out Bit6=Out Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In
Bit0=In

DDRA=(0<<DDA7) | (0<<DDA6) | (0<<DDA5) | (0<<DDA4) | (0<<DDA3) |
(0<<DDA2) | (0<<DDA1) | (0<<DDA0);

// State: Bit7=0 Bit6=0 Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T

PORTA=(0<<PORTA7) | (0<<PORTA6) | (0<<PORTA5) | (0<<PORTA4) |
(0<<PORTA3) | (0<<PORTA2) | (0<<PORTA1) | (0<<PORTA0);
```

// Port B initialization

// Function: Bit7=Out Bit6=In Bit5=Out Bit4=Out Bit3=In Bit2=In Bit1=In Bit0=In

DDRB=(1<<DDB7) | (0<<DDB6) | (1<<DDB5) | (1<<DDB4) | (1<<DDB3) | (1<<DDB2) | (1<<DDB1) | (0<<DDB0);

// State: Bit7=0 Bit6=T Bit5=0 Bit4=0 Bit3=T Bit2=T Bit1=T Bit0=T

PORTB=(0<<PORTB7) | (0<<PORTB6) | (0<<PORTB5) | (0<<PORTB4) | (0<<PORTB3) | (0<<PORTB2) | (0<<PORTB1) | (0<<PORTB0);

// Port C initialization

// Function: Bit7=Out Bit6=Out Bit5=Out Bit4=Out Bit3=Out Bit2=Out Bit1=Out Bit0=Out

DDRC=(1<<DDC7) | (1<<DDC6) | (1<<DDC5) | (1<<DDC4) | (1<<DDC3) | (1<<DDC2) | (1<<DDC1) | (1<<DDC0);

// State: Bit7=0 Bit6=0 Bit5=0 Bit4=0 Bit3=0 Bit2=0 Bit1=0 Bit0=0

PORTC=(0<<PORTC7) | (0<<PORTC6) | (0<<PORTC5) | (0<<PORTC4) | (0<<PORTC3) | (0<<PORTC2) | (0<<PORTC1) | (0<<PORTC0);

// Port D initialization

// Function: Bit7=Out Bit6=Out Bit5=In Bit4=In Bit3=In Bit2=In Bit1=Out Bit0=In

DDRD=(1<<DDD7) | (1<<DDD6) | (1<<DDD5) | (0<<DDD4) | (0<<DDD3) | (0<<DDD2) | (1<<DDD1) | (0<<DDD0);

// State: Bit7=0 Bit6=0 Bit5=T Bit4=T Bit3=T Bit2=T Bit1=0 Bit0=T

PORTD=(0<<PORTD7) | (0<<PORTD6) | (0<<PORTD5) | (0<<PORTD4) | (0<<PORTD3) | (0<<PORTD2) | (0<<PORTD1) | (0<<PORTD0);

// Timer/Counter 0 initialization

// Clock source: System Clock

// Clock value: Timer 0 Stopped

// Mode: Normal top=0xFF

// OC0 output: Disconnected

TCCR0=(0<<WGM00) | (0<<COM01) | (0<<COM00) | (0<<WGM01) | (0<<CS02) | (0<<CS01) | (0<<CS00);

TCNT0=0x00;

OCR0=0x00;

// Timer/Counter 1 initialization

// Clock source: System Clock

// Clock value: 4000,000 kHz

// Mode: Normal top=0xFFFF

// OC1A output: Disconnected

// OC1B output: Disconnected

// Noise Canceler: Off

// Input Capture on Falling Edge

// Timer Period: 16,384 ms

// Timer1 Overflow Interrupt: Off

// Input Capture Interrupt: Off

// Compare A Match Interrupt: On

// Compare B Match Interrupt: Off

TCCR1A=(0<<COM1A1) | (0<<COM1A0) | (0<<COM1B1) | (0<<COM1B0) | (0<<WGM11) | (0<<WGM10);

TCCR1B=(0<<ICNC1) | (0<<ICES1) | (0<<WGM13) | (0<<WGM12) | (0<<CS12) | (0<<CS11) | (1<<CS10);

TCNT1H=0x00;

TCNT1L=0x00;

ICR1H=0x00;

ICR1L=0x00;

OCR1AH=0x01;

OCR1AL=0x90;

OCR1BH=0x00;

OCR1BL=0x00;

// Timer/Counter 2 initialization

// Clock source: System Clock

// Clock value: 4000,000 kHz

// Mode: Normal top=0xFF

// OC2 output: Disconnected

// Timer Period: 0,064 ms

ASSR=0<<AS2;

TCCR2=(0<<PWM2) | (0<<COM21) | (0<<COM20) | (0<<CTC2) | (0<<CS22) | (0<<CS21) | (1<<CS20);

TCNT2=0xFF;

OCR2=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization

TIMSK=(0<<OCIE2) | (0<<TOIE2) | (0<<TICIE1) | (1<<OCIE1A) | (0<<OCIE1B) | (0<<TOIE1) | (0<<OCIE0) | (0<<TOIE0);

// External Interrupt(s) initialization

// INT0: On

// INT0 Mode: Any change

// INT1: On

// INT1 Mode: Rising Edge

// INT2: Off

GICR|=(1<<INT1) | (1<<INT0) | (0<<INT2);

MCUCR=(1<<ISC11) | (1<<ISC10) | (0<<ISC01) | (1<<ISC00);

MCUCSR=(0<<ISC2);

GIFR=(1<<INTF1) | (1<<INTF0) | (0<<INTF2);

// USART initialization

// Communication Parameters: 8 Data, 1 Stop, No Parity

// USART Receiver: On

// USART Transmitter: On

// USART Mode: Asynchronous

// USART Baud Rate: 4800

UCSRA=(0<<RXC) | (0<<TXC) | (0<<UDRE) | (0<<FE) | (0<<DOR) | (0<<UPE) | (0<<U2X) | (0<<MPCM);

UCSRB=(0<<RXCIE) | (0<<TXCIE) | (0<<UDRIE) | (1<<RXEN) | (1<<TXEN) | (0<<UCSZ2) | (0<<RXB8) | (0<<TXB8);

UCSRC=(1<<URSEL) | (0<<UMSEL) | (0<<UPM1) | (0<<UPM0) | (0<<USBS) | (1<<UCSZ1) | (1<<UCSZ0) | (0<<UCPOL);

```
UBRRH=0x00;

UBRRL=0x33;


// Analog Comparator initialization

// Analog Comparator: Off

// The Analog Comparator's positive input is

// connected to the AIN0 pin

// The Analog Comparator's negative input is

// connected to the AIN1 pin

ACSR=(1<<ACD) | (0<<ACBG) | (0<<ACO) | (0<<ACI) | (0<<ACIE) |
(0<<ACIC) | (0<<ACIS1) | (0<<ACIS0);

SFIOR=(0<<ACME);


// ADC initialization

// ADC disabled

ADCSRA=(0<<ADEN) | (0<<ADSC) | (0<<ADATE) | (0<<ADIF) |
(0<<ADIE) | (0<<ADPS2) | (0<<ADPS1) | (0<<ADPS0);


// SPI initialization

// SPI disabled
```

SPCR=(0<<SPIE) | (0<<SPE) | (0<<DORD) | (0<<MSTR) | (0<<CPOL) | (0<<CPHA) | (0<<SPR1) | (0<<SPR0);

// TWI initialization

// TWI disabled

TWCR=(0<<TWEA) | (0<<TWSTA) | (0<<TWSTO) | (0<<TWEN) | (0<<TWIE);

// Alphanumeric LCD initialization

// Connections are specified in the

// Project|Configure|C Compiler|Libraries|Alphanumeric LCD menu:

// RS: PORTC Bit 0

// RD: PORTC Bit 1

// EN: PORTC Bit 2

// D4: PORTC Bit 4

// D5: PORTC Bit 5

// D6: PORTC Bit 6

// D7: PORTC Bit 7

// Characters/line: 16

```
//// Watchdog Timer initialization

//// Watchdog Timer Prescaler: OSC/2048k

//WDTCR=(0<<WDTOE) | (1<<WDE) | (1<<WDP2) | (1<<WDP1) |
(1<<WDP0);


// Globally enable interrupts

#asm("sei")


TIMSK=(0<<OCIE1A)   ;

lcd_init(16);

lcd_gotoxy(5,1);

lcd_putsf("Hello!");

delay_ms(2000);

lcd_clear();

PORTB.0 = 1;



while (1)

    {
```

```
if (PORTB.0 == 1 ){
 lcd_gotoxy(5,1);
 lcd_putsf("Waiting");
 lcd_gotoxy(3,2);
 lcd_putsf("for a bite"); }

 else if (PORTD.6 == 1){
    cord_left();}

 else if(PORTD.4 == 1){
    end_pulling();
    putsf("Fish was pulled out in \r");
     putchar(Th+0x30);
     putchar(Tl+0x30);
    putsf(" seconds \r");
    putsf("\r");
    putsf("The distance from which the fish was pulled \r");
    putchar(Lh+0x30);
    putchar(L1+0x30);
```

putsf(" meters \r");

break; }}}

7. Остаточний вигляду розробленої схеми МПС

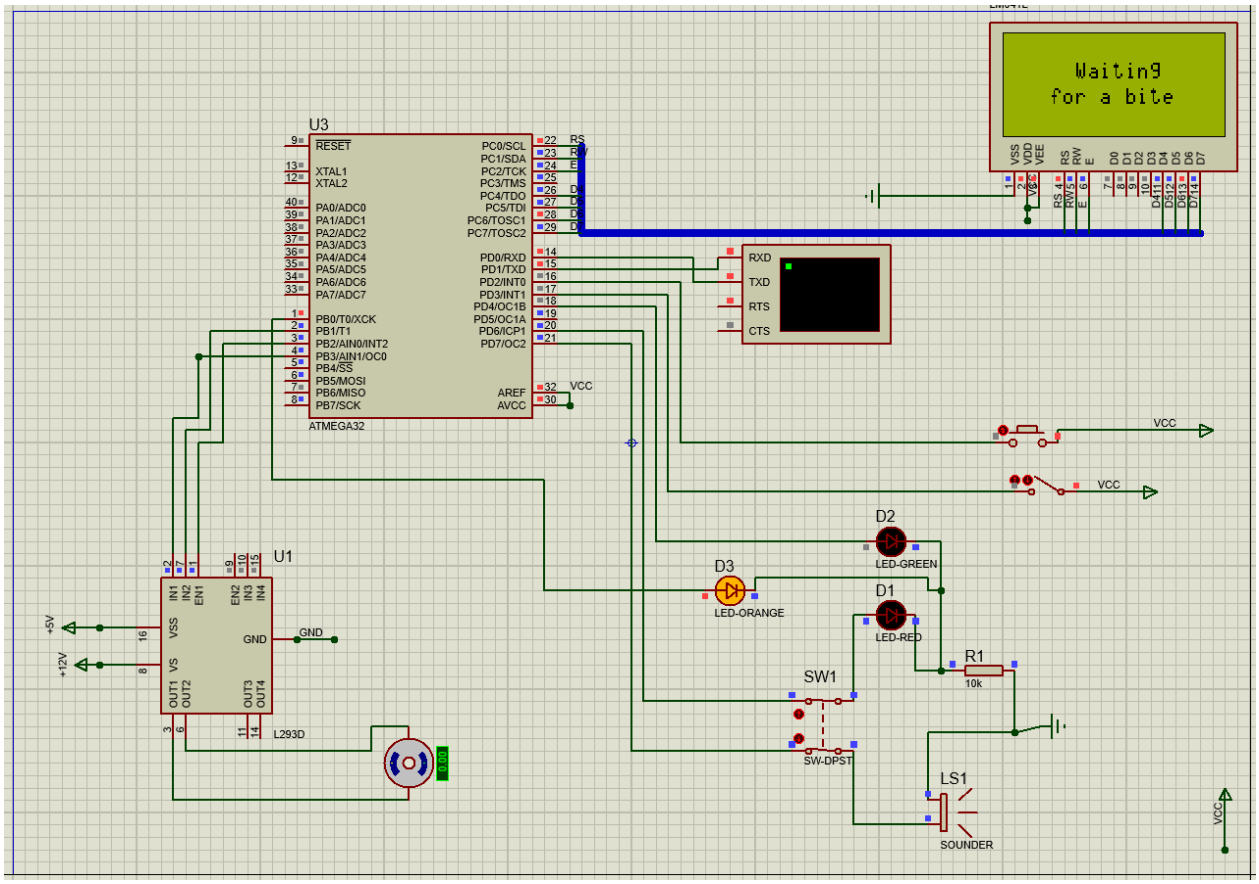


Рис.7.1 Схема МПС

8. Покрокова інструкція для моделювання електричної схеми у PROTEUS

Перед запуском системи потрібно роз’єднати ключ показаний на рис.3.1. Таким чином ми моделюємо реальну схему в якій цей ключ роз’єднаний через ліску, яка знаходиться між контактами ключа.

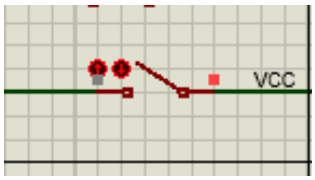


Рис. 8.1 Підготовка до роботи

Після запуску моделювання на екран виводиться привітальне повідомлення (рис.3.2).

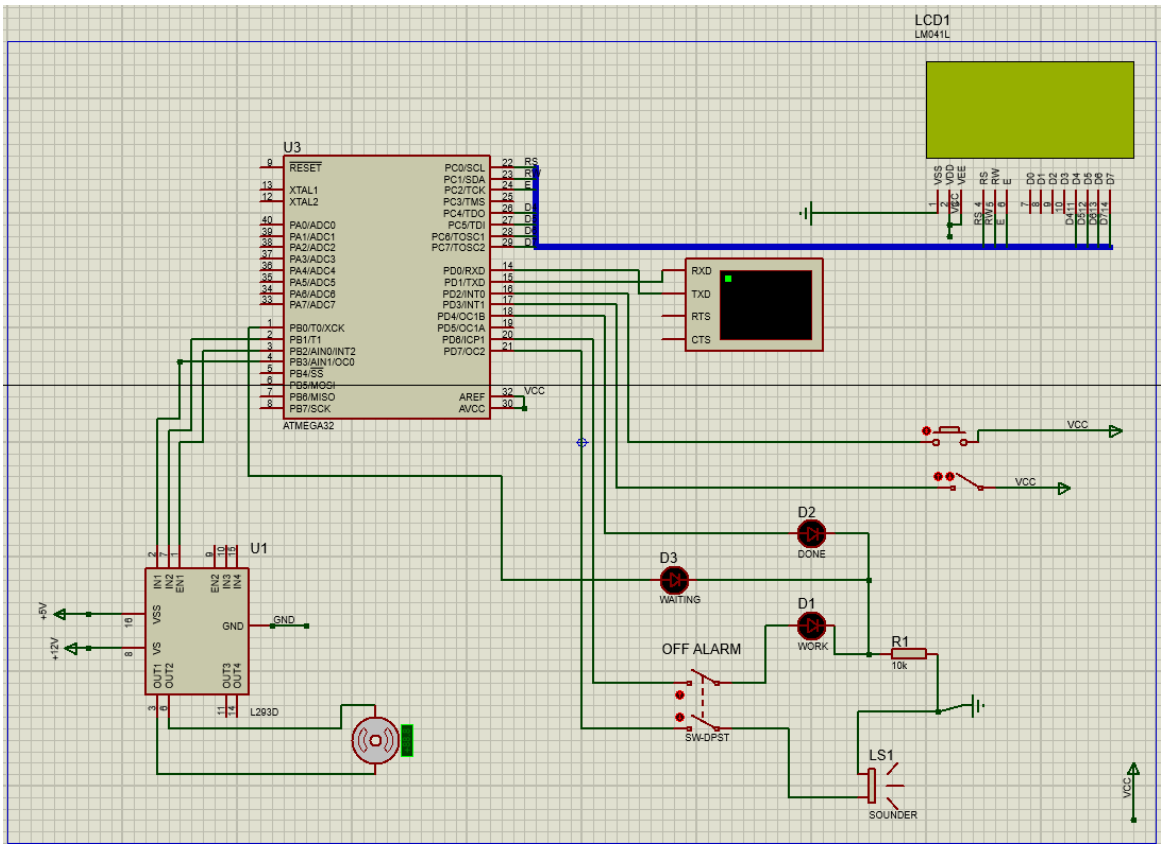


Рис. 8.2 Перший запуск

Через декілька секунд після запуску системи, вона переходить в режим очікування смикання риби. На екран виводиться відповідне повідомлення яке інформує що система працює справно і очікує сигналу. Під час цього процесу від оператора нічого не потребується, окрім як почути сигнали спрацювання системи.

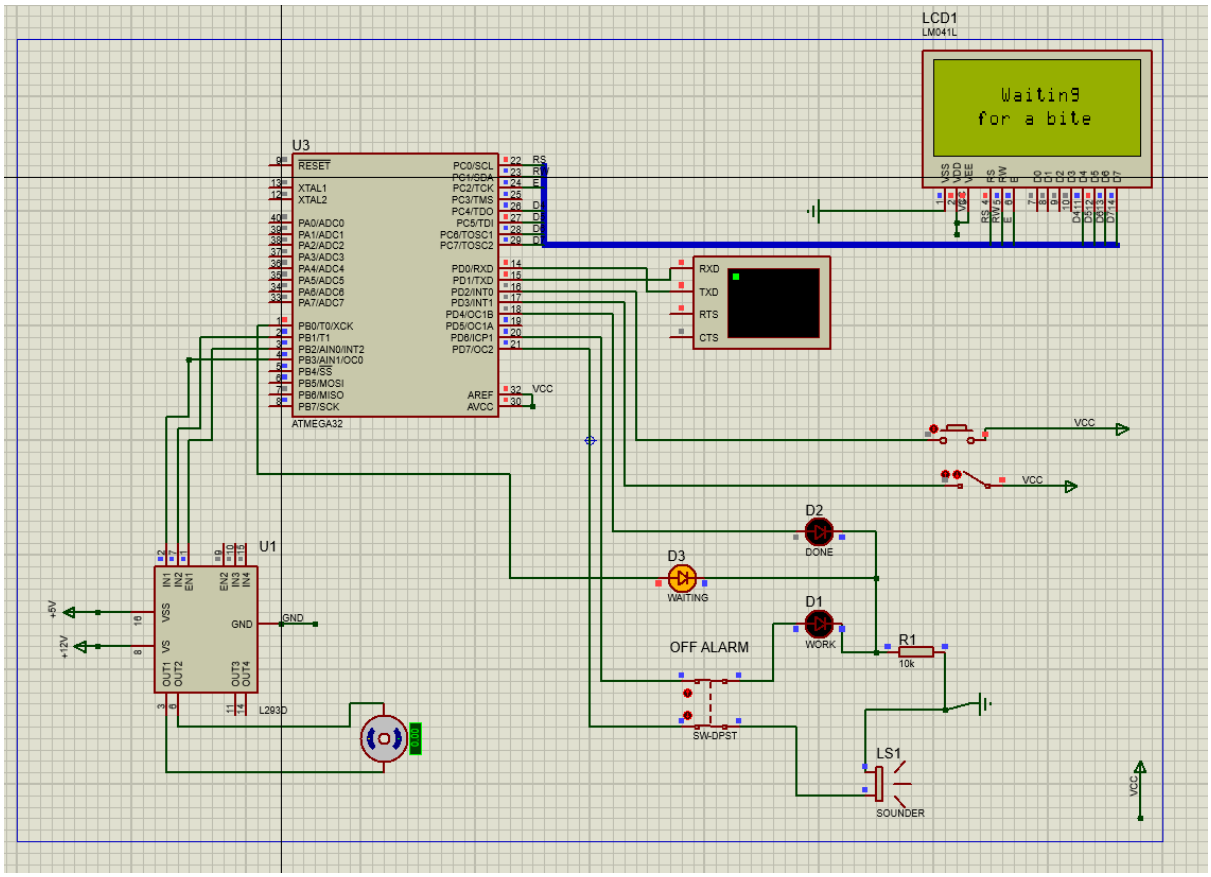


Рис. 8.3 Очікування смикання

Щоб змоделювати смикання риби замкнемо ключ показаний на рис.3.4.

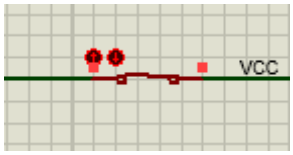


Рис.8.4 Відбулось смикання

Після замикання ключа автоматично спрацьовує система оповіщення кльову та вмикається мотор. До системи оповіщення входить червоний світлодіод та динамік. З моменту як замикається ключ, на екран виводиться повідомлення що риба на гачку та розраховується довжина ліски яку вже намотало на катушку.

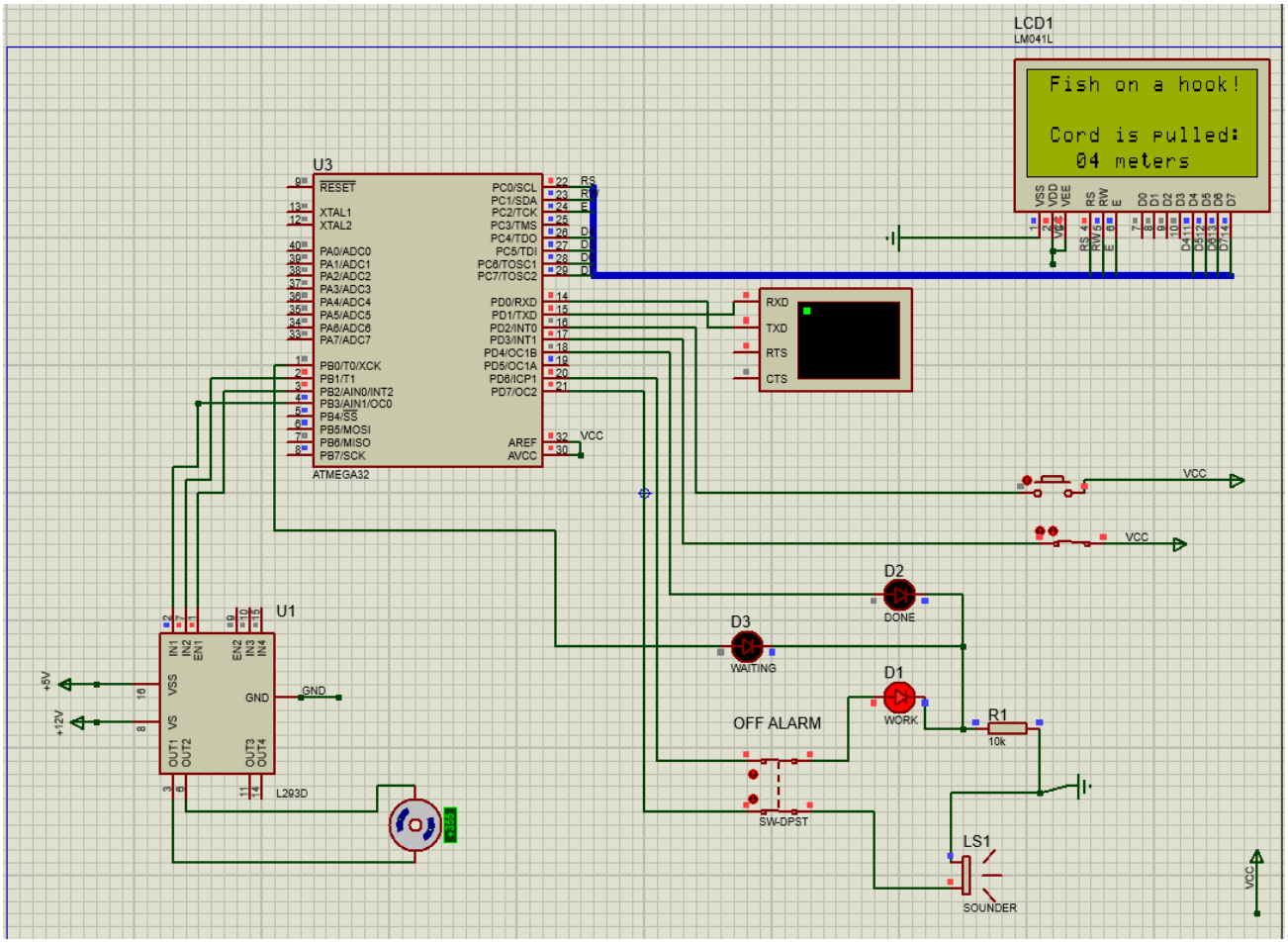


Рис.8.5 Система в роботі

В системі передбачено вимкнення оповіщення без вимкнення мотору. Щоб це зроби потрібно перемкнути ключ який показаний на рис.3.6.

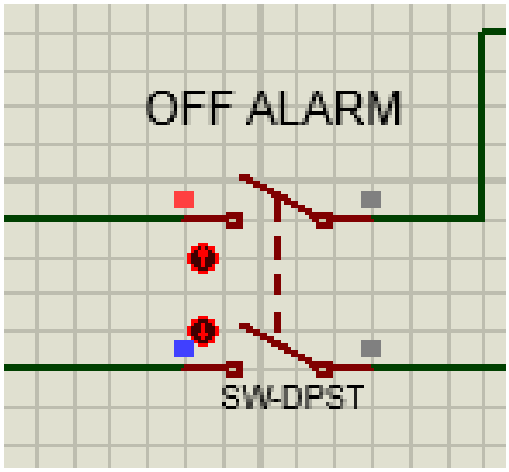


Рис.8.6 Вимкнення системи оповіщення

Завершення роботи системи здійснюється за допомогою кнопки показаної на рис.3.7.

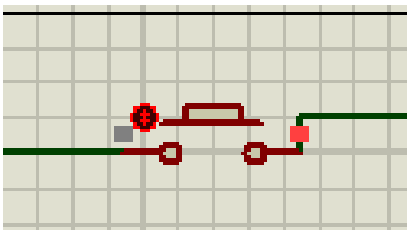


Рис.8.7 Кнопка завершення роботи системи

Після завершення роботи на екрані можна побачити побачити повідомлення, що мотор вимкнуто та потрібно перевірити гачок. Вмикається зелений світлодіод який інформує що системи завершила роботу. Також у віртуальному терміналі можна побачити інформацію про процес витягання риби, а саме час який витягалась риба та довжина намотаної ліски, тобто відстань з якої риба тягнулась.

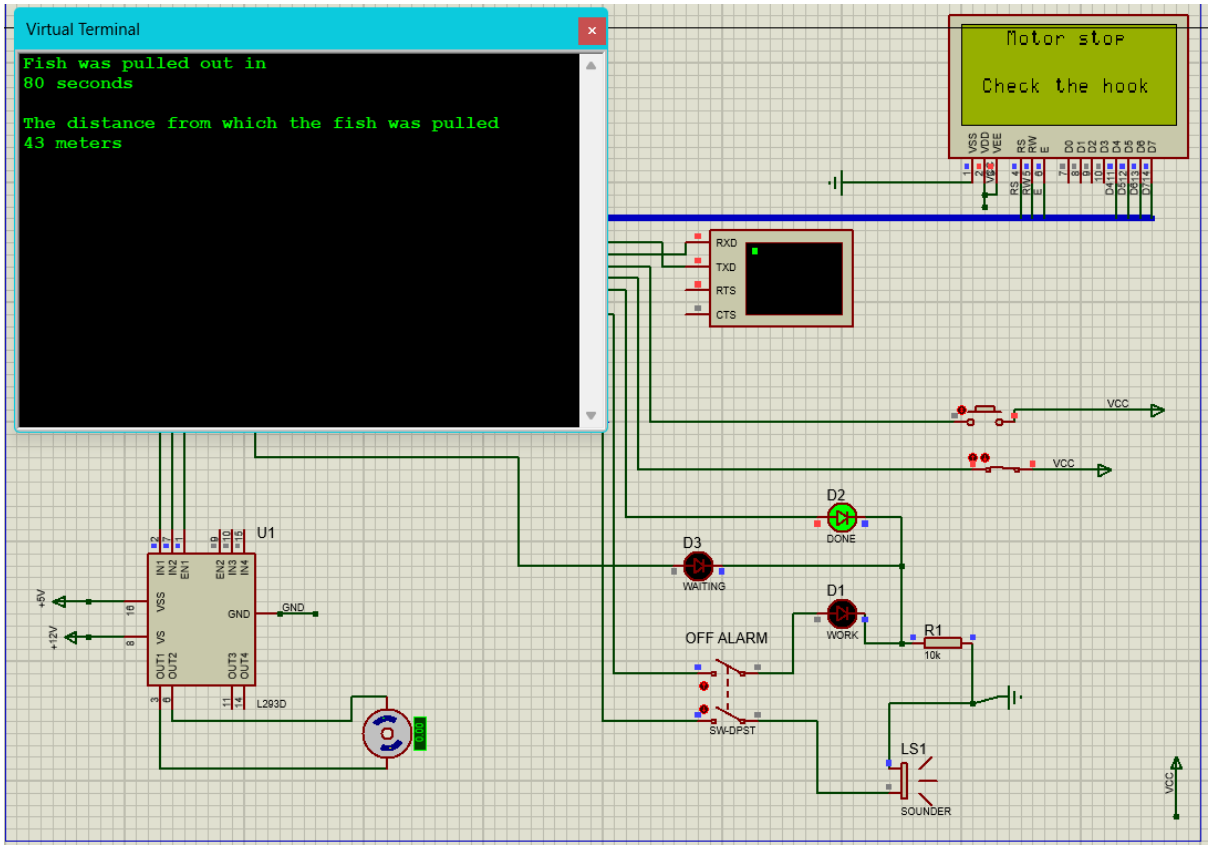
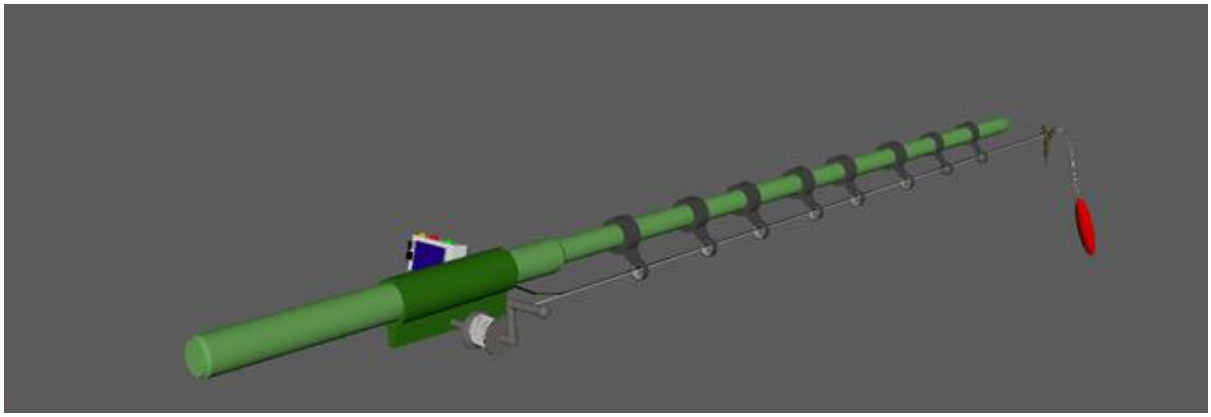


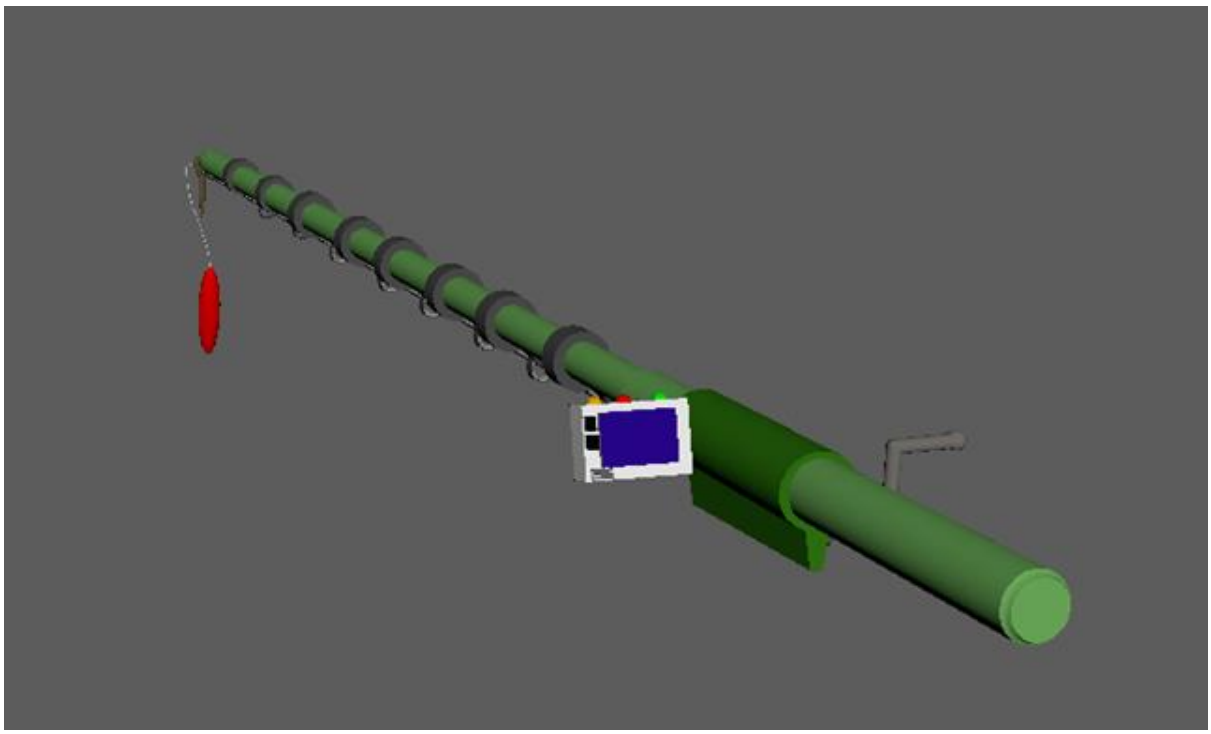
Рис.8.8 Завершення роботи системи

Щоб розпочати роботу заново треба виконати дії які описані на початку інструкції та перезавантажити систему.

9. 3D модель курсового проєкту

3D модель спінінга створена допомогою середовища AutoDesk Maya .До спінінгу приєднано нашу систему, яка містить: LCD дисплей, світлодіоди, датчики, динамік, кнопки (Рис 9.1).





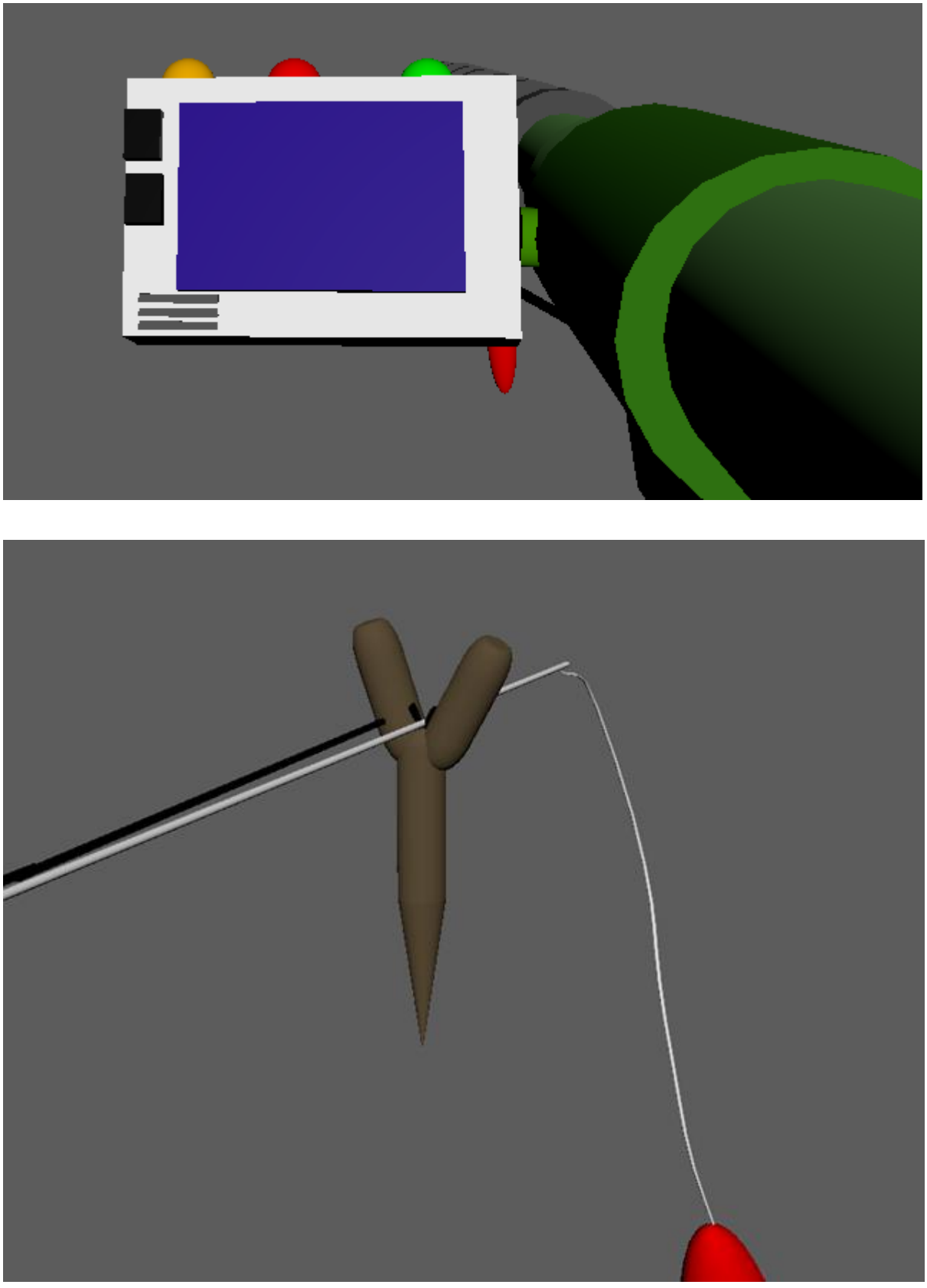


Рис. 9.1-4. 3D модель в середовищі AutoDesk Maya

10. Макет друкованої плати

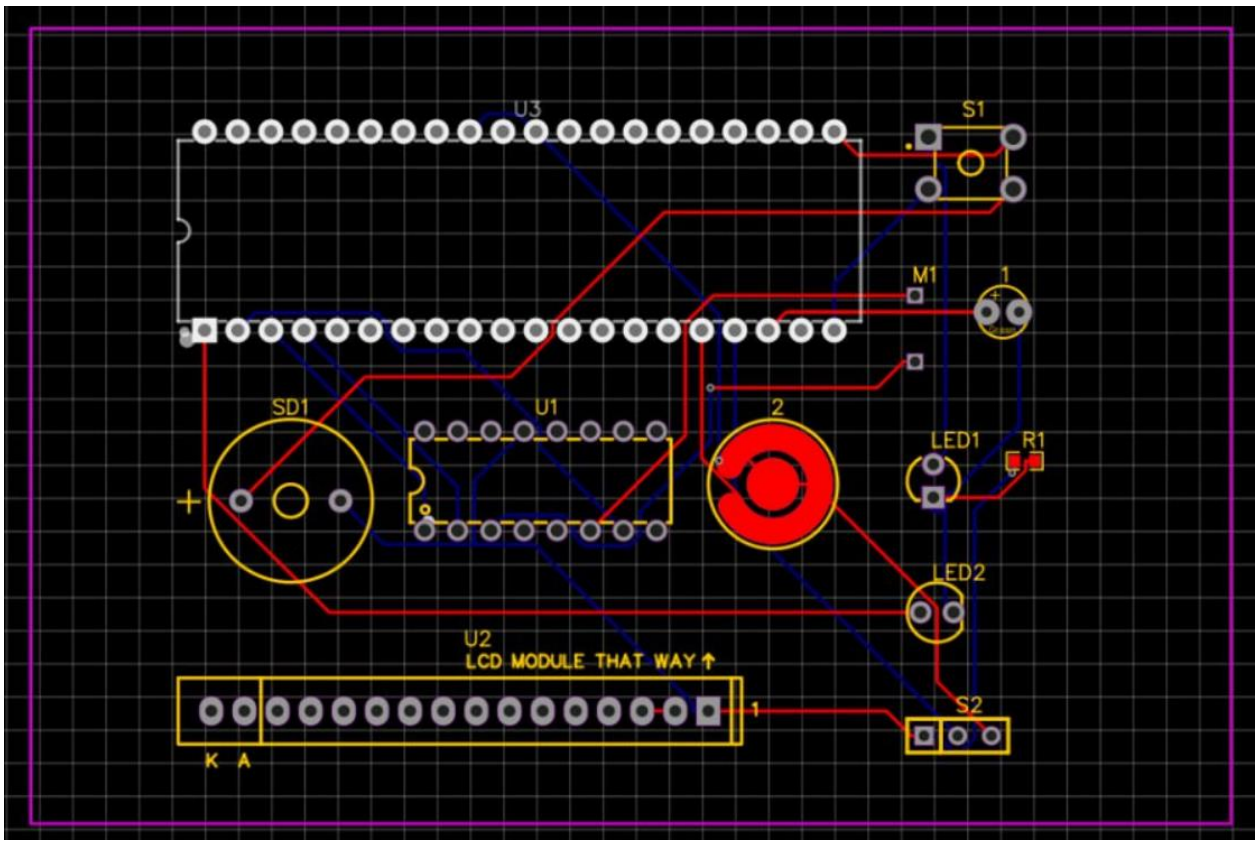


Рис.10.1 Макет друкованої плати

Висновки

В ході виконання курсової роботи було створено спінінг з мотором-котушкою, який реагує на смикання ліски за допомогою датчиків. Завдання полягало у тому, щоб мотор вмикався, коли сила смикання досягала заданого порогу, і розпочинав витягувати рибу. Крім того, інформація з датчиків, така як залишкова довжина ліски та інші дані, виводилась на дисплей.

У середовищі Proteus та CodeVision AVR було реалізовано необхідний функціонал системи. Proteus дозволяє моделювати та симулювати роботу електронних схем, що дозволило перевірити правильність роботи спінінгу без фізичного монтажу. CodeVision AVR надав можливість розробляти програмне забезпечення для мікроконтролерів AVR, що використовувалися у системі.

Завдяки використанню датчиків та мотора-котушки, система здатна реагувати на смикання ліски та автоматично витягувати рибу, що полегшує роботу рибалки. Інформація, виведена на дисплей, надає користувачеві актуальні дані про стан системи, зокрема залишкову довжину ліски. Індикатори, вбудовані в систему, дозволяють візуально спостерігати за станом роботи системи.

Отже, розроблена система спінінгу з мотором-котушкою та датчиками, що реагують на смикання ліски, виявилась функціональною та корисною для рибалки. Вона дозволяє автоматично витягувати рибу при досягненні заданого порогу смикання, а також надає користувачеві інформацію про стан системи. Результати курсової роботи свідчать про успішне виконання поставлених завдань та демонструють ефективність використання середовищ Proteus та CodeVision AVR для розробки подібних систем.

Джерела

1. МІКРОКОНТРОЛЕРИ ТА МІКРОПРОЦЕСОРНА ТЕХНІКА : Вонсевич Костянтин Петрович, канд. техн. наук Безуглий Михайло Олександрович, д-р. техн. наук, професор; Київ КПІ ім. Ігоря Сікорського 2022. - 83 с.
2. Мікропроцесорна техніка: Комп’ютерний практикум [Електронний ресурс] : навч. посіб. для студ. спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології» / К. П. Вонсевич, М. О. Безуглий ; КПІ ім. Ігоря Сікорського. – Електронні текстові данні (1 файл: 3,53 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2021. – 94 с.
3. 3. Програмування мікроконтролерів AVR : [навчальний посібник] / С. М. Цирульник, О. Д. Азаров, Л. В. Крупельницький, Т. І. Трояновська. – Вінниця : ВНТУ, 2018. – 111 с.