“Proto III”

M^{3}G ROV Team

West Carteret High School

2005 MATE National ROV Competition

Team Members:
Matt Backman, Matt Gildner, Greg Moore, Matt Stallsworth

Teachers and Mentors:
Tom Backman, Mike Gildner, Barbara Waters

Abstract

M^{3}G designed and engineered “Proto III”, a remotely operated vehicle (ROV), built for the 2005 National ROV Competition sponsored by Marine Advanced Technology Education (MATE). The M^{3}G team, from West Carteret High School in North Carolina, has created an innovative design that features a cylindrical vehicle made out of PVC piping with a six motor system. Dynamic propulsion with excellent stability allows the ROV to move quickly through the water and perform difficult tasks. Drive motors are controlled by a serial interface that is commanded by a computer and joystick. The team uses the joystick to “drive” the ROV. Tasks are performed by using simple pneumatic devices to release probes and attach modules. The M^{3}G team was challenged by several problems which were overcome with the help of our mentors and sponsors. With the support of these people and the team’s ingenuity M^{3}G was able to complete the ROV and prepare for their first national competition with high school and college teams across the country.

Design Rationale

The M^{3}G team decided to create a prototype based on a cylindrical ROV. They started off with six inch (15.24 cm) PVC pipe. It was cut down to the 80 cm size limits and screw-on PVC caps were used to secure the ends. This allows air to be trapped inside the ROV and serves as the positive buoyancy. By calculating the amount of water that is displaced, the team was able to figure how much weight should be added to the bottom of the pipe to keep neutral buoyancy and to add stability. Unfortunately, these caps were ineffective and water was able to leak through the threads, which threw off neutral buoyancy. With "Proto II" and "Proto III", the team utilized an eight inch (20.32 cm) PVC pipe, which created a larger cavity to place the electronics. The team changed the type of caps it was using. Instead of screw-on caps, aluminum rubber caps were used, which have a center feed through port. When these caps are screwed down, the rubber is pressed out and forms a tight seal against the PVC. These caps are used on both the top and the bottom of the ROV. The bottom cap is sealed on with 3M 5200 Marine sealer since it does not need to be taken off. There is a valve located on the bottom cap and it can be used to pressurize the ROV to test the cavity for leaks. The top cap has a hole in which we can secure our tether through. Also, securing bolts were put above the top cap to keep the cap from popping up as the unit was pressurized. Guides were placed in the cavity which held the wires for the motors and kept them from tangling and stopped the cap at fixed position.

In "Proto III", it was decided that four systematic holes would be cut in the PVC at the bottom to cut down on the drag. This will allow the ROV to move more efficiently through the water and give time to fully complete each task. The vehicle was weighted by adding a calculated amount of lead to the bottom. The disk was made by melting lead weights into a mold and then attaching it to the ROV. Having an improved water tight seal in the top and the weight at the bottom helped make "Proto III" very stable and it is almost certain to remain upright. To correct any buoyancy problems that might arise during the missions, a simple balloon will be used as a buoyancy bladder. The balloon will be placed above the lead and will be filled and deflated by and air line.

M^{3}G's ROV is controlled by six bilge pump motors. Two motors are vertical thrusters, two are forward/reverse/rotational motors, and two control the lateral movement. The two motors controlling forward and reverse thrust are 1100 gallon per hour (4165 liter/hour) bilge pump motors while the remaining four are 360 gph (1362 liter/hour) bilge pump motors. These motors were fitted on with PVC T's and elbows and then screwed in with bolts. The T's and elbows were attached to one inch (2.54 cm) PVC piping that went through the ROV. Holes were drilled in the ROV for the piping to slide through, and then it was glued on with epoxy. The propulsion is

provided by brass model boat propellers. Early testing used plastic airplane propellers, but they didn't provide enough thrust. New brass propellers were drilled out at a machine shop to use adjustable collets so that they would fit onto the motors properly.

Each of the tasks demands something different from our ROV. The first will require power and force. The second will require the ROV to know its surroundings and have good visuals and will demand accuracy. The third task will also be one of accuracy. The team decided that they would not need a special tool to complete the task involving the turning of the valve. It was simple enough that the lever could be pushed by the ROV itself. Tasks 2 and 3 will require the use of a tool that can grip the implements that will be transported. To perform this job, the team built a very basic pneumatic gripper. The pneumatic gripper on our ROV consists of an expandable balloon that is placed inside a two inch PVC pipe. An air line that runs down the tether will pump carbon dioxide in and out of the balloon. When the balloon expands, it forces the probe against the wall of the housing and when it contracts, the device is released. In this way, the probe for the second task can be dropped into its hole and the module from the third task can be released once it attaches to the Velcro target. Because the design of the arm is simple, there is less chance for malfunctions during the course of the competition. It also eliminates the need for an expensive, complicated arm.

There are at least three cameras on "Proto III". It was decided to have a camera pointing at the bottom of the pool, one pointing out to see what is in front of the ROV, and one movable camera to be attached to the appendage. Lasers are also used identify the target and to see how far away the ROV is from the bottom of the pool. They are set up at an angle and the two dots combine when it is a certain distance from the bottom.

"Proto III" control system relies on the use of relay and serial interface boards made by Electronic Energy Controls which are controlled from a laptop computer. The six motors are connected to the two relay boards, RH-8 which are commanded by the serial interface boards STA-16 and EX-16. The relay boards are wired so the polarity can be switched on the bilge pumps. The boards and the serial interface are placed inside the ROV airtight cavity. The team communicates to the serial interface with RS-232 commands via Cat-5 cable which connects to the computer. A special computer program was written in "C" language to send control commands to the serial interface. The program reads the USB joystick position and sends a signal down to the serial interface. The program works by setting certain thresholds for the joystick axes. Once the threshold is crossed, a signal is sent to the ROV and turns a motor on or off. This design provides intuitive motion with precise positioning of the ROV. The power comes from our twelve volt DC battery, which we have stored in the cart on the surface, which provides the control station for the ROV.

The tether was designed to be as light as possible, consisting of only 5 electrical lines and 3 air lines. The most massive line is for main power. This multi conductor cord is made out of twelve gauge copper wire and this helps to avoid excessive voltage loss due to the approximately 20 meter length. The signal for the serial interface is sent down the Cat 5 cable. The power for the cameras and video is sent through one line for each camera. Finally, carbon dioxide is sent down clear, plastic lines. An important problem was to keep the tether as neutrally buoyant as possible so it would not pull on the ROV as it moved about the pool. Closed cell foam covering and the air lines helps keep the tether buoyancy more neutral.

Because only 5 minutes are given to set up for the ROV competition, M^{3}G designed a control cart which we would pilot our ROV from. On the control cart is a LCD screen to view the video feeds from the cameras, gages for our air lines, switches to control air flow to the balloons, and switches to turn power on for certain parts of the ROV. Our batteries, CO_2 tank, and ROV can all be stored on this cart, which will enable us to easily transport the ROV and set it up.

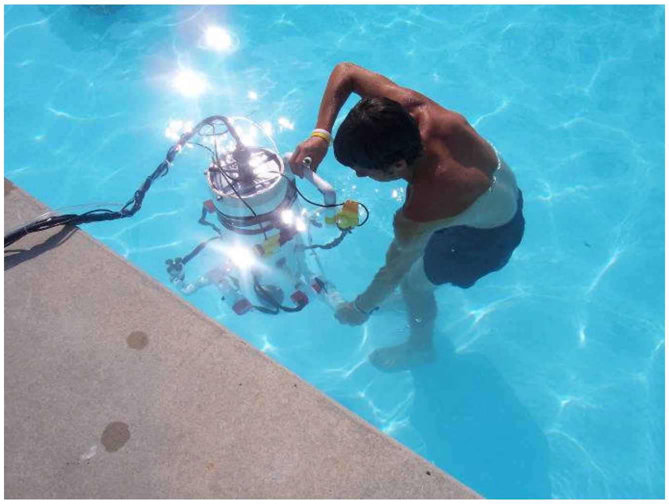
Challenges to Overcome

The ROV team encountered many challenges throughout the project. One of the most difficult challenges was making the ROV airtight to keep water out of the cavity. When water was able to get into the ROV, it affected the neutral buoyancy, causing the ROV to sink and become unable to ascend from the bottom of the pool. This was a problem with the caps from the beginning. The M^{3}G team started out trying to keep the air cavity sealed by using PVC screw-on caps. The top cap needed to be removable since the electronics are to be kept inside the cavity, so if something were to go wrong, then the team will be able to access them. The screw-on caps failed because water leaked in through the threads. Next, the team tried putting duct tape around the top of the screw-on cap. This failed too and led to the use of wax toilet bowl rings. They were placed inside the cavity as close to the bottom of the caps as possible. This worked better, but the rings were extremely messy and left a sticky residue on all it touched. The toilet bowl rings were too risky to use in case it got in the electronics and damaged them. The team went with entirely different caps for "Proto III". New caps which are made of rubber and metal were purchased. These caps are useful because they don't have any threads where water can leak. They attach by screwing down a handle. This applies pressure on the rubber and makes it expand outward. The rubber forms a nice seal against the PVC and was relatively leak free, but a very small mount of water was allowed to seep through a seam in the rubber. To keep this from happening, the team placed “monkey snot” on the rubber seam. “Monkey snot” is our slang for a non hardening sticky vinyl sealant used in weatherproofing for the telecommunications industry. The material is able to shape itself to the side of the PVC when pressure is applied to the rubber on the cap and forms a tight seal, keeping the water out. To simulate being in the water under more pressure, a bicycle pump was used to put pressure inside the airtight cavity for testing. When this was done, the cap tended to pop out. The team knew that water would get in if this happened in the water, so to prevent this from occurring, holes were drilled near the top of the PVC just above where the cap would rest. Bolts were placed in the holes and would keep the cap down, even under pressure.

Another challenge the M^{3}G had to over come was how to accomplish tasks 2 and 3. At first we did not want to use an arm at all to perform these tasks. For the second task where a probe had to be dropped, we evaluated hanging the probe from a hook on the bottom of the ROV. We anticipated that the probe would be easily placed in the junction box at the bottom of the pool. Unfortunately during testing, the probe kept falling off the hook accidentally when the ROV went to close to the bottom. For the third task, we tried to just place the module in a piece of PVC. The module would then pull out of the PVC once it attached to the Velcro. The module kept falling out of the PVC holder so this idea also did not work.

To secure the probe and the module, we decided to use a simple inflatable device. First we used an inflatable tube attached to a semi-rigid circular piece of plastic. This was taken off of a hospital oxygen mask. It was placed halfway inside a PVC tube so that it would fit around the probe and module and would hold them in place, when it was pumped up with air. When air was released from the tube, the probe and module were supposed to fall out. However, the tube fit too tight around the probe and module and they would not release them. This is what led us to use balloons. Because they were not rigid in anyway when deflated we knew they would work. Once we had tested them in the pool and were finally able to complete these tasks, we knew we had found the right design.

Troubleshooting Techniques

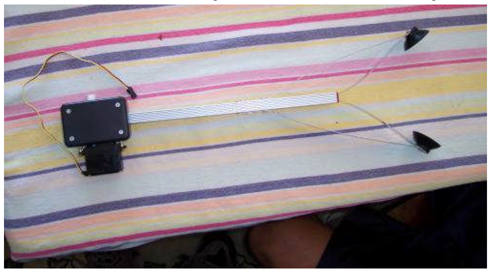


Different problems occurred during the testing. They were solved in different ways. To find out how much weight would be needed to achieve neutral buoyancy, lead fishing weights were attached to the ROV appendages. Once the ROV was neutral, the weight was added together. Testing for leaks was more difficult. Occasionally the team members would use scuba gear and go diving in the pool with the ROV. This allowed the team to find out where the leaks were coming from by observing the bubbles. Cameras were dry tested to decide what position they should be into to give the best view. The electronics, motors, and CO_2 lines were also able to be tested out of the water.

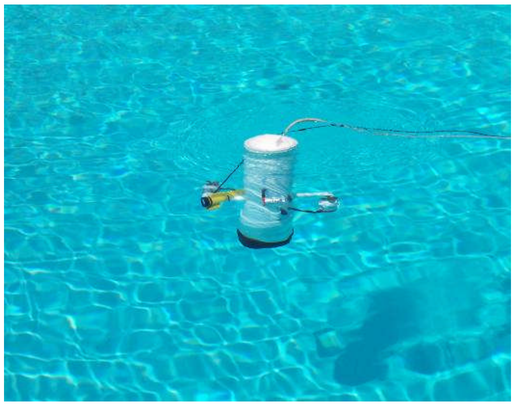
Because we wrote our own software to communicate with the serial interface, extensive debugging was accomplished before we could get the software working the way we wanted. Because all the software controlled the open/closed state of the relays based on a joystick's position, we took the relay boards out of the ROV to test them. Operational testing to determine how everything would work in the water resulted in lots of troubleshooting on the program at the pool side. When we learned that the ROV went fast backwards and that the joystick was too responsive, all we had to do was do a quick change to the program using a compiler and we were up and running. By correcting problems fast as they came, we were able to improve the ROV's performance without having to take anything apart or changing anything physically.

Lessons Learned

The ROV team had no experience constructing, designing, or using ROVs going into this competition. Everything was new to this team. One lesson that was learned by this team was the importance of keeping things simple and not rushing when building. Things were often more difficult when our ideas became more complex. For example, our arm was going to be controlled by a servo which closed a pick-up tool. The arm would have been easier to use if the tasks took place on land. Since the tasks take place in a pool, the arm needed to be waterproofed, making this idea much more challenging. It was also considered to have a tank of carbon dioxide to add pressure and use it to increase the buoyancy if needed. When it was



tested, the PVC split open and the cap shot off. Sometimes during the construction process, one of the members would rush and glue an appendage on before the placement and position was thought through. Over time the team learned to plan things out and to communicate with team mates before doing anything major change to the ROV.



The saying, “Practice makes perfect,” really came true for our ROV team. The majority of the time in the pool was spent correcting problems on our ROV and not practicing the tasks. Once construction of the ROV reached an end and most troubleshooting was completely, the team realized how far behind we were on practicing performing the tasks. We began to spend hours running to the tasks until we could do them repetitively.

Another lesson that was learned is that in the beginning you don’t necessarily learn what works, but what doesn’t work. The M^{3}G team learned what caps wouldn’t work, what type of bladders wouldn’t work, and the propellers that wouldn’t work. Eventually, you reach the point where you start to find things that do work and are usable on the ROV. By the process of engineering, testing, and elimination, the team discovered how to avoid simple flaws and build the ROV to successfully complete the intended goals.

Future Improvements

M^{3}G’s ROV is almost near completion. One of our secondary cameras has yet to be mounted. Other than that, any changes that will be made will most likely be to the pneumatic gripper. These small tweaks will result from experience we get as we practice more and more. Our buoyancy balloon also will have to be worked on before it is fully operationally.

Before we make our trip to Houston, we want to make “Proto III” to look as good as it can. The team is going to paint the ROV with a flame design. M^{3}G final ROV will be orange with blue flames. Also we are going to place illuminating tape on certain sections of that ROV to highlight important parts and to make them stand out underwater.

Last, our control cart will have to be finished. Once it has been completed, it will make practice trials a lot easier and we will be more efficient with out time.

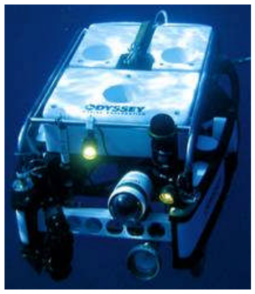
ROV's Currently in the News

ROV's are appearing more and more in the marine world. They can be used for many different purposes. ROVs are being used to identify different soniferous fish in the Stellwagen Bank National Marine Sanctuary (SBNMS). Rodney Roundtree and his group of scientists were able to use a ROV to learn more about fish habitat patterns and behaviors. With the ROV they were able to take video and audio samples of the fish. It is easier to send down an ROV to do the research, rather than human divers. With the use of ROVs, the group hopes to build their Soniferous Fish Locator device (SFL). The ROV will be able to serve as the platform for the SFL and carry it around.

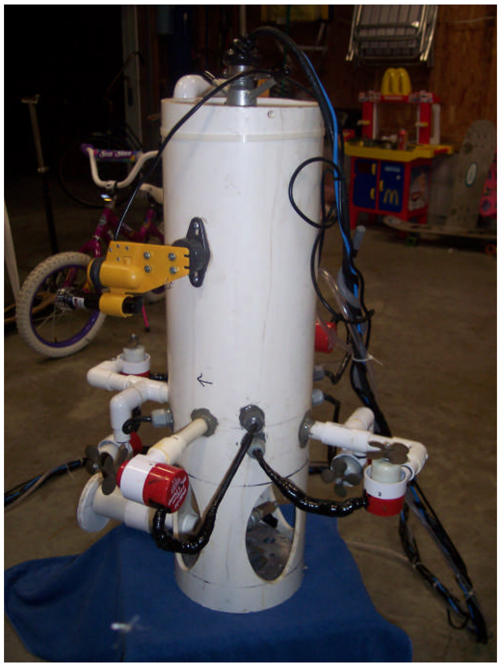
ROVs are also being used in the SBSMS to explore shipwrecks. In 2002 the shipwreck, Portland, was found using ROV's. ROVs were sent down to take footage of the shipwreck and determine that this was indeed the Portland. The ROV was able to view the marine life living around the ship and was able to get outstanding close-up inspections due to a new control system says Ivar Babb. With this new information, researchers were able to close the case of the lost passenger steamship. Without ROVs, the ship probably would not have been found. Side scans and video imaging of ROVs allowed this mystery to be solved.

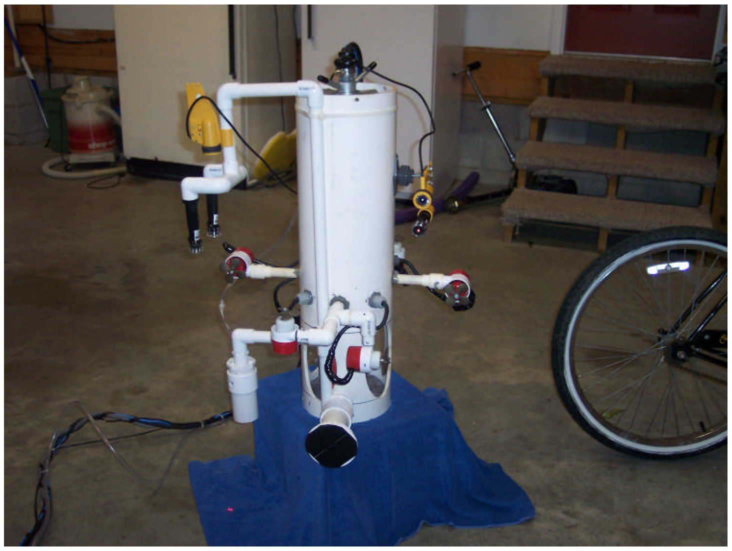
ROVs are also being used to map deep ocean habitats. They are able to do this with scanning devices and sonar. They are also able to go into the harsher environments such as the icy Arctic region where research is done in water too cold for divers to enter. The use of an ROV will protect human life by sending them into areas where pressures may be too great for a human to withstand.

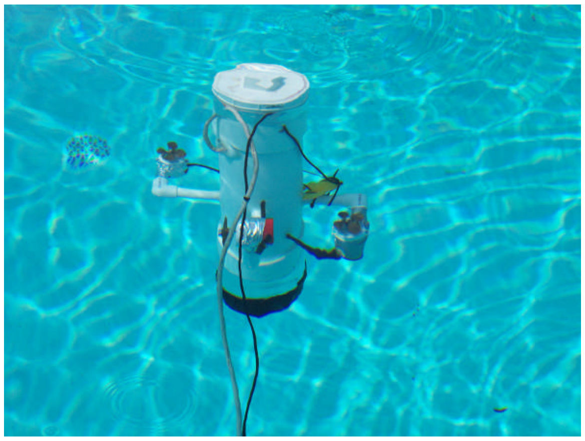
15 years ago, ROV's were not practical to be used for anything other than scientific research. With ROV's becoming cheaper and more available, the technology as trickled down to the field of treasure hunting. Greg Stemm and his treasure hunting company, *Odyssey Marine Exploration* now use ROV's to recover valuable wreaks that have been lost to humans for hundreds of years. Once he has researched were he thinks a wreak maybe, Stemm sends a investigation ROV to the seafloor to locate the wreck. This ROV will send live video feeds and maybe take up a few samples to bring back to the ship. If the data recovered supports a wreck being there, the treasure hunters will send down a salvage ROV, which will collect all the valuables and artifacts. These artifacts then can be sold for profits in the millions of dollars. Right now, *Odyssey* is salvaging its largest wreck ever, the *Sussex.* They expect to make a profit of over $2 billion when all is said and done.



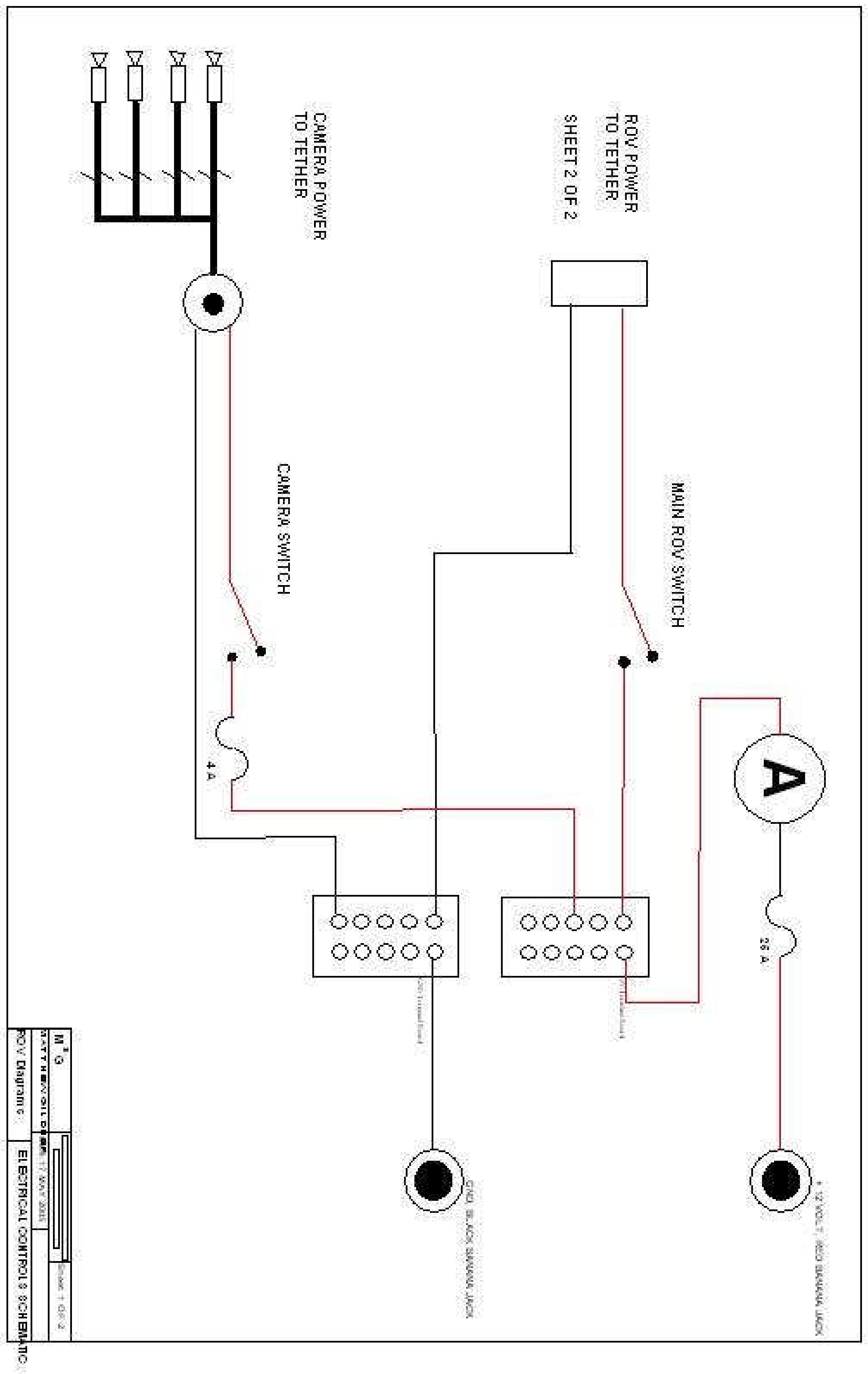
Photographs of “Proto III”



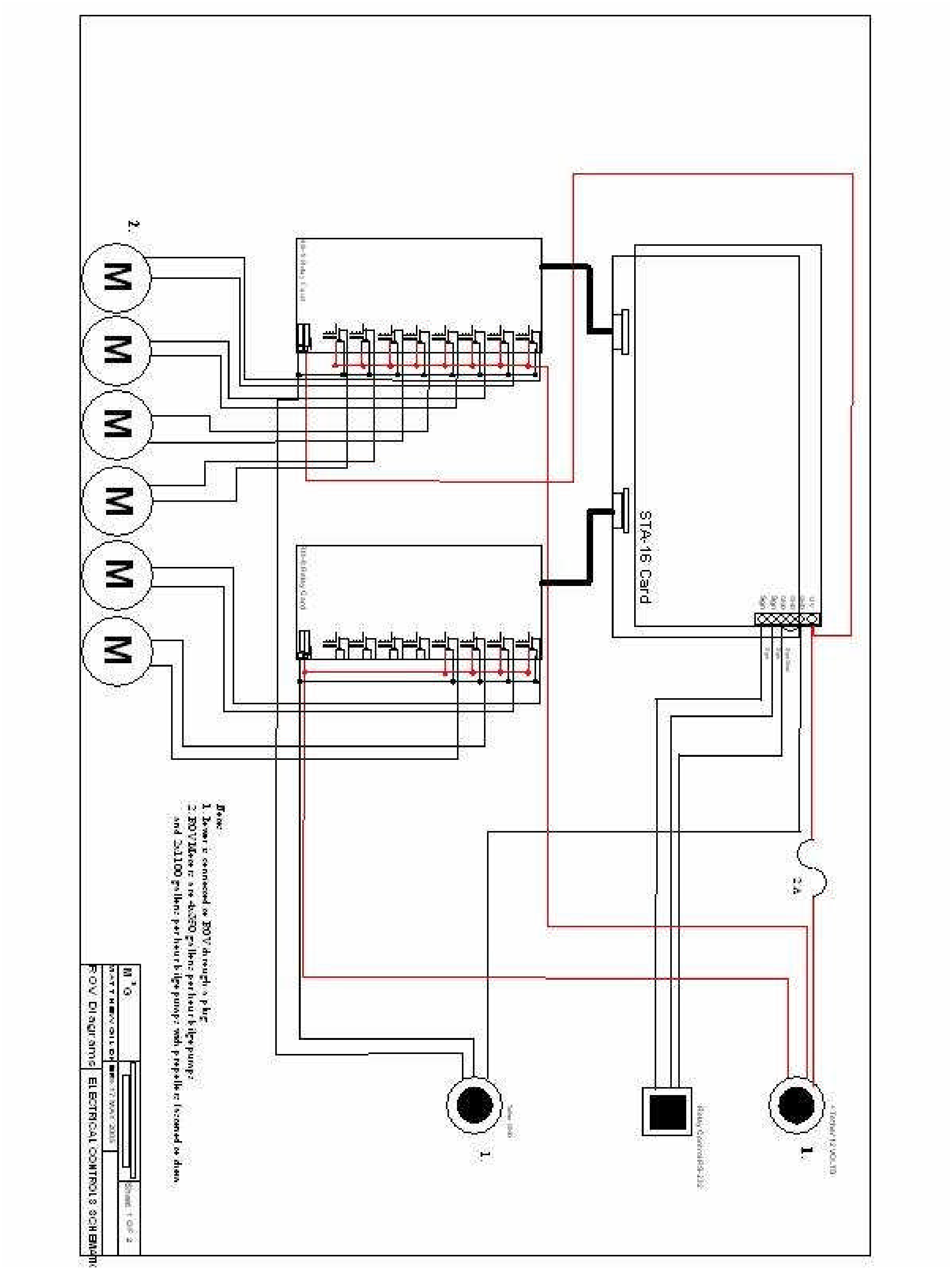




Electrical Schematic of Control Box



Electronic Schematic of Internal ROV Electronics


M
M
M
M
M
M
2.
STA-16 Card
2A
Tether 12 VOLTS
1.
1.
Note:
1. Power is connected to ROV through a plug
2. ROV Motors are 4x350 gallons per hour bilge pumps
and 2x1100 gallons per hour bilge pumps with propellers fastened to them
M3G
17 MAY 2005
Sheet 1 OF 2
ROV Diagrams
ELECTRICAL CONTROLS SCHEMATIC

M3G

ROV Software Program

```
//
// ROV Controller
//
// Uses joystick input to control relays to power ROV
// Logic:
//                   Set threshold to actuate relay closure
//                   If joystick position is greater that
threshold then turn on relay
//                   Shut off relay when joystick position
falls below threshold
//
// ---------------------------------------------------
// Compilation (lcc-win32):
//                 lc joy.c -o joy.exe -s
// ---------------------------------------------------
//

#include <stdio.h>
#include <mmsystem.h>
#include <windows.h>

#pragma lib <winmm.lib>
#pragma optimize(3)

#define COMM "COM1"
#define JOYSTICK 0
#define INTERVAL 300  // PCBUDDY operates internally on
22ms frame rate, so
                     // setting it bit higher here gives a
"safety buffer"

#define VER "ROV Controller"
/*
void InitCommPort();
HANDLE hCom;
*/

HANDLE InitCommPort();


int main(void) {

          //
          // init stuff
          //

          HANDLE hCom;
          HANDLE hCom2;
          HANDLE con;

          CONSOLE_SCREEN_BUFFER_INFO console_info;
          COORD cursor_home = {0,8}, cursor_zero = {0,0};

          DCB dcb;
          JOYINFOEX joy;
          //JOYINFO joy;

          DWORD wchars;
          char string[1024];
          unsigned char ch[10];
          unsigned char value;
          int POVthreshold, write2relay;
          int POVthreshold2, write2relay2;
          int zthreshold, write2relay3;
          int zthreshold2, write2relay4;
          int state=0;
          int savestate=0;
          int i=0;
          unsigned char sync;
          unsigned char servo;
          unsigned char servoposition;


          // Open and setup the serial port to the relay
controllers
          hCom=InitCommPort("COM1");

          hCom2=InitCommPort("COM6");

          // setting realtime priority is recommended in
order for all data to be sent at same time;
          // this application takes very little CPU time
and shouldn't cause any delays to the system;
          // on another hand if another application running
simultaneously (for example video decoder)
          // would be taking some CPU - it would cause
violent shaking to servos, etc.
          SetPriorityClass(GetCurrentProcess(),
REALTIME_PRIORITY_CLASS);

          // console i/o
          con = GetStdHandle(STD_OUTPUT_HANDLE);
          GetConsoleScreenBufferInfo(con, &console_info);
          SetConsoleCursorPosition(con, cursor_zero);
          FillConsoleOutputCharacter(con, ' ',
console_info.dwSize.X * console_info.dwSize.Y, cursor_zero,
&wchars);
          SetConsoleTitle("ROV Controller");

          printf("%s\n[Interface: Serial] [IN: JOY%d] [OUT:
%s] [FR: %dms] [PRI: RealTime]\n", VER, JOYSTICK, COMM,
INTERVAL);

          // joystick i/o
          printf("Number of Joystick devices: %d, Using
device: %d.\n", joyGetNumDevs(), JOYSTICK);

          printf("====[ STATUS
]============j============================================
\n\n        Joystick     Channel\n -----------------------
---");

/*
          value = 33|35|37|39|41|43|45|47;
          value = 32|34|36|38|40|42|44|46;

          for(i=32;i<64;i=i+2){
                    value=(unsigned char)i;
                    WriteFile(hCom,&value,1,&wchars,NULL);

                    sleep(1000);
                    printf("%d\n",i);
          }
printf("hello on\n");
          for(i=33;i<48;i=i+2){
                    value=(unsigned char)i;
                    WriteFile(hCom,&value,1,&wchars,NULL);

                    sleep(1000);
                    printf("%d\n",i);
          }
printf("hello off\n");
exit(0);
*/
          //
          // main loop
          //y axis
          POVthreshold = 0;
          write2relay = 0;
          POVthreshold2 = 0;
          write2relay2 = 0;
          zthreshold = 0;
          write2relay3 = 0;
          zthreshold2 = 0;
          write2relay4 = 0;
          sync = 255;
          servo = 0;
          servoposition = 0;

          while(joyGetPosEx(JOYSTICK,
&joy)==JOYERR_NOERROR) {

          //servo control

                    if (joy.dwButtons==64){
                              servoposition = 750;

          WriteFile(hCom2,&sync,1,&wchars,NULL);

          WriteFile(hCom2,&servo,1,&wchars,NULL);

          WriteFile(hCom2,&servoposition,1,&wchars,NULL);
                    }
                    if (joy.dwButtons==128){
                              servoposition = 0;

          WriteFile(hCom2,&sync,1,&wchars,NULL);

          WriteFile(hCom2,&servo,1,&wchars,NULL);

          WriteFile(hCom2,&servoposition,1,&wchars,NULL);
                    }
          //motor control

                    savestate=state;

          if((joy.dwYpos>300)&(joy.dwYpos<700)&(joy.dwRpos<
220)&(joy.dwRpos>40)) //do nothing
                              state=0;
                    else
if((joy.dwYpos>700)&(joy.dwRpos<220)&(joy.dwRpos>40)) //go
forward
                              state=1;
                    else
if((joy.dwYpos<300)&(joy.dwRpos<220)&(joy.dwRpos>40)) //go
```

```
back
                              state=2;
                    else
if((joy.dwYpos>300)&(joy.dwYpos<700)&(joy.dwRpos>220)&(joy.
dwRpos<250)) //rotate right
                              state=3;
                    else
if((joy.dwYpos>300)&(joy.dwYpos<700)&(joy.dwRpos<40)&(joy.d
wRpos>20)) //rotate left
                              state=4;
                    else
if((joy.dwYpos>700)&(joy.dwRpos>220)&(joy.dwRpos<250))
//forward, rotate right
                              state=5;
                    else
if((joy.dwYpos>700)&(joy.dwRpos<40)&(joy.dwRpos>20))
//forward, rotate left
                              state=6;
                    else
if((joy.dwYpos<300)&(joy.dwRpos>220)&(joy.dwRpos<250))
//back, rotate right
                              state=7;
                    else
if((joy.dwYpos<300)&(joy.dwRpos<40)&(joy.dwRpos>20))
//back, rotate left
                              state=8;
                    else
if((joy.dwYpos>300)&(joy.dwYpos<700)&(joy.dwRpos>250))
//rotate right far
                              state=9;
                    else
if((joy.dwYpos>300)&(joy.dwYpos<700)&(joy.dwRpos<20))
//rotate left far
                              state=10;
                    else
if((joy.dwYpos>700)&(joy.dwRpos>250)) //forward, rotate
right far
                              state=11;
                    else
if((joy.dwYpos>700)&(joy.dwRpos<20)) //forward, rotate left
far
                              state=12;
                    else
if((joy.dwYpos<300)&(joy.dwRpos>250)) //back, rotate right
far
                              state=13;
                    else
if((joy.dwYpos<300)&(joy.dwRpos<20)) //back, rotate right
far
                              state=14;
                    else
if((joy.dwPOV<31600)&(joy.dwPOV>22400)) //trim right
                              state=15;
                    else
if((joy.dwPOV>4600)&(joy.dwPOV<13400)) //trim left
                              state=16;
                    else state=-1; //error condition

                    if (state!=savestate){
                              switch(state){
                                        case 0:
                                                  for(i=0;
i<16; i++)

          ch[i]=i*2+32;
                                                  break;
                                        case 1:
                                                  ch[0]=33;
                                                  ch[1]=34;
                                                  ch[2]=37;
                                                  ch[3]=38;
                                                  break;
                                        case 2:
                                                  ch[0]=32;
                                                  ch[1]=35;
                                                  ch[2]=36;
                                                  ch[3]=39;
                                                  break;
                                        case 3:
                                                  ch[0]=33;
                                                  ch[1]=34;
                                                  ch[2]=36;
                                                  ch[3]=38;
                                                  break;
                                        case 4:
                                                  ch[0]=32;
                                                  ch[1]=34;
                                                  ch[2]=37;
                                                  ch[3]=38;
                                                  break;
                                        case 5:
                                                  ch[0]=33;
                                                  ch[1]=34;
                                                  ch[2]=36;
                                                  ch[3]=38;
                                                  break;
                                        case 6:
```

```
                              ch[0]=32;
                              ch[1]=34;
                              ch[2]=37;
                              ch[3]=38;
                              break;
                    case 7:
                              ch[0]=33;
                              ch[1]=34;
                              ch[2]=36;
                              ch[3]=38;
                              break;
                    case 8:
                              ch[0]=32;
                              ch[1]=34;
                              ch[2]=37;
                              ch[3]=38;
                              break;
                    case 9:
                              ch[0]=33;
                              ch[1]=34;
                              ch[2]=36;
                              ch[3]=39;
                              break;
                    case 10:
                              ch[0]=32;
                              ch[1]=35;
                              ch[2]=37;
                              ch[3]=38;
                              break;
                    case 11:
                              ch[0]=33;
                              ch[1]=34;
                              ch[2]=36;
                              ch[3]=39;
                              break;
                    case 12:
                              ch[0]=32;
                              ch[1]=35;
                              ch[2]=37;
                              ch[3]=38;
                              break;
                    case 13:
                              ch[0]=33;
                              ch[1]=34;
                              ch[2]=36;
                              ch[3]=39;
                              break;
                    case 14:
                              ch[0]=32;
                              ch[1]=35;
                              ch[2]=37;
                              ch[3]=38;
                              break;
                    default:
          }
for(i=0; i<4; i++)
WriteFile(hCom,&ch[i],1,&wchars,NULL);
}
// Z controls up and down
if(joy.dwZpos<64){
          //turn on relay 1
          ch[0]=41;
          ch[1]=45;
          if(!zthreshold){
                    write2relay3=1;
          }
          zthreshold = 1;
}
else {
          //turn off relay 1
          ch[0]=40;
          ch[1]=44;
          if(zthreshold){
                    write2relay3=1;
          }
          zthreshold = 0;
}

if(write2relay3){
WriteFile(hCom,&ch[0],1,&wchars,NULL);
WriteFile(hCom,&ch[1],1,&wchars,NULL);
write2relay3=0;
printf("Ch[0]: %d, Ch[1]: %d \n",
ch[0], ch[1]);
}
//lower the rov
if(joy.dwZpos>192){
          //turn on relay 2
          ch[0]=43;
          ch[1]=47;
          if(!zthreshold2){
                    write2relay4=1;
          }
          zthreshold2 = 1;
}
else {
          //turn off relay 2
```

```
                              ch[0]=42;
                              ch[1]=46;
                              if(zthreshold2){
                                        write2relay4=1;
                              }
                              zthreshold2 = 0;
                    }

                    if(write2relay4){

          WriteFile(hCom,&ch[0],1,&wchars,NULL);

          WriteFile(hCom,&ch[1],1,&wchars,NULL);
                              write2relay4=0;
                                        printf("Ch[0]: %d,
Ch[1]: %d \n", ch[0], ch[1]);
                              }


          // trim right

          if((joy.dwPOV<31600)&(joy.dwPOV>22400)){

                              ch[0]=49;
                              ch[1]=53;
                              if(!POVthreshold){
                                        write2relay=1;
                              }
                              POVthreshold = 1;
                    }
                    else {
                              //turn off relay 1
                              ch[0]=48;
                              ch[1]=52;
                              if(POVthreshold){
                                        write2relay=1;
                              }
                              POVthreshold = 0;
                    }

                    if(write2relay){

          WriteFile(hCom,&ch[0],1,&wchars,NULL);

          WriteFile(hCom,&ch[1],1,&wchars,NULL);
                              write2relay=0;
                              printf("Ch[0]: %d, Ch[1]: %d
\n", ch[0], ch[1]);
                    }

                    //trim left
                    if((joy.dwPOV>4600)&(joy.dwPOV<13400)){
                              //turn on relay 2
                              ch[0]=51;
                              ch[1]=55;
                              if(!POVthreshold2){
                                        write2relay2=1;
                              }
                              POVthreshold2 = 1;
                    }
                    else {
                              //turn off relay 2
                              ch[0]=50;
                              ch[1]=54;
                              if(POVthreshold2){
                                        write2relay2=1;
                              }
                              POVthreshold2 = 0;
                    }

                    if(write2relay2){

          WriteFile(hCom,&ch[0],1,&wchars,NULL);

          WriteFile(hCom,&ch[1],1,&wchars,NULL);
                              write2relay2=0;
                              printf("Ch[0]: %d, Ch[1]: %d
\n", ch[0], ch[1]);
                    }

                    // Console window update
                    snprintf(string, 1024, " X: %12d  ->
1:%4d   \n Y: %12d  ->  2:%4d   \n Z: %12d  ->  3:%4d    \n
R: %12d  ->  4:%4d      \n POV: %12d  ->  5;%4d     \n
Buttons: %12d  ->  6;%4d      \n",
                                                  joy.dwXpos,
ch[0],       joy.dwYpos, ch[1],       joy.dwZpos, ch[3],
joy.dwRpos, ch[4],     joy.dwPOV, ch[5],
(int)servoposition, ch[6]);

                    SetConsoleCursorPosition(con,
cursor_home);
                    WriteConsole(con, string,
strlen(string), &wchars, NULL);

                    sleep(INTERVAL);

          }
```

```
          return 0;
}
HANDLE InitCommPort(comname)
char comname[256];
{
          DCB dcb;
          COMMTIMEOUTS to;
          DWORD dwError;
          BOOL fSuccess;
          printf("in INIT: %s\n", comname);
          HANDLE hCom;

          hCom = CreateFile( comname,
                    GENERIC_READ | GENERIC_WRITE,
                    0,    // comm devices must be opened
w/exclusive-access
                    NULL, // no security attributes
                    OPEN_EXISTING, // comm devices must use
OPEN_EXISTING
                    0,    // not overlapped I/O
                    NULL  // hTemplate must be NULL for
comm devices
                    );
          if (hCom == INVALID_HANDLE_VALUE)
          {
                    // handle error
                    dwError = GetLastError();
                    printf("Could not open port");
                    return(NULL);
          }
          // Omit the call to SetupComm to use the default
queue sizes.
          fSuccess = SetupComm(hCom,256,256);
          if (!fSuccess)
          {
                    // Handle the error.
                    printf("SetupComm failed");
                    return(NULL);
          }
          // Get the current configuration.
          fSuccess = GetCommState(hCom, &dcb);
          if (!fSuccess)
          {
                    // Handle the error.
                    printf("GetCommState failed");
                    return(NULL);
          }
          // Fill in the DCB: baud=9600, 8 data bits, no
parity, 1 stop bit.
          dcb.BaudRate = 9600;
          dcb.ByteSize = 8;
          dcb.Parity = NOPARITY;
          dcb.StopBits = ONESTOPBIT;
          dcb.XoffChar = 0X13;
          dcb.XonChar = 0X11;
          dcb.fDsrSensitivity = 0;
          dcb.fRtsControl = RTS_CONTROL_DISABLE;
          dcb.fParity = NOPARITY;
          dcb.fOutxCtsFlow = FALSE;     // enable CTS
transmit handshaking
          dcb.fOutxDsrFlow = FALSE;
          dcb.fDtrControl = DTR_CONTROL_ENABLE; // DTR
disabled, but allows controll lines
          dcb.fDsrSensitivity = FALSE;
          dcb.fTXContinueOnXoff = TRUE; //
          dcb.fOutX = FALSE;            // disable xon/xoff
completely
          dcb.fInX = FALSE;             //
          dcb.fErrorChar = FALSE;
          dcb.fNull = FALSE;
          dcb.fBinary = TRUE;
          dcb.fRtsControl = RTS_CONTROL_ENABLE; // RTS
enabled but no handshake & allow controll lines
          dcb.fAbortOnError = FALSE;    // continue on
error...(there should be error handling implemented
instead...)

          fSuccess = SetCommState(hCom, &dcb);
          if (!fSuccess)
          {
                    // Handle the error.
                    printf("SetCommState failed");
                    return(NULL);
          }
          //set read timeout to return immediately if no
bytes are read
          memset(&to,0,sizeof(to));
          to.ReadIntervalTimeout = MAXDWORD;
          to.ReadTotalTimeoutConstant = 0;
          to.ReadTotalTimeoutMultiplier = 0;
          SetCommTimeouts(hCom,&to);

          return(hCom);
}
```

Acknowledgments

- BRR Technologies, Financial and technical support, Morehead City, NC
- Ken O'Leary, donation of laptop computer, Wilmington, NC
- Dave Hartman, Russell Heat Transfer; Ellicott City, MD-Donation
- Soundview Rotary Club; Morehead City, NC-Donation
- Styron & Styron Insurance; Morehead City, NC-Donation
- Philip S. Church, WCOM 101.7; Morehead City, NC-Donation
- Arnold Sanderson, Jr., Connecting Point Computers; Morehead City, NC-Donation
- Sam Rollings, Coastal Critters-DBA Subway; Morehead City, NC-Donation
- Lookout Rotary Club of Morehead City, NC-Donation
- William F. & Janie D. Taylor; Morehead City, NC-Donation
- Richmond & Elizabeth Wright; Morehead City, NC-Donation
- Masterclean; Morehead City, NC-Donation
- Dr. Jeff Moore-Ideas during the meetings and helpful thoughts
- Joe Mitchell-Help constructing the cart for transporting the ROV and for use of vehicle and trailer to get to Houston, TX.
- John Weeks-Help fitting the brass propellers onto the adapters for the motors
- Joe Jones, Neosis, donation of photoluminescence material, Morehead City, NC

Budget/Expense Sheet

Team Name M3G
Instructors Tom Backman, Mike Gildner Barbara Waters

Date	Description	Notes	Amount	Balance
12/7/2004	BRR Technologies	Morehead City	$3,000.00	$3,000.00
12/20/2004	Gas and food expenses	Trip to Cape Fear CC	-58.55	2941.45
1/14/2005	MATE support check	#17473	100.00	3041.45
12/29/2004	washers	Williams Hardware	-4.21	3037.24
12/28/2004	4" HT shrink tub, 3/4" heat shrinks, 2 primary wire, ICLM	Lowe's	-28.16	3009.08
12/29/2004	Great Stuff Twin, 2 4" HT shrink tub, TEF tape .5x250", 3/4" coupling, TEF tape .5x260", cleaner 4 oz., 4 oz., all purpose cleaner, 2 3/4"x5" 540 PVC pipe, 2 6" MIPT	Lowe's	-20.22	2988.86
12/28/2004	Plug PVC, 2 6" SPX FIDT ADPT PV, 6"x10' PVC	Lowe's	-70.02	2918.84
12/27/2004	2 PK4 AGC 4 amp, 2 proj box, 2 4 fuse block, 2 heat shrink color, 2 4" heat shrink color, 4 DPDT toggle switch, 2 PK 36 shrink 1" PCS, 1 PK 8 jumper leads, 2 8 POS bus strip, 1 8 terminal strip, 1 6 terminal strip, 90' #22 solid UL		-77.45	2841.39
12/26/2004	8 Dumas plastic prop 1/8", 2 Hobbico prop adaptor, Superstar EP&ARF	Tower Hobbies	-20.57	2820.82
1/9/2005	Hobbico Prop Adaptor Propellers (2)	Tower Hobbies	-29.43	2791.39
12/29/2004	2 chrome-plated rods 1/8" dia., 3 screw shaft collar	McMaster-Carr	-21.12	2770.27
12/29/2004	2 screw shaft collars 3/8", 1 screw shaft collar 1/4"	McMaster-Carr	-16.34	2753.93
12/27/2004	1 each: green, red, yellow, brown tape	Country Electric Supply Co	-14.86	2739.07
12/27/2004	4 bilge pump 360 gph, 2 con-butt, 1 ammeter 0-25 amps	West Marine	-131.04	2608.03
2/5/2005	LOG Extreme 3D Pro Joystick	On Cue	-37.44	2570.59
2/10/2005	Donation-Carrier Commercial Refridgeration	Dave Hartman, Russell Heat Transfer	200.00	2770.59
2/8/2005	20 8.5"x11" copies	UPS Store	-2.14	2768.45
2/12/2005	8 oz. All purpose cement, 4 1/2" couplings, 1/2x5' PVC pipe, 1/2" cap sch 40, 2 1/2" PVC cap, 1/2" ball valve PE, 1/2" cap thrd sch 4, 2 wax bowl ring, 1/75 oz. No. 5, Thr. 1/2"x 5' 540 PVC	Lowe's	-18.31	2750.14
2/12/2005	1 standard servo	GBI Hobbies	-16.05	2734.09
2/12/2005	PK 30 4" wire ties, proj box 3"x2"x1"	Radio Shack	-4.47	2729.62
2/12/2005	alum, flat 1/8", 1 1/2" gripper PA	Lowe's	-3.21	2726.41
2/19/2005	bilge pump 1100 gph	West Marine	-37.44	2688.97
2/19/2005	3 3/4" T PVC, 8 1"x3/4" 5x5 PVC 2 3/4" cap CPVC, 2 3/4" 900 EL CPVC, 4 3/4"x5' PVC	Lowe's	-24.03	2664.94

2/19/2005	2 1/2"x5' type M Co, 3/4"x5' 540 PVC pipe, 2 1/2" copper cap		-10.31	2654.63
2/24/2005	6 #3129 bronze propellers 3" dia.	Dumas Products	-185.00	2469.63
3/6/2005	1 brass pipe brush, 1 brass pipe nip, Ch. Tank safety valve, 1/4" female brass, 1/4" ball valves, 1"x1/2" galv red, 1" galvanized CA, 1/4" brass m PLU, JH Teflon tape	Lowe's	-20.88	2448.75
3/31/2005	Donations #3430, #3431	Soundview Rotary Club	600.00	3048.75
3/30/2005	Donation #5779	Styron & Styron Insurance	500.00	3548.75
3/30/2005	Donation	Philip S. Church WCOM 101.7	50.00	3598.75
3/30/2005	Donation #9388	Connecting Point Computers, Arnold Sanderson, Jr.	100.00	3698.75
3/30/2005	Donation #2811	Coastal Critters-DBA Subway, Sam Rolling	100.00	3798.75
3/30/2005	Donation #1382	Lookout Rotary Club	500.00	4298.75
3/30/2005	Donation #2030	William F. & Janie D. Taylor	100.00	4398.75
4/1/2005	Donation #8048	Richmond & Elizabeth Wright	100.00	4498.75
2/18/2005	Newman Tools, Inc.		-377.02	4121.73
3/26/2005	Luigi's Pizza for meeting		-29.96	4091.77
3/26/2005	2 3/4"x10' 540 PVC, 4 3/4" T sch 40, 6 3/4" ell sch 40, 2 1.75 oz. No. 5 Thr, 2 backer rod FGAPS, 3 3/4" heat shrinks, 5 4" HT shrink tub, 16 14 AWG terminal		-40.57	4051.2
3/16/2005	serial servo control, PC cable, shipping costs	Scott Edwards Electronics	-59.00	3992.2
3/25/2005	S-Video amplifier	Radio Shack	-5.32	3986.88
3/26/2005	1 bilge pump 1100 gph, 4 bilge pump 360 gph, 1 caulk 5200 3 oz.	West Marine	-121.92	3864.96
3/27/2005	JH JB Weld epoxy	Lowe's	-4.22	3860.74
3/29/2005	2 1/2 non-met. Cord connect, 4 oz. Blk liquid tap	Williams Hardware	-11.74	3849
4/4/2005	4 port mini USB HUB	Radio Shack	-10.69	3838.31
4/4/2005	75 Cat 5 EPR RSER	Lowe's	-12.04	3826.27
4/8/2005	In-Line coupler	Lowe's	-3.18	3823.09
4/10/2005	jump starter, batteries, ATC 25 fuses	Wal-Mart	-56.43	3766.66
3/8/2005	reaching tool	Walgreen's	-10.69	3755.97
4/9/2005	1/4"x5' PEX pipe	Lowe's	-1.80	3754.17
4/10/2005	2 6 mm collett style prop adaptor, 8 3.2 mm collett style prop adaptor	Park Flyer Motors	-56.00	3698.17
4/10/2005	caulk	West Marine	-11.76	3686.41
4/11/2005	2 2.5 oz. LD FREE .032	Radio Shack	-6.40	3680.01
4/11/2005	14 4" HT shrink tub., 2 thumb screws 10-24	Lowe's	-25.37	3654.64
4/11/2005	THPE LQD ELCT/ 4 oz. Bk	West Marine	-9.08	3645.56
4/11/2005	8 ultraviolet collett for 3	Hobby Lobby International	-43.19	3602.37

4/16/2005	12 oz. Foam insulation	Lowe's	-4.79	3597.58
4/16/2005	3 wmold wire channel, Bosch 18 pc. Titanium, marine goop 3.7 oz., 4 oz. All purpose cement, 4 oz. Cleaner, 3/4" slvr & Deming, 4 carriage bolts, 4 Hex nuts	Lowe's	-100.95	3496.63
4/18/2005	2 underwater video cameras-color	Ebay	-123.98	3372.65
4/19/2005	2 underwater video cameras-color	Ebay	-125.00	3247.65
5/10/2005	donation for ROV project	Masterclean	150.00	3397.65
5/25/2005	hardware and travel expenses		-4900.00	-1502.35