Introduction

Our project **‘Underwater Vehicle Control’** is basically a miniature adaptation of submarine. As our land is criss-crossed by innumerable river, we often need to check out underwater situation. This project is a low cost solution to the problem. With some modification it also can be used as underwater rescue system, underwater surveillance system as well as underwater object tracking unit. Regarding to our environment, this project has great prospect.

System Overview:

The project is divided into two parts-i) building a structure for underwater vehicle and ii)controlling the structure. At first we had to assemble the structure whose neutral point is below surface water level. Though submarines use pumps to control vertical alignment, we used thrust created by propeller complexity of design and high cost factor.

Structure

The vehicle is cylindrical with sharp angular dome shaped front side similar to bullet. Front portion is angular so that it can cruise under water with minimum water repulsion. It was built molding stainless steel sheet. The shell was inflated and made watertight. Two pieces of cork sheets was used as fin to balance and keep the centre of gravity at mid area.

Fig: top view of Underwater Vehicle Structure

Control System

The whole control system is divided into two parts

1. Vertical Position Control

Vertical position is controlled by propeller connected to a dc motor. Here we used a gear motor to generate more torque rather than speed. In order to control both upward and downward direction, gear motor was connected 12 v dc source through IC L298. The speed is regulated by PWM signal created by microcontroller.

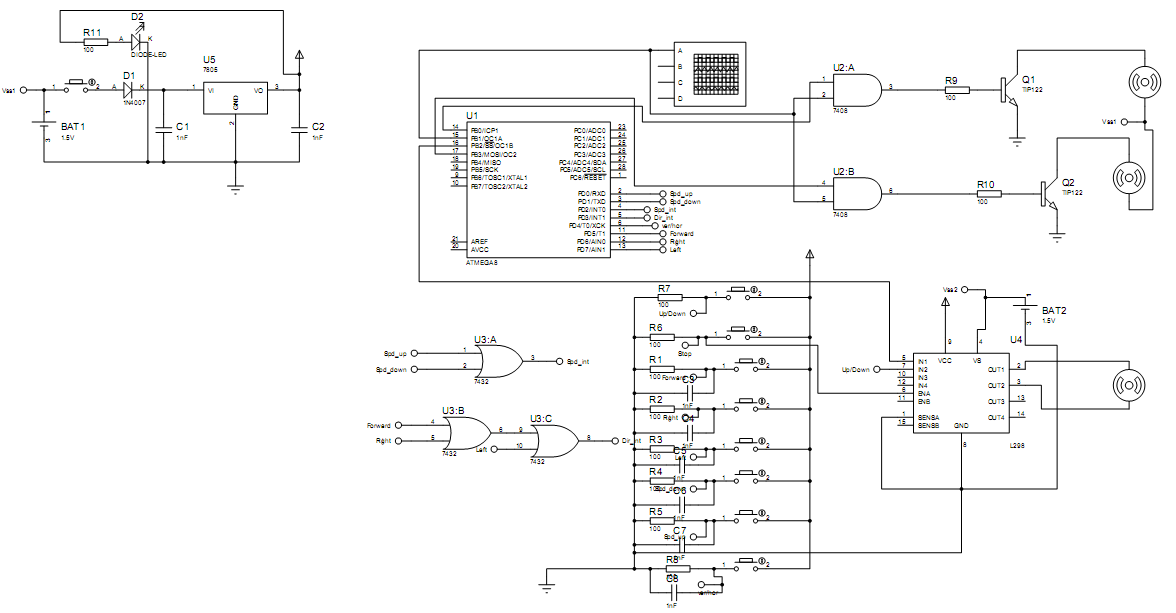
1. Horizontal Position Control

For horizontal position control, we used two brushless DC motors. Speed can be controlled from low to high in 10 steps. For each interrupt the microcontroller gets, duty cycle of the PWM is increased by 10%. When we want to move the vehicle towards left, then left motor is turned off and right motor is rotated at full speed and vice-versa.

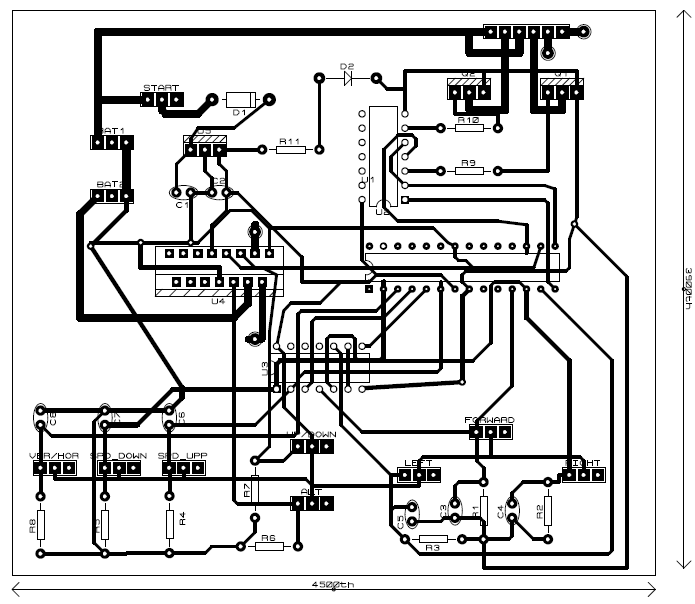
Hardware Requirement

1. DC gear motor
2. Brushless DC motor
3. ATmega8
4. L 298
5. L 7805
6. 7408(AND Gate)
7. 7432(OR Gate)

Circuit Diagram:



PCB Design



Code

#include <avr\io.h>

#include <avr\interrupt.h>

int i,k,l,v;

void Int\_init(void);

ISR (INT1\_vect)

{ int j,m;

for(j=0;j<12;j++) //delay to chcek bouncing

for(m=0;m<100;m++)

;

//Checking which button was pushed

if(PIND & (1<<PD5)) //Start/stop button is pushed

{ if(l) //l=1 start l=0 stop

{

PORTB|=9;

i=125; //enable both motor at 50% duty cycle

}

else

{

PORTB &=0xF6; //both the motor is disabled

}

l^=1; //Toggle l

}

else if(PIND & (1<<PD6)) //Right button is pushed

{

PORTB|=1; //enable motor 1

PORTB &=0b11110111; //disable motor 2

//Same duty cycle

PORTC|=(1<<5);

if(l==0)

PORTC|=(1<<4);

}

else if(PIND & (1<<PD7)) //LEFT button is pushed

{

PORTB|=8; //enable motor 2

PORTB &=0b11111110; //disable motor 1

//Same duty cycle

PORTC|=(1<<5);

if(l==0)

PORTC|=(1<<4);

}

else //No button was pushed

;

}

ISR (INT0\_vect)

{

int j,m;

for(j=0;j<12;j++) //delay to chcek bouncing

for(m=0;m<100;m++)

;

if(PIND & (1<<PD4)) //Horizontal speed

{

if (PIND & (1<<PD0)) //Speed Up

{

if(i<=230)

i+=25;

else

i=250;

}

else if(PIND & (1<<PD1)) //Speed Down

{

if(i>=25)

i-=25;

else

i=0;

}

}

else

{ //Vertical speed

if (PIND & (1<<PD0)) //Speed Up

{

if(v<=230)

v+=25;

else

v=250;

}

else if(PIND & (1<<PD1)) //Speed Down

{

if(v>=25)

v-=25;

else

v=0;

}

}

}

void Int\_init(void)

{

// To set RISING edge of the interrupt input to generate an interrupt request

MCUCR |= (1<<ISC11) | (1<<ISC10); // INT1 to rising edge

MCUCR |= (1<<ISC01) | (1<<ISC00); // INT0 to rising edge

// To enable INT1 & INT0

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

//To clear INT0 port ( PD2 pin ) so that this pin operates as input

DDRD &= ~(1<<PD2);

//To clear INT1 port ( PD3 pin ) so that this pin operates as input

DDRD &= ~(1<<PD3);

}

/\* signal handler for timer overflow interrupt TOV0 \*/

/\* signal handler for timer overflow interrupt TOV0 \*/

ISR(TIMER1\_COMPA\_vect) {

OCR1A=i;

}

ISR(TIMER1\_COMPB\_vect) {

OCR1B=v;

}

void square\_wave\_init(void){

/\* use PortB for output \*/

DDRB = 0xFF;

/\*Timer/Counter1, Output Compare A,B Match Interrupt Enable\*/

TIMSK |= (1<<OCIE1A) | (1<<OCIE1B);

// Initialize Output Compare Register - 1A

TCCR1A=0b10100001; //NON INVERTING PWM

OCR1A = 100;

OCR1B =100;

TCCR1B |= (1<<WGM12) | (1<<CS12) | (1<<CS10);

}

/\*ISR(ADC\_vect)

{

i = ADCH;

ADCSRA |=(1<<ADSC); // ADC Start Conversion.

} \*/

/\* void ADC\_init()

{ DDRC &=!(1<<PC0); //Use ADC0 as input

ADMUX |=(1<<REFS0) | (1<<ADLAR); // AVcc as Vref.

// Channel Selected : ADC0 Left adjusted result

ADCSRA |=(1<<ADEN); // ADC Enable.

ADCSRA |=(1<<ADSC); // ADC Start Conversion.

ADCSRA |=(1<<ADFR); // ADC Free Running.

// Note ADFR is ADATE for other ATmega ICs.

ADCSRA |=(1<<ADIE); // ADC Interrupt Enable.

ADCSRA |= (1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0); // ADC Prescaler Select.

} \*/

int main (void)

{

Int\_init();

// To set port C as output

DDRC=0xFF;

// To set port B as output

DDRB=0xFF;

// To set port D as input

DDRD=0;

// To Enable global Interrupt

sei();

// ADC\_init();

square\_wave\_init();

k=0;

i=0;

l=1;

v=0;

PORTB=0;

PORTC=0;

for (;;) {

if (PORTC & (1<<5))

{

while((PIND & (1<<PD6)) | (PIND & (1<<PD7)))

;

if(PORTC & (1<<4))

PORTB |=9;

else

PORTB &=0xF6;

PORTC&=!(1<<5);

PORTC&=!(1<<4);

}

}

}

Future Improvements

Vertical control can be improved by using a sonar detector IC. It can be used as underwater surveillance system if we attach a camera to it. With a proper feedback system and analysis the vehicle can be used as AOV for object tracking, object follower.