Autonomous Underwater Agent "Pipin"

by

Marius Bratrein

for

EEL 5666 Intelligent Machines Design Laboratory

Dr. K.L. Doty
Dr. A.A. Arroyo

December 12, 1997

EXECUTIVE SUMMARY

The Pipin autonomous underwater agent was designed with the intention of making a low-cost and simple, yet fully functionable robot that will perform underwater. The design was kept electronically simple, so more time could be spent on mechanical engineering issues such as waterproofing the housing for the robot. Another major time-consuming component was designing and implementing the balancing scheme. The robot is now working to a degree where more advanced sensors and behaviour implementation is possible. Plans for future improvements include increasing its depth capabilities, and obstacle avoidance through sonar emission and signal processing.

TABLE OF CONTENT **Page**

INTRODUCTION

This report deals with designs and implementations for an autonomous underwater robot named "Pipin". A large part of the design for this robot is mechanical with the exception of a few sensor design challenges. I will describe some of the major obstacles of underwater robot design and how I dealt with them. I have developed a sensor that's unique to the underwater environment that I will explain, and also show some graphs and test results from testing my sensor. While reading through the report, it is important to remember the intention of the design; to keep it inexpensive and simple, yet functional. Better and more expensive underwater robots have already been built, so I set out to designing a simple, yet fairly robust submarine with a target price of somewhere under $500.

I start off describing the robot from a system level standpoint in order to give a broad view of the project. The key aspects of the robot, its environmental capabilities, and electro-mechanical features are described next. Although the biggest challenge of this project was robot platform motion underwater, I spent some time developing a full-system pressure sensor starting with a Honeywell silicone pressure sensor. I go into some detail about the design consideration for the support circuitry, along with a description of my initial testing and verification of the feasibility of the pressure sensing system.

HIGH-LEVEL SYSTEM DESCRIPTION

The robot is balanced to about one ounce negatively buoyant. The vertical motion control is done by a motor mounted on top of the robot pulling the robot up, while gravity takes care of the downwards motion. Dual side motors propel the robot in the horizontal plane. The digital hardware for system control consists of a modified Motorola 68HC11 Evaluation Board Unit connected to a Mekatronix ME11 expansion board. Figure 1 shows the schematic block diagram. Horizontal and vertical control is achieved through sensor feedback, and motor control. The buoyancy of the canister is offset be a weighting scheme that insures stability in two degrees of freedom.

All code is written in Interactive C from the M.I.T. 6.270 robotic course. Appendix A has all the test code in addition to the full behavior code.

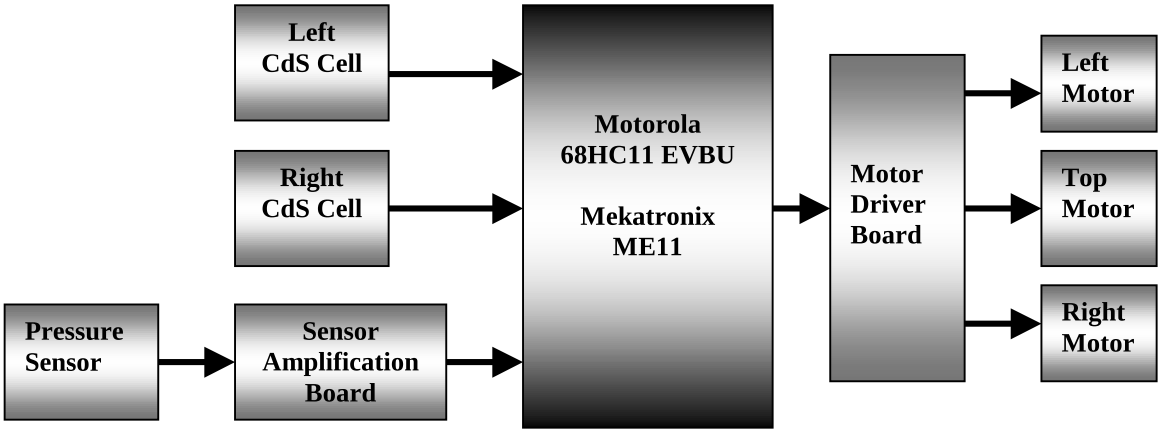


Figure 1. Schematic Block Diagram of Pipin.

MECHANICAL CHALLENGES

Housing

A design for an underwater cave-diving light by U.S. Navy Engineer Bill Gavin formed the basis of the robots housing. The design has a depth rating of well over 500 feet and has an excellent record of reliability. The housing consist of a cylindrical acrylic canister of ½ inch thickness with a glued-on bottom, and o-ring seal at the top. The lid on the top of the canister forms the top of the compression fitting, and is held down by three hasps mounted in approximately 180 degrees increments The acrylic is fairly easy to work with, and not too expensive. Jason Richards helped me with cutting , gluing and mounting hardware onto the acrylic. He makes dive lights as a hobby, and does excellent work.

Balance, Buoyancy and Trim

In modern submarines depth control is achieved through a ballast system using compressed air and water to change the buoyancy characteristics of the submarine. My idea of keeping things simple resulted in a design where the buoyancy of the robot is keep fairly closed to neutral with a slight negative error. The robot sinks to the bottom if it remains passive in the water column. A motor mounted on top of the robot provides upward propulsion while mother earth and gravity takes care of the robot sinking. In the robot proposal I had the vertical motor mounted on the bottom of the robot, pulling it down while the robot would float in passive mode. I had to reconsider this strategy because the propeller would be the first thing to hit the bottom when the robot was sinking, resulting in a very mechanically brittle design. Now I have the motor mounted in a helicopter like fashion, but on a very short arm in order to keep the center of gravity low. I added external weight to the robot in order to achieve lateral stability. The two metal pipes serve several purposes; the act as a keel for stability and they form the robots legs, being the first part to touch the bottom thereby protecting the propellers from hitting the bottom. The metal pipes along with a few lead pieces for fine-tuning the buoyancy were all centered in the sub to keep it from going nose up or down while travelling. I also mounted the motor arms at a third of the length of the canister resulting in the remaining two-thirds functioning as a rudder increasing forward motion stability.

ELECTRICAL CHALLENGES

One of the biggest considerations for designing electronic equipment for underwater use, is to prevent short-circuits for happening. Salts dissolved in water makes for good conductivity, and requires tight waterproof shielding of all electric parts. Interfacing through the robot housing to external parts such as motors is accomplished through sealed female banana plugs, and water-proof switches with rubber boots.

The motors run off the 10.6 volts battery and not the micro-controller digital output ports used to control them. It was therefore necessary to design and build a motor driver board using Darlington power transistors biased for switching mode operations. The circuits was outlined by teaching assistant Scott Jantz, and modeled and simulated by me. Figure 4 shows the motor driver circuit.

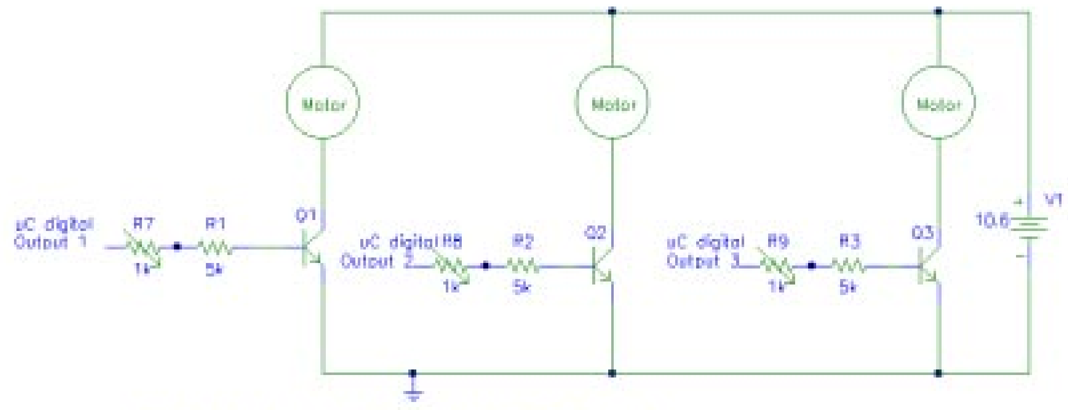


Figure 4. Motor Driver Circuit

Motor Shielding

The motors are the only electro-mechanical actuators on the robot, and some intricate problems exist here. It's difficult, expensive, and time-consuming to make an electric motor water-proof. The majority of the final stages of testing this robot will be done in local fresh-water springs and in pools where conductivity can be neglected. All motors used in this design have been burn-tested underwater to assure functionality. The motors run at the less efficient parts of the speed-torque curve, and are not very efficient energy vise. By using a powerful battery, and limiting pools runs to under 30 minutes this did not become a problem.

PRESSURE SENSOR DESIGN, IMPLEMENTATION AND TESTING

Physics of Water Pressure

The pressure exerted by water on a submerged object increases with increasing depth. Every 33 feet of freshwater results in a pressure change of one atmosphere. (1 ATM). Pressure can also be reported as pounds per square inch (PSI), relation to atmospheres in the conversion formula 1ATM = 14.7 PSI.

Physics of a Depth Sensor

The depth sensor is a sealed pressure sensor with an amplifying circuit. The pressure sensor consists of a resistor that varies with pressure applied to it, and when biased and put in a Wheatstone-bridge circuit (Fig.2) outputs a small voltage increasing with increasing pressure. The Honeywell 136 pressure sensor reads pressure greater than surface pressure which is approximately 1 atmosphere or 14.7 psi. The data sheet specifies that for each psi increase after this, an increase in output voltage of 2.3 mV can be expected. The maximum range of this pressure sensor is 100psi, which comes out to a depth of over 220 feet, more than plenty for my design.

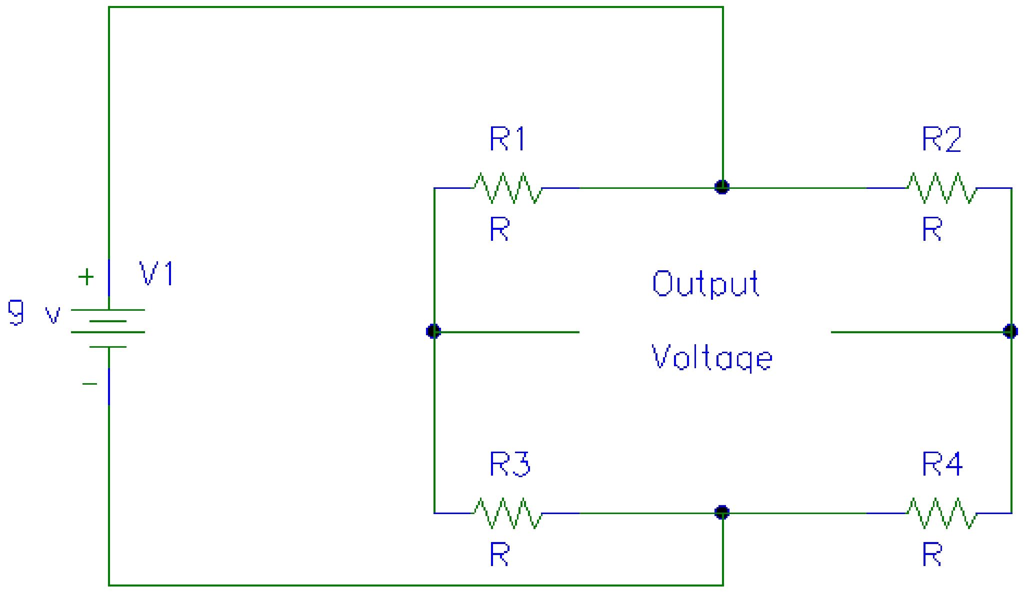


Figure 2.Schematic of the pressure sensor

Analog to Digital Conversion of Sensor Output

The Motorola 68HC11 analog to digital converter is 8-bit wide, and assumes 0-5 volts to be the range of your input signal. This results in a resolution of 5/256 v or 19.53 mv. If the Honeywell sensor was connected directly to the 68HC11 the first noticeable increase in pressure would happen at 19.53/2.3 = 8.49 psi or around 19 feet. Needless to say, that is not good enough resolution for a robot that will function in the 0-65 foot range. A DC-amplifier is connected between the pressure sensor and the A/D converter on the micro-controller board. This amplifier brings up the voltage so that five volts are fed into the 68HC11 when the robot reaches 66 feet, or 2 atmospheres pressure. The amplifier is non-trivial in the sense that the it needs a gain of around 75, but the output voltage has to use 0 volts as reference in order for the 68HC 11 to read it correctly. The A/D converter on the Motorola board is not protected against higher voltages than 5 volts so great care must be taken. Figure 3 shows a dc amplifying circuit that uses 0-5 volts as its output range. Operating amplifiers are ideal for this kind of circuit, but since we don't have access to both negative and positive voltages on the robot, and compromise is met by using an external battery. This battery will only drive the sensor, so current draw will be much less than for the battery supplying power to the motors. Single-supplied op-amps were considered for this application, but the drawback is that they bias the output node to the halfway point between 0 and Vcc thereby drastically reducing the range of the A/D converter, which is against the original intention of this dc amplifier.

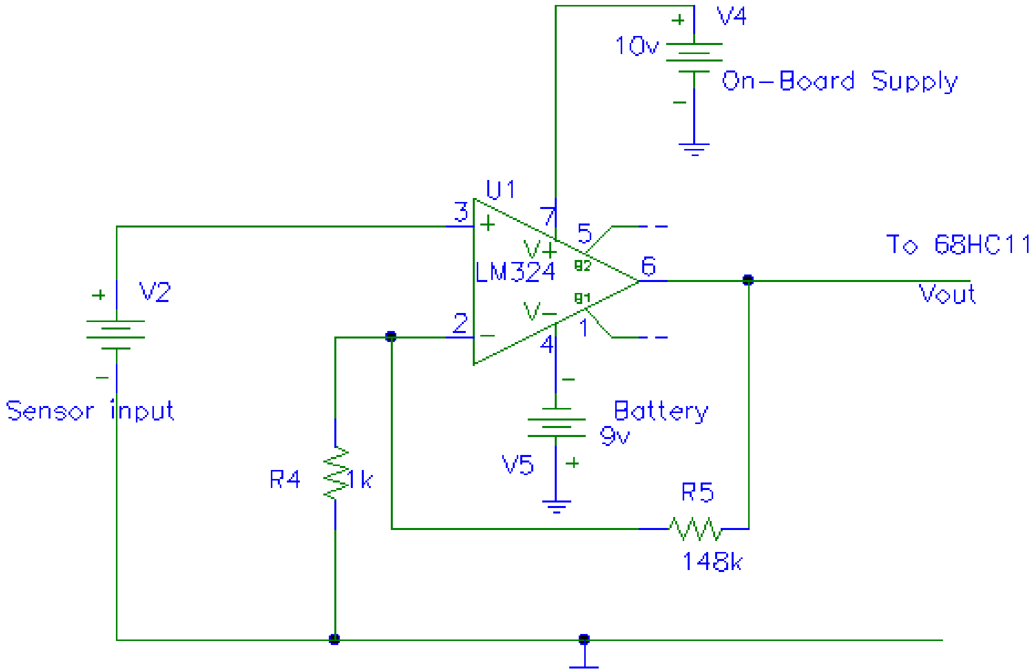


Figure 3. A/D conversion for Sensor

Test Results

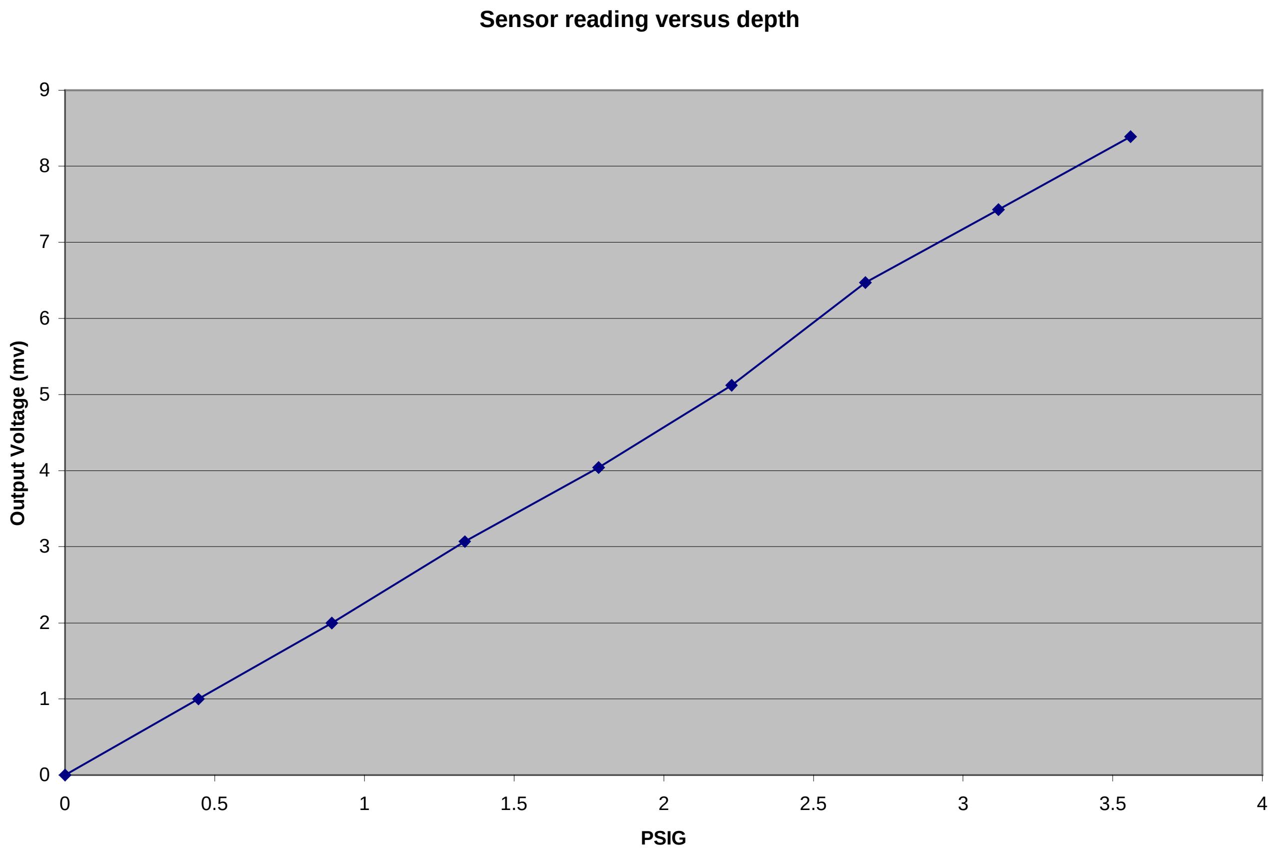
In-water tests were conducted with the pressure sensor to assure linearity. Although linearity is not a strict requirement for the functioning of the robot, it’s still a goal for the designer. The pressure sensor was biased with 9 volts, and taken to various depths in a pool while reading the output voltage with a volt meter. (see Table.1 for reference) A zero-reading was done on the surface and a 0 mV intercept was confirmed. At the time of testing the dc-amplifier was not working properly, so the measured output voltage is directly from the pressure sensor, before dc amplification. A digital depth gauge from Uwatec was used to record the ’actual’ depth for which these voltage measurements are referenced against. This depth gauge is accurate to about one feet, and only has increments of 1 feet. Table 1 shows output voltage readings versus depth gauge readings.

Table 1. Sensor reading versus depth

Depth (feet)	Pressure(psig)	Output voltage (mV)
0	0	0
1	.445	1
2	.891	2
3	1.336	3.07
4	1.782	4.04
5	2.227	5.12
6	2.673	6.47
7	3.118	7.43
8	3.56	8.39

Graphical Interpretation of Test Results

The results proved the sensor to be very linear at shallow depths. The readings at one and two feet should be disregarded since they were estimated. The digital depth gauge only works at three feet or deeper. The tiny discrepancies at six, seven and eight feet were due to the way the experiment were conducted. I did not dare to do any deeper measurements than what the pool allowed for as the pressure sensor is only temporarily mounted, and will not high pressure.



ENVIRONMENTAL TESTING

All testing was conducted in the swimming pool at Brandywine Apartments. This non-heated pool cooled rapidly as the semester progressed with a ending temperature in the low sixties. Compiling code and then having to travel 400 yards to test the code turned out to be time-consuming. Time delays in control functions were estimated at first, and several trips were needed to fine-tune the accuracy of the behaviors. There is much more work left to be done in optimizing the software for the aquatic environment.

In order to save time, I constructed a test board that emualtes the motors so that I could do testing in house without putting the whole assembly together, and not having to worry about motors spinning at high speeds, nor high battery drain. I also came up with a quick way of testing the pressure sensor. By adding a two inch length of surgical tubing for one to blow into, I was able to do some rough calibration and testing of the pressure sensor. This simple method also allows for easy demonstration of the sub in classroom demonstrations where water is not available.

CONCLUSION

In order to complete the proposed design, I had to go through several creative processes, as well, as revising ideas, and asking for outside assistance form teaching assistants and fellow IMDL students. Actual circuit design and verification through building real-life applications is seldom done in a class context, with this class as a major exception. I've learned tremendously about electro-mechanical interfacing, microprocessor control, along with the public speaking and report writing practice I have gotten. My interest in underwater robots does not stop at basic positioning and motion control. I have plans to incorporate more features when time permits. Robert Pitzer from MIL has offered me ultrasonic transmitter receiver pairs with high-order bandpass filter that I can use to incorporate sonar navigation or obstacle avoidance. I also want to extend the depth rating on my robot. The canister design is proven to several hundred feet depths, but as of now, the pressure sensor mounting is the weak point. I have plans to attached a bracket to support the sensor from behind. Most of my efforts this semester have been focused on the mechanical aspects to this design, leaving plenty of work in software coding and behavior control algorithms.

APPENDIX A

Module Test Code

Light Tracking

```
int left_cell;
int right_cell;

void main()
          {
      while(1) {
            left_cell  = analog(1);
                  right_cell = analog(2);

            if (left_cell > right_cell) {
                  poke(0x7000,02);
                  sleep(0.4);
                  }

            if (left_cell < right_cell) {
                  poke(0x7000,01);
                  sleep(0.4);
                  }
            }
      }
```

Depth Control

```
int left_cell;
int right_cell;
int depth;

void main()
          {
                  poke(0x7000,0x01);
                  sleep(2.0);

                  poke(0x7000,0x02);
                  sleep(2.0);
```

```
                        poke(0x7000,0x04);
                        sleep(2.0);


            while (1)
                        {

                        depth = analog(0);
                        if (depth > 10) { poke(0x7000,0x04);
                                                sleep(5.0);
                                                poke(0x7000,0x00);
                                                }

                left_cell  = analog(1);
                        right_cell = analog(2);

                if (left_cell > right_cell) {
                        poke(0x7000,02);
                        sleep(1.5);
                        }

                if (left_cell < right_cell) {
                        poke(0x7000,01);
                        sleep(1.5);
                        }

        }
```

Motor Test Code

```
/* MOTOR TEST CODE
/* bit 0 => left motor
/* bit 1 => right motor
/* bit 2 => top motor

/* analog(0) => pressure
/* analog(1) => left CdS cell
/* analog(2) => right CdS cell

void main()
        {
        poke(0x7000,0x03);      /* left AND right motors on
        sleep(3.0);             /* wait for 3 seconds
```

```
        poke(0x7000,0x00);      /* both motors off

        poke(0x7000,0x04);      /* top motor on
        sleep(5.0);             /* wait for 5 seconds
        poke(0x7000,0x00);      /* top motor off

        poke(0x7000,0x07);      /* all motors on
        sleep(3.0);             /* wait for 10 seconds
        poke(0x7000,0x00);      /* all motors off
        }
```

Complete Behavior Control Code

```
/*******************************************************/
/* CONTROL PROGRAM FOR AUTONOMOUS UNDERWATER  */
/*  AGENT "PIPIN"                              */
/*  BY MARIUS BRATREIN, FALL 1997, EEL5666     */
/*  INTELLIGENT MACHINES DESIGN LABORATORY     */
/*******************************************************/

/* HARDWARE DESCRIPTION                         */
/* the left/right two motors and the top motors */
/* are activated through digital outputs mapped */
/* to $7000, using darlington Q drivers.        */

/* Three A/D readings take place.               */
/* Pressure, Left, Right                        */

/* left motor is  bit 1, 0x01                   */
/* right motor is bit 2, 0x02                   */
/* top motor is   bit 3, 0x04                   */

/* pressure sensor is analog(0)                 */
/* left CdS sensor is analog(1)                 */
/* rightCDS sensor is analog(2)                 */

/********** VARIABLES *******************/
int bottom;
int surface;
int left;
int right;
int depth_old;
int depth_new;
int depth;
int cruise_depth;
```

```
/***** GET SURFACE *********************************************/
/* Immediately upon activating the program, the robot takes              */
/* the pressure reading and assumes it to be the surface value           */
/* variable = surface                                                    */
/****************************************************************/
void get_surface()
{
surface = analog(0);
}

/***** GET BOTTOM *******************************************/
/* The robot checks depth every 5 seconds.                              */
/* When there are no more depth changes, assume bottom depth            */
/* variable = bottom                                                    */
/**************************************************************/
void get_bottom()
        {
        depth_old = analog(0);
        sleep(5.0);
        depth_new = analog(0);

        while ( depth_new > depth_old ) {    depth_old = depth_new;
                                             sleep(5.0);
                                             depth_new = analog(0);
                                             }
        bottom = depth_new;
        }

/***** MAINTAIN DEPTH *************************************/
/* Check new depth. IF new depth > old depth (from main)               */
/* Run top rotor for 1 sec (approx)                                    */
/************************************************************/
void maitain_depth()
        {       depth = analog(0);
                if (depth>cruise_depth) {    poke(0x7000,0x04);   /* top motor on */
                                             sleep(5.0);           /* wait 1 sec   */
                                             poke(0x7000,0x00);   /* top motor off*/
                                }
        }
```

```
/***** MAINTAIN HEADING ******************************************/
/* Check sensor_left, sensor_right                                          */
/* IF left>right run right motor 1 second (approx) END function             */
/* IF right>left  run left motor 1 second (approc) END FUNCTION             */
/* end function                                                             */
/********************************************************************/
void maintain_heading();
        {       left = analog(1);               /* get left CdS cell  */
                right = analog(2);              /* get right CdS Cell */

                if (left>right) { poke(0x7000,0x02);            /* right motor on  */
                                sleep(1.0);                     /* wait 1 second   */
                                  poke(0x7000,0x00);            /* right motor off */
                                }


                else      { poke(0x7000,0x01);          /* left motor on   */
                            sleep(1.0);                 /* wait 1 second   */
                            poke(0x7000,0x00);          /* left motor off   */
                          }
        }

/***** GO STRAIGHT **************************************************/
/* Both motors on full throttle for three secs, slight turning :-(              */
/**********************************************************************/
void go_straight()
        {
        poke(0x7000,0x03);                      /* left AND right motors on   */
        sleep(3.0);                             /* wait 3 seconds                */
        poke(0x7000,0x03);                      /* left AND right motors off. */
        }

/***** MAIN ***************************/
/* first get surface pressure reading         */
/* then get bottom pressure reading           */
/* cruise_depth = (bottom-surface)/2          */
/* ROBOT will cruise at 1/2 pool depth       */
/* where it was calibrated                   */
/* Both motors on, wait 2 seconds            */
/* maintain depth, maintain heading          */
/* repeat                                    */
/*************************************/
void main()
      {
        get_surface();
        get_bottom();
```

```
cruise_depth=(bottom-surface)/2;      /* find mid-point*/

while (1) {       go_straight();              /* infinite loop */
                  maintain_depth();
                  maintain_heading();
            }
}
```