AC 2007-2150: INTERNET-CONTROLLED UNDERWATER VEHICLE

Omer Farook, Purdue University-Calumet

OMER FAROOK is a member of the faculty of the Electrical and Computer Engineering Technology Department at Purdue University Calumet. Professor Farook received the Diploma of Licentiate in Mechanical Engineering and BSME in 1970 and 1972 respectively. He further received BSEE and MSEE in 1978 and 1983 respectively from Illinois Institute of Technology. Professor Farook's current interests are in the areas of Embedded System Design, Hardware – Software Interfacing, Digital Communication, Networking, C++ and Java Languages.

Alan Balich, Purdue University Calumet

ALAN BALICH received his B.S. in Electrical and Computer Engineering Technology from Purdue University, Calumet in 2007. His current interests reside in remotely operated vehicles (air, water, and ground based), robotics, and embedded systems (specifically, microcontrollers programmed using BASIC, C, and Assembly languages).

INTERNET CONTROLLED UNDERWATER VEHICLE

Abstract

The paper provides an overview of design, development, and testing of the Internet Controlled Underwater Vehicle. As a senior design project it provides the students an integrating experience of the knowledge and skills that have been acquired in their pursuit of a Bachelor of Science Degree in Electrical and Computer Engineering Technology at Purdue University, Calumet.

The paper examines in detail the previous research and development schemes that were used in creating the structure(s), housing of the electronics and propulsion systems of typical remotely operated vehicles. The paper focuses on the advantages and benefits achieved in the current design of the Internet Controlled Underwater Vehicle.

The paper elaborates on the electronics used in the control and communication between the end user and the vehicle. Furthermore, details of the of the propulsion system, control system, and the necessary communication protocols are furnished.

I. Introduction

The Internet Controlled Underwater Vehicle (ICUV) is both a mechanical and electrical device that can be used to explore the undersea environment without having to physically be at the location. It is a vertically oriented, neutrally buoyant vehicle capable of moving in the following directions: forward, backward, right, left, up and down. The ICUV uses a web-enabled camera, web-server, and a microcontroller to allow any user with an available Internet connection to manipulate the position and depth of the vehicle. A graphical user interface is provided by means of the web-server so as to require a small learning curve when it comes to controlling the ICUV. To the users, it would appear as if they were playing a video game, but the ICUV is a physical device executing the user's commands in real-time.

Not only can the users manipulate the physical attributes of the ICUV, but they can also choose to receive data from the vehicle. This data represents physical conditions that surround the ICUV. The ICUV is currently out-fitted with a digital compass and a temperature sensor. The digital compass provides the current heading of the ICUV, where as the temperature sensor reports the current temperature of the water.

The final form of data transmitted from the ICUV is that of a live video feed[8] from a web-enabled camera. The camera has the ability to pan and tilt so as to allow the user to pan the lenses 100° and tilt the lenses 45° from the center point.

II. System Description

Mechanical and Physical Design

The mechanical and physical design of the ICUV is unique because of the vertical orientation as its floats in water. Most remotely operated underwater vehicles are horizontally oriented, cylindrical tubes with a propeller protruding from the back. The ICUV is vertically oriented with two plastic spheres located at both the top and bottom of the body. The top sphere contains the electronics for controlling the propulsion system and reading both sensors, while the bottom sphere contains the web-enabled camera. The propulsion system is located in-between both spheres.

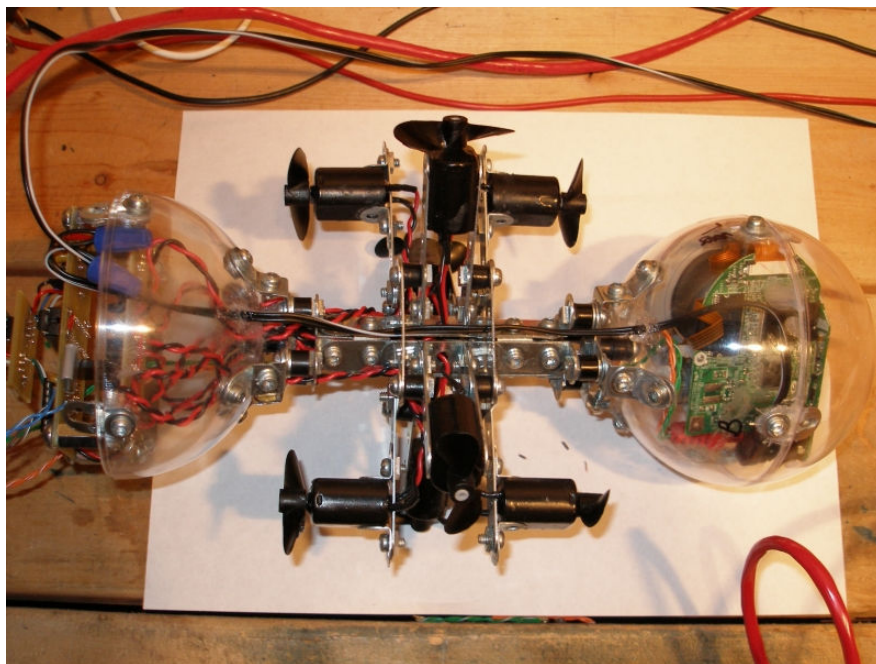A picture of the completed Internet Controlled Underwater Vehicle is shown in Figure 1 below.

Figure 1: Completed ICUV

Vertical Orientation vs. Horizontal Orientation

Vertical orientation was chosen over horizontal orientation because it offered better maneuverability in tight spaces. For example, the investigation of a ship wreck does not necessarily mean that there will be a large enough hole for a typical underwater vehicle to fit through. Since the ICUV is vertically oriented, the user could easily enter the ship wreck through a port-hole window or other small opening that may prevent larger remotely operated vehicles from entering.

Vertical orientation allowed the ICUV to be more easily balanced when it came to making it neutrally buoyant. The top electronics sphere is less massive than the camera module, thus, with the camera in the bottom sphere, the ICUV maintains a vertical orientation. One gram masses were used to make final adjustments to the ICUV's neutral buoyancy

Eight Thruster Propulsion System

The ICUV contains an eight thruster propulsion system in order to propel itself in one of six directions. An eight thruster propulsion system allows for better accuracy and maneuverability in tight spaces where there is little room to travel. With a separate thruster controlling each direction of travel, a turning radius no longer exists in terms of movement. Other underwater vehicles may rely on a rudder control system to serve as their means for direction control. The problem occurs in that the vehicle still has to propel itself forward in order to move water across the rudders so the vehicle can turn. In a space restricted area, there may not be enough room for the vehicle to move enough water across the rudders for them to effectively turn the underwater vehicle. The eight thruster propulsion system I have built effectively eliminates this problem. All directions are controlled by their own thruster that moves water propelling the ICUV in the user determined direction.

Depth control of the ICUV is achieved by using four thrusters from the propulsion system. In this case, the four depth control thrusters are separated into two groups of two thrusters per group. Since the vehicle is neutrally buoyant, a small application of force will easily move the ICUV. Therefore the pair of thrusters facing towards the surface of the water will push the vehicle down, or decrease the overall depth of the ICUV. The opposite is true for the second pair of depth control thrusters, they are facing down towards the sea bed, thus, pushing the ICUV up towards the surface of the water when activated.

Electronics System Design

The electronics system in the ICUV is responsible for receiving commands from the user and the interpreting those commands so as to cause the ICUV to perform the action requested by the user. It is also the responsible for reading the sensors, then sending the sensor data back to the user for human interpretation. Finally, the electronics system also provides the proper interfacing requirements to allow the low voltage/current microcontroller to control the higher voltage/current thrusters.

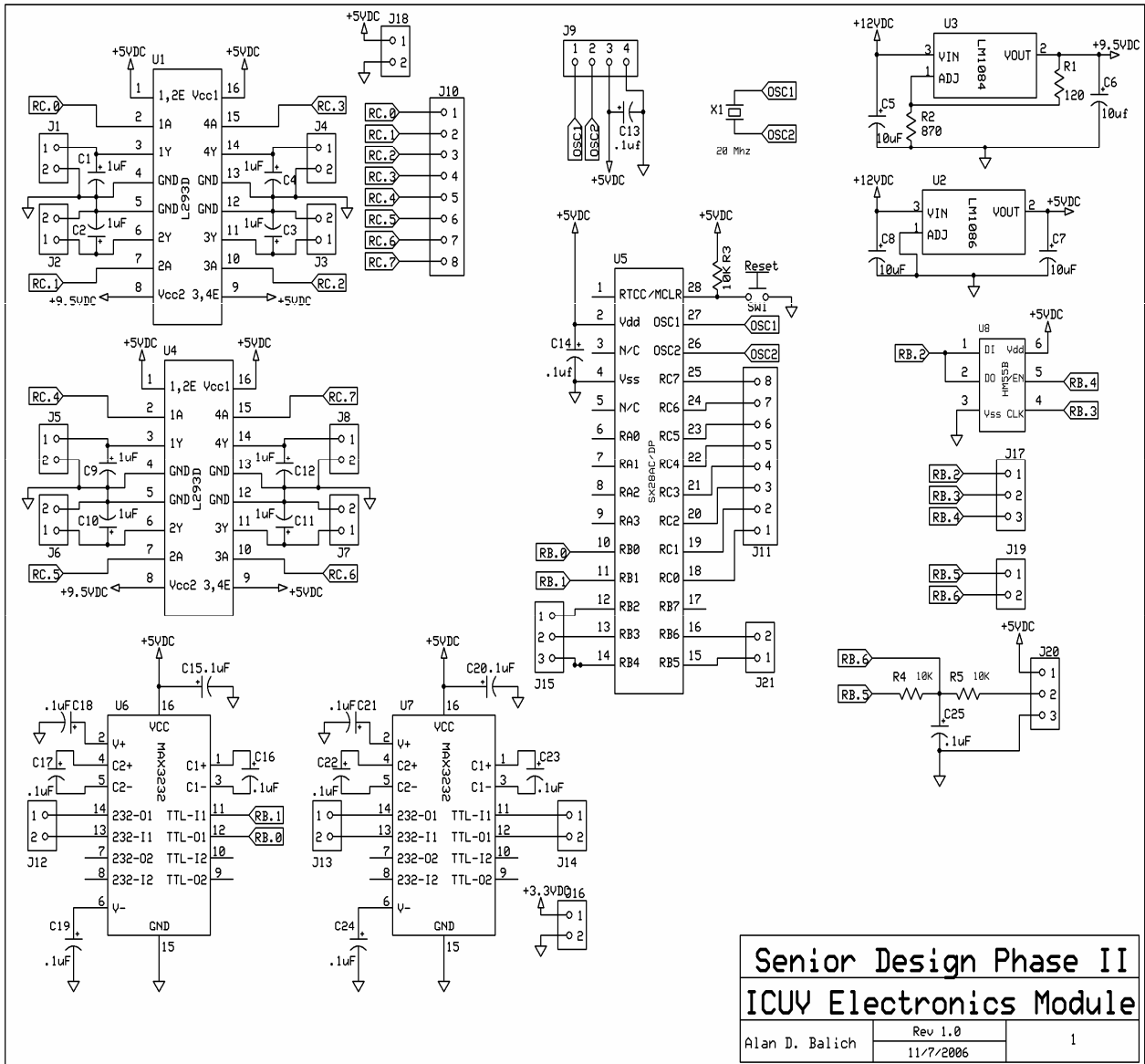A schematic of the ICUV's electronics module is shown in Figure 2.

Figure 2: ICUV Electronics Module Schematic

Thruster Interface System

The thruster interface system consists of a LM1084 voltage regulator and two L293DNE[1] high voltage/current drivers. The LM1084 is an adjustable voltage regulator that can provide a maximum regulated voltage of 15 VDC at a current of 5 ADC. Each thruster in the propulsion system requires approximately 9 VDC at 200 mA DC to operate, therefore, the voltage regulator has been setup to output 9.5 VDC from a 12 VDC input (12 VDC, 5 AH Lead Acid Battery).

The L293DNE high voltage/current driver allows the low voltage/current SX28 microcontroller to control each of the eight individual thrusters. The drivers have fly-back diodes integrated directly into the integrated circuit chip, thus, external diodes are not required. The fly-back diodes prevent any noise or EMF generated by motors from interfering with the microcontroller.

Each L293DNE chip contains four driver channels and each channel is capable of supplying up to 36 VDC at 600 mA. In the case of the ICUV's thrusters, the L293DNE will only have to control 9.5 VDC at 200 mA which is well within the maximum limits of the chip.

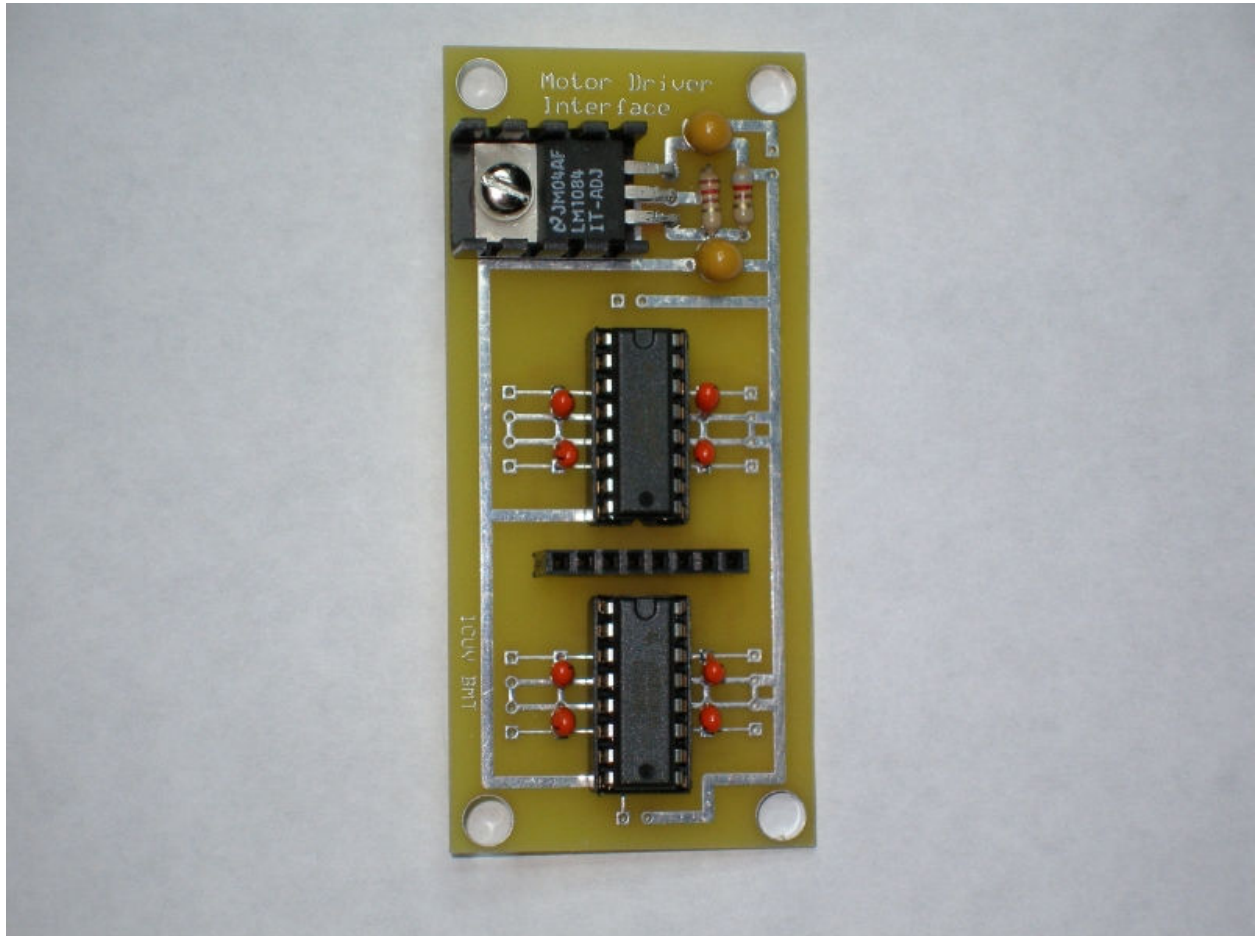A picture of the completed Thruster Interface System is shown in Figure 3 below.

Figure 3: Completed Thruster Interface System

Parallax SX28 Microcontroller

The Parallax SX28[2] microcontroller is the heart of the electronics system and the ICUV itself. It is responsible for providing all of the necessary processing power in terms of interpreting commands, controlling the motors, and communicating with the sensors and web-server. The following communication protocols and devices have been implemented in the firmware:

The interface between the web-server and the SX28 microcontroller is handled by a single asynchronous serial port which consists of a transmission line and a receiving line. Since the SX28 does not have a built in UART, the serial port is implemented in the firmware that is loaded into the chip's flash storage section.

Not only is the asynchronous serial protocol being used, but a synchronous serial protocol is being used to communicate with the digital compass.

The SX28 does not have an onboard, hardware analog-to-digital converter (ADC), therefore, one is implemented in the firmware. A Sigma-Delta ADC is used to convert the analog voltage generated by the LM34DZ temperature to a digital value that closely represents the analog output from the temperature sensor.

The SX28 microcontroller is capable of handling all of the tasks because of its high-speed operation. It has a top speed of 75 MHz, but in the case of the ICUV, it is running at 20 MHz for that this speed allows the microcontroller to handle communications between the web-server, sensors, and controlling of the propulsion system. One of the important factors is the 1-to-1 execution cycle of the assembly instructions. This means that most of the assembly instructions will take 1 cycle of CPU time to execute. Therefore, at 20 MHz, 1 cycle of CPU time is equal to 50 nanoseconds per instruction.

Given that several devices are being interfaced with the SX28, the amount of available input/output pins makes the SX28 versatile. It has 21 general purpose I/O pins available which are divided into the following three groups:

Port RA contains 5 I/O pins
Port RB contains 8 I/O pins
Port RC contains the final 8 I/O pins.

In the ICUV, Port RA is not used, but Port RB and RC are used in the following ways:

Port RB is used as the sensor interface and serial port. The HM55B digital compass requires 3 I/O lines (Clock, I/O, Chip Select) while the Sigma-Delta ADC requires 2 I/O lines (Capacitor Feedback and Capacitor Charge Time Counter). The transmitting and receiving lines for the asynchronous serial port are also implemented by using another 2 of Port RB's I/O lines.

All 8 I/O lines of Port RC are used to control the eight individual thrusters of the propulsion system. Port RC is interfaced to the thrusters by means of the L293DNE driver which isolated and protects Port RC from voltage spikes or over-current conditions.

A picture of the completed SX28 Control Interface is shown in Figure 4.
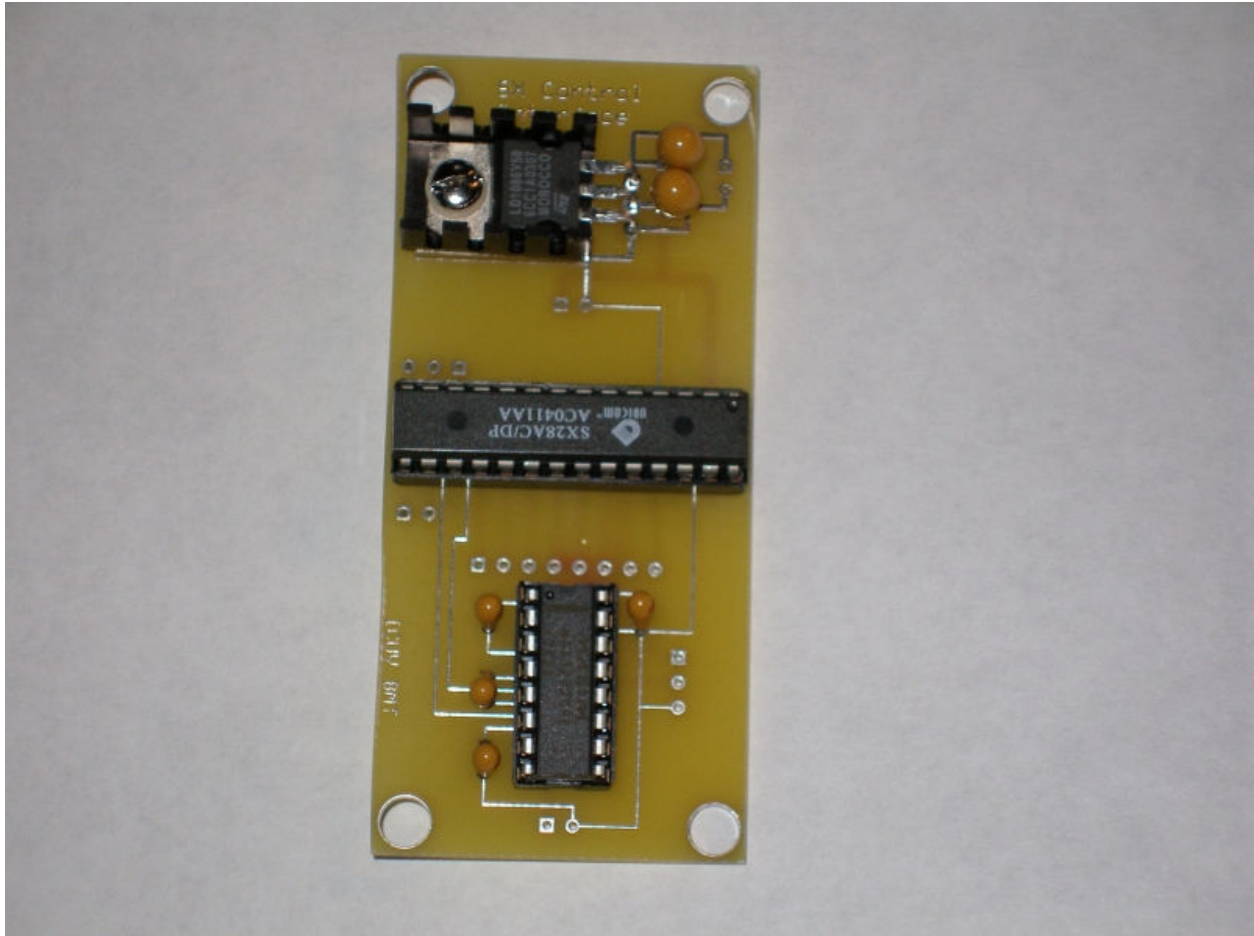
Figure 4: Completed SX28 Interface System

Maxim MAX3232 RS-232/TTL Transceiver

The RS-232 specification is used to implement the asynchronous serial protocol between the SX28 microcontroller and the web-server. Since both the SX28 and the web-server are TTL level devices, the length of the cable in which the serial signals will travel over is limited in length. By using the RS-232 specification, the maximum length of the cable can be increased to approximately 45 feet at a speed of 9600 bps. Hence, using the MAX3232[3] transceiver chips allows the ICUV to have a maximum cable length of 45 feet between the underwater vehicle and the web-server.

The MAX3233 can be powered by a 3.3 VDC or 5 VDC logic supply and requires only four external 0.1 uF capacitors. It uses a series of charge pumps to generate the +12 and -12 VDC RS-232 voltages which are temporarily stored in the four external capacitors.

Hitachi HM55B Digital Compass

The HM55B[4] digital compass is used to determine the current heading of the ICUV in the following terms: a compass rose direction (North, East, South, and West) and a degree (ex. heading 350° North by North-West).

In terms of resolution, the HM55B digital compass can differentiate between 64 different headings. Therefore, this comes out to about 5° per heading change when it is rotated about its z-axis towards a compass specific direction, such as North.

LM34DZ Temperature Sensor & Sigma-Delta Converter

The LM34DZ[5] temperature sensor is used to determine the current water temperature the ICUV is currently encountering in its environment. Its output is calibrated in terms of a change in 1° Fahrenheit causes a 10 mVDC change in the output of the sensor. Therefore, applying what was stated above means that a temperature of 72° F will mean an analog output of approximately 720 mVDC.

In order to allow the SX28 microcontroller to transmit the current temperature sensor reading back to the user, the analog output from the temperature sensor must be converted to a digital value. Since the SX28, does not have an onboard, hardware ADC, a Sigma-Delta ADC has been created in firmware. The Sigma-Delta ADC requires two 10 k-ohm ohm resistors and a single 0.001 uF capacitor in for the SX28 to properly convert the analog voltage to its digital representation.

Parallax Embedded Web-Server

The Parallax Embedded Web-Server[6] serves as the host for the graphical user interface that the user will interact through their web-browser. Effectively, a web-page is created using the Hyper-Text Markup Language (HTML) and is then transferred to the web-server by means of the File Transfer Protocol (FTP). All the user has to do is navigate to that web-page by means of a Uniform Resource Locator (URL) and allow the web-browser to render the page properly. The Parallax Embedded Web-Server and the features that are being used to communicate with the ICUV are described below.

The web-server contains a Motorola Coldfire processor that is interfaced with an SDRAM (64 kB) and a Flash Ram (512 kB). The flash ram is where the web-page is stored while all variables are located in the SDRAM. These variables are then referenced in HTML so as to allow one of the following two things to happen: data received over the serial can be stored in one of the variables and then later displayed on the web-page, or a user could input data into a text box on the web-page which is then sent out over the serial link to the receiving device.

In the case of the ICUV, all sensor data is continuously transmitted over the serial link to two separate variables and displayed on the webpage. As for the user's control over the travel of the ICUV, the user clicks a button on the web-page that corresponds to the firing of a particular thruster. This is not automatically transmitted over the serial link to the SX 28 microcontroller,

but it is the job of the SX28 to poll the web-server and see if that specific variable has been updated by a user event.

III. Firmware Operation

The native language of the SX28 microcontroller is that of SX assembly (SASM). Given today's modern PC technology, a program called a compiler can be created in order to take a higher-level language and "compile" into a lower-level language that the microcontroller executes.

In the case of the ICUV project, a higher-level language (Parallax BASIC[7]) was used in order to expedite the process of attaining the necessary functionality for the ICUV. The BASIC high-level language is closer to actual English language, thus, the same meanings in the spoken English language are inherent in the BASIC language. This allows the firmware in the ICUV's microcontroller to be more easily understood by a variety of people as it is further developed and expanded.

The code snippets shown below are described in order to convey the main function of the ICUV. Its main function is to receive a command from the user and act upon the command in the form of travel in a user specified direction.

Initialization Section

```
Initialize:
PAUSE 200
TRIS_C  =  %00000000
RC      =  %00000000
En      =  1
Clk     =  0
ThrDir  =  0
AdcRaw  =  0
StrPntr =  0
```

In the initialization code, the microcontroller pauses its execution for 200 milliseconds in order to allow the web-server to complete its boot-up procedure. The next seven lines of code initialize the values of the listed variables to a predetermined state. In this case, all variables, except "En" are initialized to a value of zero. The "En" variable is initialized to one because this value disables the digital compass.

Main Program Execution Section

```
DO
ChkPwsThr
ChkThr
GetTemp
FormatData AdcCal
PwsTempVar1
```

LOOP

In the main program code, the microcontroller enters an infinite loop and calls all the subroutines listed after the DO statement. In the fourth subroutine FormatData AdcCal, the value currently stored in the AdcCal variable is being passed to the subroutine. The same is also true for those subroutine calls in which the variable is preceded by the subroutine name and a character space. When the microcontroller encounters the LOOP statement it simply goes back to the address of the DO statement and executes all the subroutine calls again.

Reading Variable 00 From The Parallax Webserver

```
ChkPwsThr:
StrPntr = 0
DO
LOOKUP StrPntr, "!", "N", "B", "0", "R", "0", "0", 0, Work
IF Work = 0 THEN EXIT
SendData Work
INC StrPntr
LOOP
RecieveData
ThrDir = SerRxBuffer
RETURN
```

The Parallax Webserver contains 100 variables that are capable of holding 64 bytes per variable. In this case, variable 00 on the webserver will be used to hold a single alphabetic character that will be sent to the microcontroller in order to turn on the proper thruster(s). This variable is written when the user clicks a button on the webpage interface that writes a character corresponding to a direction of travel.

In the code snippet above, the first line is the label of the subroutine that is called by the main program execution loop. The second line initializes the StrPntr variable to zero since the string of characters that is sent to the webserver begins at address zero. With the StrPntr variable initialized to zero, the program enters a   DO…LOOP that performs the following actions in order:

1) Looks up the current character stored at the address pointed to by the variable StrPntr.
2) Stores the character pointed to by StrPntr in the Work variable.
3) Check the Work variable for a zero value. If Work equals zero then exit out of the DO…LOOP, otherwise send the character in the work variable to the serial port for transmission to the webserver by calling the SendData subroutine
4) Increment the StrPntr variable by one.
5) If the DO…LOOP is exited then call the ReceiveData subroutine and wait to receive data to the serial port from the webserver.
6) Once data has been received from the webserver, store the data in the variable SerRxBuffer

7) Return to the next instruction in the main program loop after the previous subroutine call.

Determining Which Thruster To Activate

```
ChkThr:
IF ThrDir = "F" THEN
ActvThr Forward
ENDIF
IF ThrDir = "B" THEN
ActvThr Backward
ENDIF
IF ThrDir = "R" THEN
ActvThr Right
ENDIF
IF ThrDir = "L" THEN
ActvThr Left
ENDIF
IF ThrDir = "U" THEN
ActvThr Up
ENDIF
IF ThrDir = "D" THEN
ActvThr Down
ENDIF
IF ThrDir = "O" THEN
ActvThr Off
ENDIF
RETURN
```

Once the command has been received from the webserver, the ChkThr subroutine is called. The ChkThr subroutine tests the value that is stored in the ThrDir variable. Remember, the value stored in the ThrDir variable is the value that was received from the webserver which was chosen by the user. The ThrDir variable is then tested against a series of IF…THEN statements. If the ThrDir meets the equality condition set forth by the IF…THEN statement, then the proper pin is set to the on state on port RC. Port RC is directly connected to the motor driver interfaces and those interfaces drive the thrusters.

A complete listing of the SX28 source code is included in Appendix A.

IV. Summary and Future Work

The thrusters can not reverse their direction of rotation, thus, some advanced maneuverability is sacrificed. For example, by being able to reverse the direction of rotation of the thrusters would allow the ICUV to be offset from its 90° vertical orientation. The ability to offset the angle of vertical orientation would allow further maneuverability of the ICUV in the event of the need to fit in tight spaces.

The determination of the current heading of the ICUV is based upon the angular offset of the digital compass from magnetic North. Since, the ICUV's propulsion system does not require the entire vehicle to change its angle of travel, the digital compass serves a purpose in aiding the determination of initial orientation. Afterwards, the different directions of travel the ICUV may take do not require a change in the angular position of the vehicle. This is due to the four thrusters that are pushing the ICUV in the user controlled direction.

V. Pedagogy

The ICUV project was accomplished by an individual student in order to meet the necessary requirements for the following capstone courses: Senior Design, Phase I and Senior Design, Phase II. Both of theses courses are required for the Bachelor of Science Degree in the Electrical and Computer Engineering Curriculum and are taken during the seventh and eighth semesters. Thus, the project and all the work involved with the project, represent the summation of the student's application of the skills and knowledge acquired over the past four years.

In this project, the student was expected to rely on the skills and knowledge he has acquired along with the research that is relevant to his project. The role of the project advisor is to monitor the student's progress in the creation of the project and evaluate how the student has used his knowledge and skills in the overall project. Given that the advisor provides little aid in helping the student select, research and create his project, thus, the student learns and practices the art of self-directed learning. In all, the Senior Design capstone course is the idea behind motivating the student to continue his education and learning processes over his entire lifetime.

Appendix A.
Parallax SX28 BASIC Source Code For ICUV

```
'!ICUV!ICUV!ICUV!ICUV!ICUV!ICUV!ICUV!ICUV!ICUV!ICUV!ICUV!ICUV!ICUV!ICU
'                                                                  !ICUV
'  File...... ICUV_CMD_Interpreter.SXB                             !ICUV
'  Purpose... To recieve commands from the end user interface and  !ICUV
'             interpret those commands to perform the action       !ICUV
'              specified by the end user.  Also, to transmit data back !ICUV
'             to the end user interface (Water Temperature/Heading) !ICUV
'  Author.... Alan Balich                                          !ICUV
'  E-mail.... alinious@gmail.com                                   !ICUV
'  Started... October  01,  2006                                   !ICUV
'  Updated... December 03,  2006                                   !ICUV
'  Version... 2.0                                                  !ICUV
'!ICUV!ICUV!ICUV!ICUV!ICUV!ICUV!ICUV!ICUV!ICUV!ICUV!ICUV!ICUV!ICUV!ICU


DEVICE SX28, OSCXT2, TURBO, STACKX, OPTIONX
IRC_CAL IRC_FAST
FREQ 20_000_000


'PIN DEFINITIONS------------------------------------------------------
ASI       PIN   RB.0 INPUT                'Serial Input Pin
ASO       PIN   RB.1 OUTPUT               'Serial Output Pin
DinDout   PIN   RB.2 INPUT                RB.2 transceives to/from Din/Dout
Clk       PIN   RB.3 OUTPUT               'RB.3 sends pulses to HM55B's Clk
En        PIN   RB.4 OUTPUT               'RB.4 controls HM55B's /EN(ABLE)
AdcFb     PIN   RB.5 OUTPUT          'RB.5 ADC Feedback
AdcIn     PIN   RB.6 INPUT CMOS          'RB.6 ADC Input
ThrCntrl  VAR  RC                         'Thruster Control Port
'END PIN DEFINITIONS--------------------------------------------------


'SERIAL PORT SETTINGS-------------------------------------------------
BaudRate  CON               "T2400"       'Baud rate/logic definition for asynchronous serial
routines
                                          'Transmit/Recieve using true logic @ 9600bps
'END SERIAL PORT SETTINGS---------------------------------------------


'PROPULSION SYSTEM DEFINTIONS-----------------------------------------
Up        CON     %10100000
Down      CON     %01010000
Left      CON     %00000001
Right     CON     %00000100
Forward   CON     %00001000
Backward  CON     %00000010
Off       CON     %00000000
'END PROPULSION SYSTEM DEFINTIONS-------------------------------------


'COMPASS DEFINITIONS & SETTINGS--------------------------------------
YOffset   CON     0               ' Enter measured y at north here
```

```
XOffset    CON         0                   ' Enter measured x at west here


ResetHM    CON              %0000                   ' Reset command for HM55B
Measure    CON              %1000                   ' Start measurement command
Report     CON              %1100                   ' Get status/axis values command
Ready      CON              %1100                   ' 11 -> Done, 00 -> no errors
NegMask    CON              %1111100000000000       ' For 11-bit negative to 16-bits
'END COMPASS DEFINITIONS & SETTINGS--------------------------------------------


'SERIAL PORT VARIABLES-----------------------------------------------------------
SerTxBuffer VAR             BYTE            'Buffer to hold data to transmit to user
SerRxBuffer VAR             BYTE            'Buffer to hold commands recieved from user
'END SERIAL PORT VARIABLES-------------------------------------------------


'PROPULSION SYSTEM VARIABLES------------------------------------------------------
ThrDir     VAR         BYTE                'Variable to hold specific thruster to fire
'END PROPULSION SYSTEM VARIABLES--------------------------------------------------


'COMPASS VARIABLES-----------------------------------------------------------
X          VAR     BYTE(2)             ' y-axis data
Y          VAR     BYTE(2)             ' x-axis data
StatusFlags VAR     BYTE                ' HM55B Status flags
'END COMPASS VARIABLES--------------------------------------------------------


'ANALOG TO DIGITAL CONVERTER VARIABLES-------------------------------------------
AdcRaw     VAR     BYTE
AdcCal     VAR     BYTE
'END ANALOG TO DIGITAL CONVERTER VARIABLES----------------------------------------


'SERIAL DATA FORMATING (ASCII) VARIABLES-----------------------------------------
Char3    VAR         BYTE
Char2    VAR         BYTE
Char1    VAR         BYTE
Dig3     VAR         BYTE
Dig2     VAR         BYTE
Dig1     VAR         BYTE
Work     VAR         BYTE
Temp     VAR         BYTE
'END SERIAL DATA FORMATING (ASCII) VARIABLES-------------------------------------


'PARALLAX WEB-SERVER (PINK) VARIABLES--------------------------------------------
StrPntr    VAR         BYTE
'END PARALLAX WEB-SERVER (PINK) VARIABLES----------------------------------------


PROGRAM Initialize


'SUBROUTINE DECLARARTIONS--------------------------------------------------------
ChkPwsThr      SUB  0
ChkThr         SUB  0
ActvThr        SUB  1
GetTemp        SUB  0
```

```
FormatData      SUB   1
PwsTempVar1     SUB   0
GetHeading      SUB   0
PwsCompVar2     SUB   0
SendData        SUB   1
RecieveData     SUB   0
'END SUBROUTINE DECLARARTIONS--------------------------------------------------

Initialize:
        PAUSE 200                       'Wait for Parallax Web-Server to startup
        TRIS_C  = %00000000
        RC      = %00000000
        En      = 1                     ' Disable HM55B
        Clk     = 0                     ' Start with clock line output-low
        ThrDir  = 0
        AdcRaw  = 0
        StrPntr = 0

DO
        ChkPwsThr
        ChkThr
        GetTemp
        FormatData AdcCal
        PwsTempVar1
        GetHeading
        FormatData X(0)
        PwsCompVar2
        FormatData Y(0)
        PwsCompVar2
'       FormatData X(1)
'       FormatData Y(1)
LOOP

ChkPwsThr:
        StrPntr = 0
        DO
                LOOKUP StrPntr, "!", "N", "B", "0", "R", "0", "0", 0, Work
                IF Work = 0 THEN EXIT
                SendData Work
                INC StrPntr
        LOOP
        RecieveData
        ThrDir = SerRxBuffer
RETURN

ChkThr:
        IF ThrDir = "F" THEN
                ActvThr Forward
        ENDIF
        IF ThrDir = "B" THEN
                ActvThr Backward
```

```
                ENDIF
                IF ThrDir = "R" THEN
                        ActvThr Right
                ENDIF
                IF ThrDir = "L" THEN
                        ActvThr Left
                ENDIF
                IF ThrDir = "U" THEN
                        ActvThr Up
                ENDIF
                IF ThrDir = "D" THEN
                        ActvThr Down
                ENDIF
                IF ThrDir = "O" THEN
                        ActvThr Off
                ENDIF
RETURN


ActvThr:
        ThrCntrl = __PARAM1
RETURN


GetTemp:
        ANALOGIN AdcIn, AdcFb, AdcRaw, 4
        AdcCal = AdcRaw - 29
RETURN


GetHeading:
        En = 1                          ' Reset HM55B
        En = 0

        SHIFTOUT DinDout, Clk, MSBFIRST, ResetHM\4

        En = 1
        En = 0                          ' Start measurement

        SHIFTOUT DinDout, Clk, MSBFIRST, Measure\4

        StatusFlags = 0                 ' Clear previous status flags

        DO                              ' Repeat until measurement ready
                En = 1                          ' Request measurement status
                En = 0
                SHIFTOUT DinDout, Clk, MSBFIRST, Report\4
                SHIFTIN  DinDout, Clk, MSBPOST,  StatusFlags\4 ' Get measurement status
        LOOP UNTIL StatusFlags = Ready

        SHIFTIN  DinDout, Clk, MSBPOST, X(1) \3     ' Get 11 signed x-axis bits
        SHIFTIN  DinDout, Clk, MSBPOST, X(0) \8     ' Get 11 signed x-axis bits

        SHIFTIN  DinDout, Clk, MSBPOST, Y(1) \3     ' Get 11 signed y-axis bits
```

```
        SHIFTIN  DinDout, Clk, MSBPOST, Y(0) \8      ' Get 11 signed x-axis bits

        En = 1                              ' Disable HM55B
RETURN

FormatData:
        Temp = __PARAM1

        Dig3 = Temp / 100
        Work = Dig3 + 48
        Char3 = Work

        Dig2 = Temp / 10
        Work = Dig3 * 10
        Dig2 = Dig2 - Work
        Work = dig2 + 48
        Char2 = Work

        Work = Dig3 * 100
        Dig1 = Temp - Work
        Work = Dig2 * 10
        Dig1 = Dig1 - Work
        Work = dig1 + 48
        Char1 = Work
RETURN

PwsTempVar1:
        StrPntr = 0
        DO
                LOOKUP StrPntr, "!", "N", "B", "0", "W", "0", "1", ":", Char3, Char2, Char1, 0, Work
                SendData Work
                IF Work = 0 THEN EXIT
                INC StrPntr
        LOOP
RETURN

PwsCompVar2:
        StrPntr = 0
        DO
                LOOKUP StrPntr, "!", "N", "B", "0", "W", "0", "2", ":", Char3, Char2, Char1, " ", 0,
Work
                SendData Work
                IF Work = 0 THEN EXIT
                INC StrPntr
        LOOP
RETURN

SendData:
        SEROUT ASO, BaudRate, Work
RETURN
```

```
RecieveData:
        SERIN ASI, BaudRate, SerRxBuffer, 200, NoChar
NoChar:
RETURN
```

References

[1] Texas Instruments L293DNE High Current/Voltage Driver Datasheet
http://focus.ti.com/lit/ds/symlink/l293d.pdf

[2] Parallax, Inc. SX28 Microcontroller Datasheet
http://www.parallax.com/dl/docs/prod/datast/SX20AC-SX28AC-Data-v1.6.pdf

[3] Maxim RS-232 Asynchronous Transceiver Datasheet
http://datasheets.maxim-ic.com/en/ds/MAX3222-MAX3241.pdf

[4] Hitachi HM55B Digital Compass Datasheet
http://www.parallax.com/dl/docs/prod/compshop/HM55BModDocs.pdf

[5] National Semiconductor LM34DZ Fahrenheit Temperature Sensor Datasheet
http://www.national.com/ds.cgi/LM/LM34.pdf

[6] Parallax Embedded Web-Server Datasheet
http://www.parallax.com/dl/docs/prod/comm/30013-PINK-v1.02.pdf

[7] Parallax BASIC Compiler (SX-Key IDE v 3.2.3)
The IDE contains the reference manual for the Parallax Basic lexicon.
http://www.parallax.com/sx/sxb.asp

[8] Panasonic Wired Network Camera BL-C10A Manual
http://service.us.panasonic.com-BLC10A.pdf