Zuni/Tam Web Based Monitoring System

Summery:

The Web Based Monitoring System will provide us the ability to monitor and control the following ship board systems on the former USS Zuni/USGCC Tamaroa over the web while no one is on board for the following.

1. Monitor the 9 float switches connected to the 120VAC Bilge Alarm Panel for flooding conditions and be able to silence outside alarm.
2. Monitor the two 3 Phase Circuit Breakers in the Switchboard for power outages.
3. Monitor 4 additional Bilge Float Switches when the power fails to the 120 VAC Bilge Alarm Panel.
4. Monitor the Addressable Fire Alarm Panel for alarm and trouble conditions and silence the horns on the fire alarm panel.
5. Monitor and control the Solar Charging System.

The idea for this design started years ago, with the idea of monitoring our ship through text messages. But, the problem was that most systems out there didn't offer what we were looking for. Those systems ether didn't have the I/O we needed or it was pay by each text message that was sent, which would be a costly problem when false alarms were trigger.

So, when I joined the Parallax Forums I noticed a text message GSM modem based monitoring system done by forum member Kevin Brown and this give me the idea I was looking for. So, this pointed me to use a GSM modem and have it connect to the Spinneret and I can have all the I/O I needed to meet our needs. This mint I needed to make an I/O board that would work with the Spinneret and this pointed to me using the Propeller chip for this board, which it would communicate to the spinneret.

But after talking to forum members like Mike Green, my idea of using the GSM modem may not work like I was thinking it would. So, I looked at another GSM modem that had Ethernet that was on the board. But, I needed something work without a lot of programming changes from the original. So, I had to make a decision on how I am going to communicate with the Spinneret. So, I posted a question on the Parallax forums and Mike G. replied with the answer I was looking for and this was to use a CBA250 4G/3G network router (made by Cradle Point) that works with a standard USB modem and that pointed me to another product made by the same company and it had IP pass through which works even better. This way it would provide another benefit and that was Internet access for a space that has poor cell phone signal.

System Components:

This system includes the following

1. Cradle Point CBA250 4G/3G network router which provides the connection for the USB modem which communicates with the Spinneret using the RJ45 jack and provides future internet connection using a 5 port Ethernet switch.
2. Parallax Spinneret Web Server for storing the web page content and communicating with the I/O board (Propeller I/O Board) using the J6 connection which is used for bi-directional serial communication.
3. I/O board (Propeller I/O Board) which monitors the inputs and controls the outputs, which is programmed using spin, which communicates alarm and status values to the Spinneret using bi-directional serial. This can be expended in the future using RS-485 communications.

System Operation:
The monitoring system will operate as follows:

120VAC Bilge Alarm Inputs
If any of the 9 Bilge Alarm inputs are triggered for more than 3 seconds, it will do the following.

1. Turn on the Alarm Light that is connected at output 19 on the I/O board (Propeller Pin 22) and turn on the RC-4 (K1) via the serial port on the I/O board (Propeller Pin 26).

2. Set a alarm value for that input in variable "txalarmval" and send that value via the serial port on the I/O board (Propeller Pin 27) to the Spinneret (J6 connection) to be displayed on the web page after all bilge inputs are scanned by the Propeller.

12VDC Bilge Alarm Inputs
When the shore power is off, the generators aren't running, or the power breaker is off, or the bilge panel power is turned off. This will cause the "acbilgeloss" relay to be in a de-energize state and cause the 12V battery to provide backup power for the following bilge alarm inputs on the I/O board and will lockout the any other bilge alarm inputs.

1. Input 3 (Propeller Pin 6) "Bilge2"
2. Input 5 (Propeller Pin 8) "Bilge4"
3. Input 9 (Propeller Pin 12) "Bilge8"
4. Input 10 (Propeller Pin 13) "Bilge9"

When any of the above inputs are triggered for more then 3 seconds and the "acbilgeloss" is still in the de-energize state it will cause the Alarm Light that is connected at output 19 on the I/O board to turn on (Propeller Pin 22) and turn on the RC-4 (K1) via the serial port on the I/O board (Propeller Pin 26) and will set a alarm value in "txalarmval" for the following inputs

1. Input 3 "Bilge2" value for "txalarmval" is 10
2. Input 5 "Bilge4" value for "txalarmval" is 11
3. Input 9 "Bilge8" value for "txalarmval" is 12
4. Input 10 "Bilge9" value for "txalarmval" is 13

When any of the bilge alarm inputs are triggered and when another bilge alarm is triggered it will take priority over the first input that was triggered. This means the value in the variable "txalarmval" is over written and the new value is displayed on the webpage. This will apply for both modes of operation.

When the Alarm Light is on or the web page is looked at by the staff members of our group. They can be silence in two ways in order to silence the outside horn and turn off the Alarm Light.

1. Push the Silence pushbutton that is connected at input 16 on the I/O board (Propeller pin 19), which will turn off the horn and Alarm Light.
2. Push the Silence pushbutton that is on the web page, which sets a "b" in variable "rxsilence", which will turn off the horn and Alarm Light.

When the bilge alarm inputs (float switches) reset (return to normal) this will cause a value of 0 to set in the variable txalarmval"

Fire Alarm System

If the Fire Alarm panel is in alarm, the alarm relay in the panel will trigger input 11 on the I/O board (Propeller pin 14), which will then set a alarm value of 14 for this input in the variable "txalarmval" and send that value via the serial port on the I/O board (Propeller Pin 27) to the Spinneret (J6 connection) to be displayed on the web page after all alarm inputs are scanned by the Propeller.

When the Fire Alarm panel is in alarm, the panel itself stays in alarm until an on-call staff member from our group is dispatched to the ship. The alarm value of 14 in "alarmval" will continue to be sent to the Spinneret to keep the web page updated. While this is all happening, the outside Horn/Strobe can be silenced in two ways until the Fire Alarm panel is reset by our on-call staff member from our group.

1. Push the Silence pushbutton that is connected at input 16 on the I/O board (Propeller pin 19), which will turn on the Horn Silence relay at is connected the I/O board output 21 (Propeller pin 24).
2. Push the Silence pushbutton that is on web page, which sets a "f" in variable "rxsilence"

When the Fire Alarm panel is in trouble (not alarm) indicating that there is problem with the Fire Alarm panel this will cause the trouble relay in the panel to trigger the I/O board input 12 (Propeller pin 15). This will set a value of 1 in the variable "txstatval" and send that value via the serial port on the I/O board (Propeller Pin 27) to the Spinneret (J6 connection) to be displayed on the web page after all alarm inputs are scanned by the Propeller. This value will continue to be sent to the Spinneret to keep the web page updated until the trouble or troubles on the Fire Alarm panel is addressed. The only other thing, other than bad smoke detectors that can cause the Fire Alarm to be in trouble is the Horn silence relay being turned on due do a alarm on the Fire Alarm panel or the relay removed form its socket.

Power Monitoring

When shore power is loss or the shore power breaker is turned off, or the circuit breaker to the 120VAC bilge alarm panel is turned off or tripped, this will cause the "acbilgeloss" relay to be in a de-energize state, causing this relay to triggered the I/O board input 1 (Propeller pin 4). This will set a value of 2 in the variable "txstatval" and send that value via the serial port on the I/O board (Propeller Pin 27) to the Spinneret (J6 connection) to be displayed on the web page after all alarm inputs are scanned by the Propeller.

When the Port Shore power breaker is turned off or power is lost (breaker being tripped or utility is lost) this will cause the Port Shore power breaker relay in the main switchboard to de-energize, causing this relay to triggered the I/O board input 13 (Propeller pin 16) for 10 seconds or more. This will set a value of 3 in the variable "txstatval" and send that value via the serial port on the I/O board (Propeller Pin 27) to the Spinneret (J6 connection) to be displayed on the web page after all alarm inputs are scanned by the Propeller.

When the Port power breaker is turned off or power is lost (breaker being tripped or utility is lost) this will cause the Port Shore power breaker relay in the main switchboard de-energize, causing this relay to triggered the I/O board input 14 (Propeller pin 17) for 2 seconds or more. This will set a value of 4 in the variable "txstatval" and send that value via the serial port on the I/O board (Propeller Pin 27) to the Spinneret (J6 connection) to be displayed on the web page after all alarm inputs are scanned by the Propeller.

Propeller (Spin) Code:

I/O controller:

The code for the I/O controller is straight forward in which the code uses only two Full Duplex Serial objects for to handle the serial ports. One is for the RS485 and the bi-directional serial to the Spinneret board and another serial port for the RC-4 and any other I/O boards.

The I/O controller uses two methods like (one in Main repeat loop and one child for each of the inputs) this to read the inputs and sets the comm. variables for the Spinneret and turn on the outside alarm light. Note this spin code is shown for 5 out of the 9 inputs.

```
PUB Main
if ((timer1 < 30) and (ina[Acbilgeloss] == 0))
  Bilgeinput1

pub Bilgeinput1
timer1 := ++timer1 * ina[Bilge1]   ' update timer1
if timer1 == 30                    ' looks for "timer1" = 30
  outa[Alarmlt] := 1               ' turns the Alarm Light on
  txalarmval := 1                  ' 120VAC Bilge Panel input 1 alarm value
                                   ' to be transmitted to the spinneret
```

The I/O controller uses two methods like (one in Main repeat loop and one child for each of the inputs) this to read also the inputs and sets the comm. variables for the Spinneret and turn on the outside alarm light, but it also prevents the comm. variable from being sent to the Spinneret when the Acbilgeloss is on. Note this shown is done for 4 out of 9 inputs

```
Pub Main
if (timer2 < 30)
  Bilgeinput2

pub Bilgeinput2
timer2 := ++timer2 * ina[Bilge2]   ' update timer2
if timer2 == 30                    ' looks for "timer2" = 30
 outa[Alarmlt] := 1                ' turns the Alarm Light on
'set_relay(%00, 1, 1)              ' turns on RC4 relay "K1"
   if (ina[Acbilgeloss] == 1)      ' 12VDC Bilge Panel loss relay de-energize
     txalarmval := 10              ' Acbilgeloss alarm value
  else
     txalarmval := 2               ' 120VAC Bilge Panel input 2 alarm value
                                   ' to be transmitted to the spinneret
```

The above code is design to be read when any one of the TimerX is not at its set point (less than 30). When it reaches its set point the main code will not call its child method (such as Bilge1).

The I/O controller uses this code to read the input connected to the Fire Alarm panel alarm relay and will set the comm. variable for the Spinneret and calls its child code when true. The child code allows the outside horn to be silenced when the SilencePb is true, but the txalarmval is continue to be sent to the Spinneret.

```
  if ina[Faalarm] == 1
    FA_in_alarm

Pub FA_in_alarm
 txalarmval := 14              ' F/A panel panel alarm value
```

```
 if (ina[Silencepb] == 1)
   outa[Fahornsil] := 1
```

The I/O controller uses this Main method code to read the states of all of the inputs related to the Alarm inputs and clears all the timers to “0” when “bilgestates” == 0 and sets txalarmval to 0, which is transmitted to the Spinneret when all the inputs are in a reset state.

```
   bilgestates := ina[Faalarm..Bilge1] ' store the states of INA in "bilgestates"
   if bilgestates == %0000000000       ' wait for all "bilgestates" inputs to be off
     outa[Alarmlt] := 0                ' turn off Alarmlt
     'reset_rc4(%00)                   ' turns off all relays on the RC4
     timer1 := 0                       ' clear all timers to 0
     timer2 := 0
     timer3 := 0
     timer4 := 0
     timer5 := 0
     timer6 := 0
     timer7 := 0
     timer8 := 0
     timer9 := 0
     txalarmval := 0                ' clear "txalarmval" to 0
     outa[Fahornsil] := 0           ' turn off Fahornsil when F/A panel panel
                                    ' is not in alarm
```

This I/O controller Main method code reads inputs related to the status inputs like the F/A Trouble, Acbilgeloss, Port Shore Power Breaker and the Port Power Breaker (located in the main switchboard)

```
   if timer10 < 30 and statpos == 1
     FA_trouble

   if timer11 < 30 and statpos == 2
     AC_bilge_loss

   if timer12 < 100 and statpos == 3
     Port_pwr_loss

   if timer13 < 30 and statpos == 4
     Port_breaker_status

   if restimer == 6000
     timer10 := 0
     timer11 := 0
     timer12 := 0
     timer13 := 0
     restimer := 0
     statpos := 0

   statpos ++
   restimer ++
```

This I/O controller method controls when "txalarmval" and "txstatval" is transmitted to the Spinneret, by using the variable “txpos” to control when it’s transmitted. This method is call twice in the main repeat loop in the middle and at the end.

```
pub txcomm_control



  if txpos == 1                ' transmits "txalarmval" when txpos ==1
     serial.dec(txalarmval)
     serial.tx (13)            ' sends a carriage return

  if txpos == 2                ' transmits "txstatval" when txpos == 2
    serial.dec(txstatval)
    serial.tx (13)             ' sends a carriage return

  if txpos == 4                ' resets "txpos" to 0 when "txpos" == 3
     txpos := 0

  txpos ++                     ' advances txpos which controls when
                               ' serial is transmitted
```

This I/O controller method receives the Spinneret commands using the variable “rxsilence” for resetting and silencing the I/O controller. This method is call at the beginning of the repeat loop in the Main method.

```
pub spinneret_control

  ' spinneret remote control

  rxsilence := serial.rx
  case rxsilence

    "f":
      outa[Fahornsil] := 1  ' remotely silence outside F/A horn
      'set_relay(%00, 2, 1)  ' turns off K2 relay on the RC4

    "b":
      outa[Alarmlt] := 0    ' turns off the Alarm Light
      'set_relay(%00, 1, 0)  ' turns off K1 relay on the RC4

    "x":
      outa[Alarmlt] := 0    ' turns off the Alarm Light
      outa[Fahornsil] := 0  ' resets outside F/A horn
      'reset_rc4(%00)       ' turns off all relays on the RC4
      'reboot               ' software reboot
```

Spinneret code:

The Spinneret code that was going to be used is being finished at this time, this document is being written. But, I used the Spinneret Demo code as an example, which I changed for testing and provide everyone an idea how are system is going to work.

What I did, was I kept the first part of the code from Spinneret Demo and added this for looking at the serial port (J6), which uses this method for receiving “txalarmval” and “txstatval”, which is called at the beginning of the Main repeat loop.

```
Pub IOControllerInVal

  if txrxpos == 1
    txalarmval := Serial.rx
    PauseMSec(200)

  if txrxpos == 2
    txstatval := Serial.rx
    PauseMSec(200)
```

And change this part of the Main method repeat loop to call the different parts of the HTML code based the variables “txalarmval” and “txstatval”.

```
  bytemove (@num1,STR.numberToDecimal(txalarmval, 4),5) ' show 120VAC alarm number in html code
  bytemove (@num2,STR.numberToDecimal(txalarmval, 4),5) ' show 12VDC alarm number in html code
  StringSend(0, @html1)
  StringSend(0, @html2)

  if txalarmval >= 1 or txalarmval <= 9
    StringSend(0, @html3)             ' Show 120VAC Bilge Panel alarm values

  if txalarmval >= 10 or txalarmval <= 13
    StringSend(0, @html4)             ' Show 12VDC Bilge Panel alarm values

  if txalarmval >= 14
    StringSend(0, @html5)             ' Show F/A Panel in alarm

  if txstatval == 1
    StringSend(0, @html6)             ' Show F/A Panel in trouble

  if txstatval == 2
    StringSend(0, @html7)             ' Show 120VAC Bilge Panel is Offline

  if txstatval == 3
    StringSend(0, @html8)             ' Show Port Shore Pwr Breaker is Off

  if txstatval == 4
    StringSend(0, @html9)             ' Show Port Pwr Breaker is Off
```

Also change this part of the repeat loop to control when to open the socket on the W5100 chip.

```
    ' we don't support persistent connections when txalarmval == 0
    ' and txstatval == 0, so disconnect open socket here
    if txalarmval == 0 or txstatval == 0
      W5100.SocketTCPdisconnect(0)
      PauseMSec(25)

    'Connection terminated
      W5100.SocketClose(0)
      PST.Str(string("Connection complete", PST#NL, PST#NL))
```

Also change this part of the SetButtons method, which controls silencing of the alarms.

```
PUB SetButtons     'Toggle button state and dynamically change html code



    If ButtonSelection == 24   'Silence button state when "alarmvar" >= 1
      Silence := 1 - Silence
      Serial.tx("b")
      Serial.tx(rxsilence)

    If ButtonSelection == 25   'F/A Horn button state when "alarmvar" = 14
      FAHorn := 1 - FAHorn
      Serial.tx("f")
      Serial.tx(rxsilence)

    If ButtonSelection == 26   'Resets to I/O controller when toggled
      IOReset := 1 - IOReset
      Serial.tx("x")          'Sets "rxsilence" to "x"
      Serial.tx(rxsilence)    'Sends "x" to I/O controller



    If Silence == 0
      bytemove(@buttons[86+70*0],@RED,5)  'When "txalarmval" >= 1 then do this
    Else
      bytemove(@buttons[86+70*0],@GREEN,5)'When "txalarmval" = 0 then do this

    If FAHorn == 0
      bytemove(@buttons[86+70*1],@RED,5)  'When "txalarmval" = 14 then do this
    Else
      bytemove(@buttons[86+70*1],@GREEN,5)'When "txalarmval" <= 14 then do this

    If IOReset == 0
      bytemove(@buttons[86+70*1],@RED,5)
    Else
      bytemove(@buttons[86+70*1],@GREEN,5)
```

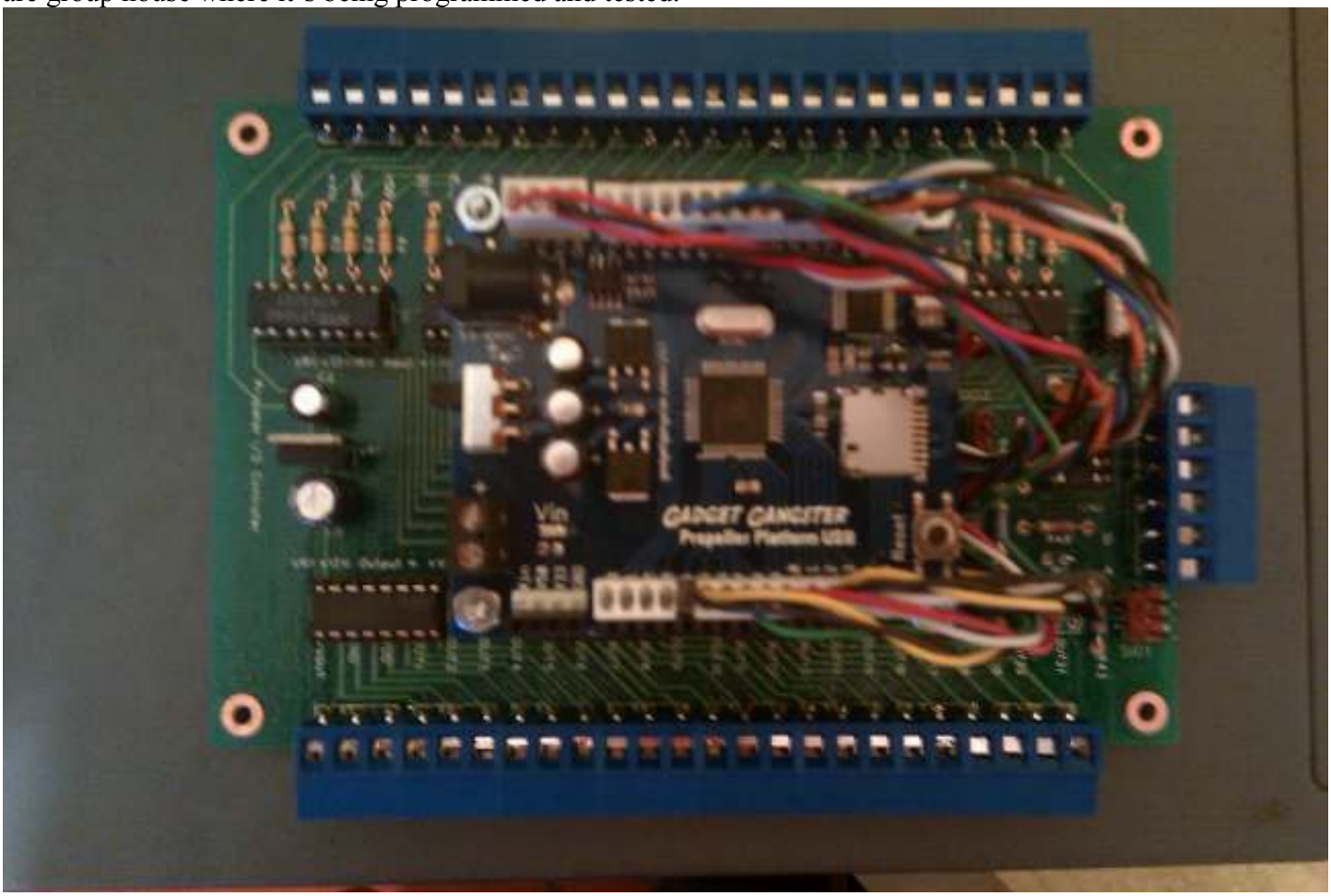
Other Uses:

Systems like this are used today in many applications and they are:

1. Energy Management Systems
2. Security Systems

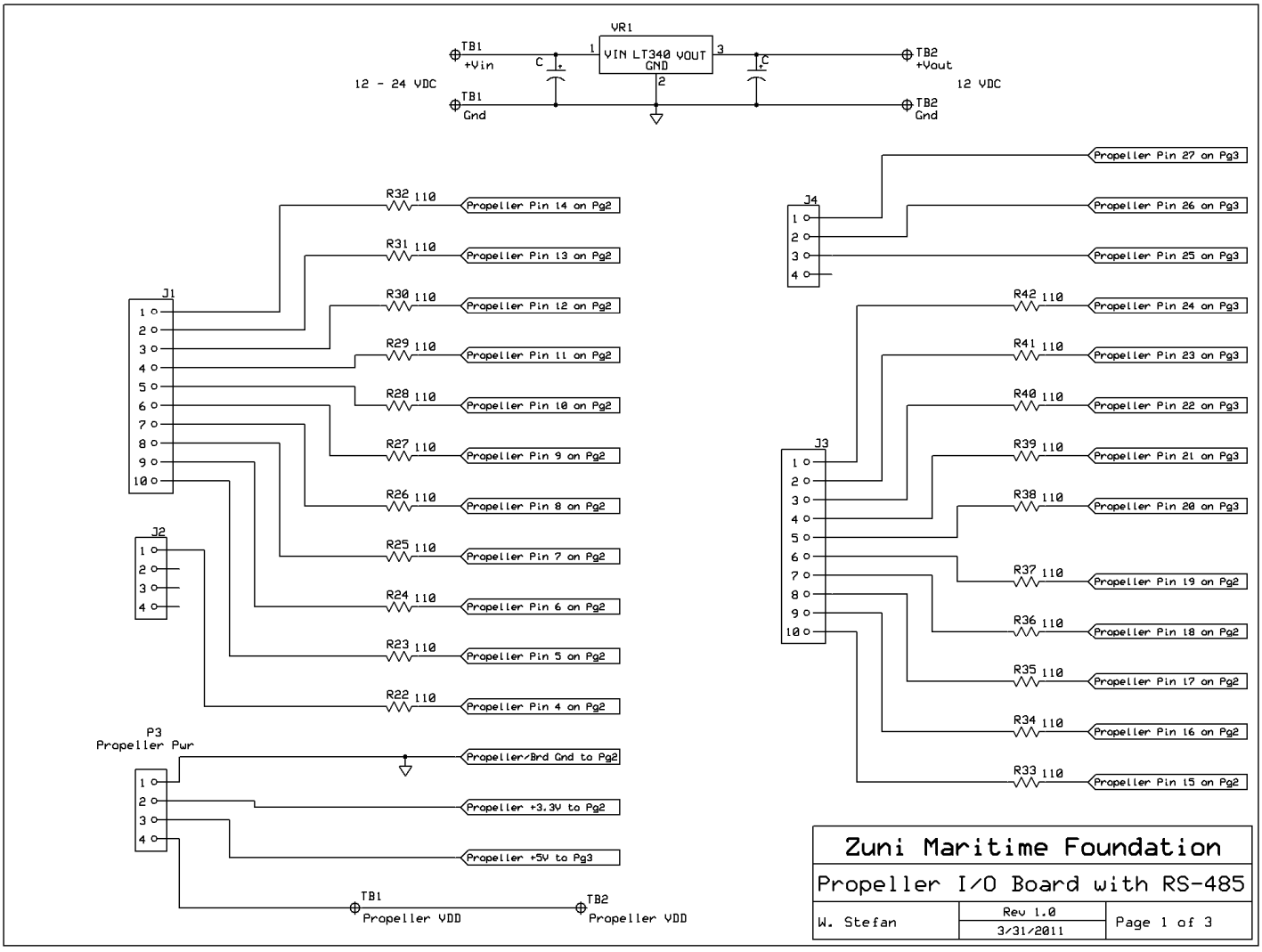
Photos and Drawings:

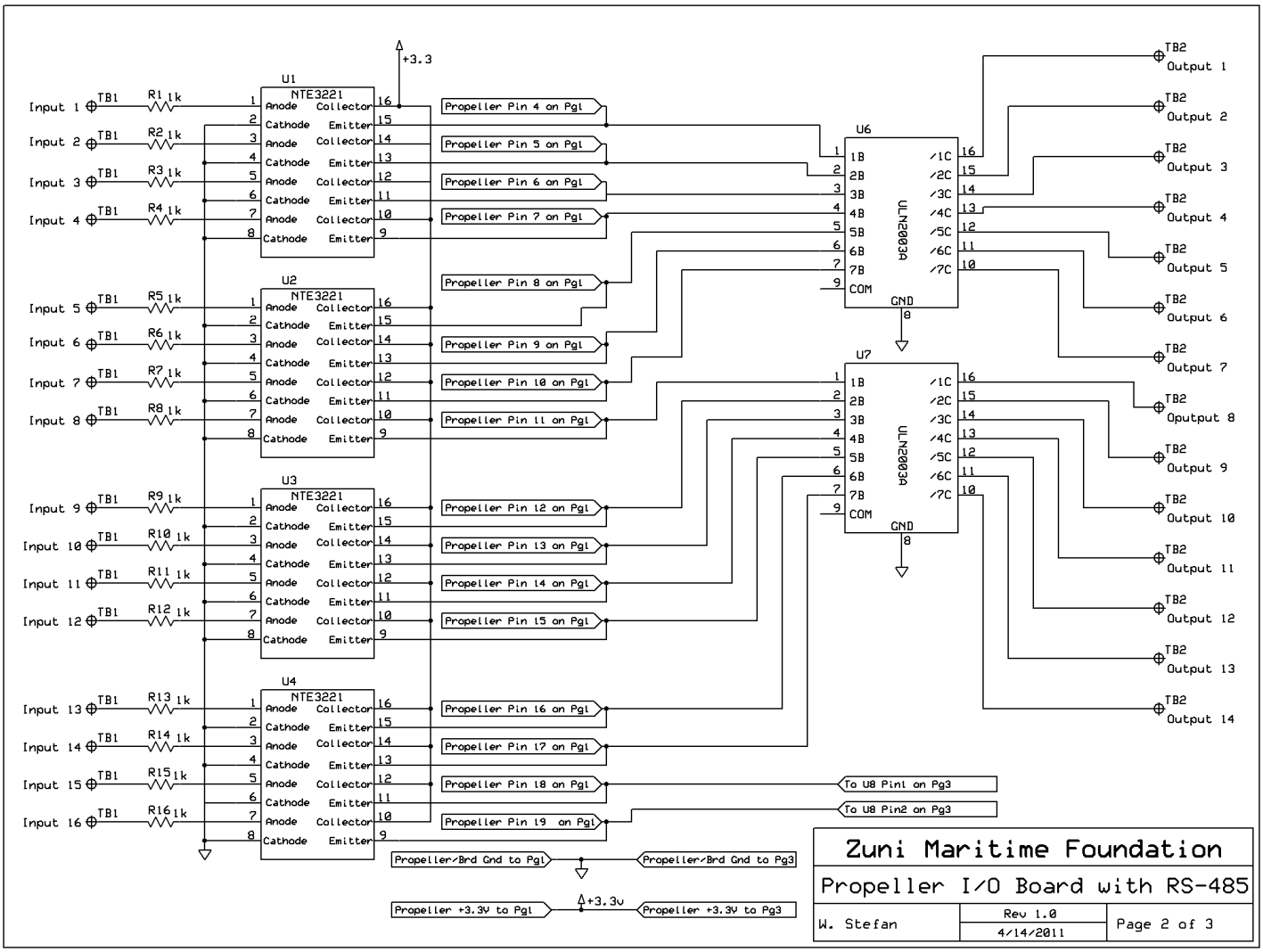
I will provide this one photo of the I/O Controller, since the Spinneret is located down at are other member from are group house where it's being programmed and tested.



The Spinneret will be connected at SIO2, which is just to the right of the SD card slot on the Propeller Platform. The inputs where the Bilge floats will be connected too are on the top and the outputs are where Alarm Horn/Light relay and F/A Horn Silence relay are connected on the bottom right. The 12VDC power supply that is located on the left of the Propeller Platform will be able to supply 1 amp of power for the Spinneret and the Propeller Platform from the +VOUT on the I/O board.

The next 3 pages of drawing below show how everything is interfaced to the Propeller Platform.

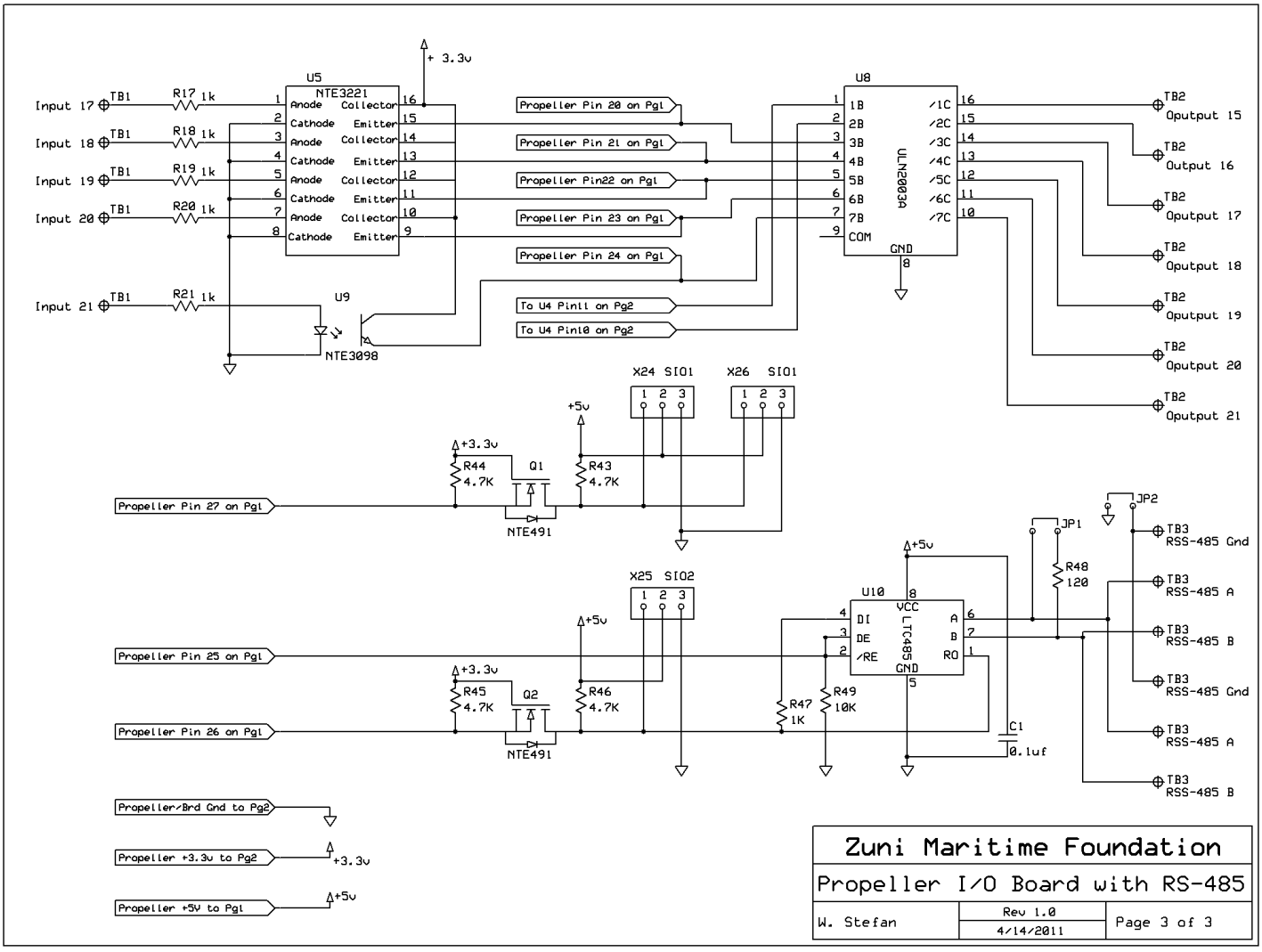

VR1
TB1
+Vin
TB1
Gnd
12 - 24 VDC
C
VIN LT340 VOUT
GND
1
2
3
C
TB2
+Vout
TB2
Gnd
12 VDC
J1
J2
J3
J4
P3
Propeller Pwr
R32 110
R31 110
R30 110
R29 110
R28 110
R27 110
R26 110
R25 110
R24 110
R23 110
R22 110
R42 110
R41 110
R40 110
R39 110
R38 110
R37 110
R36 110
R35 110
R34 110
R33 110
Propeller Pin 14 on Pg2
Propeller Pin 13 on Pg2
Propeller Pin 12 on Pg2
Propeller Pin 11 on Pg2
Propeller Pin 10 on Pg2
Propeller Pin 9 on Pg2
Propeller Pin 8 on Pg2
Propeller Pin 7 on Pg2
Propeller Pin 6 on Pg2
Propeller Pin 5 on Pg2
Propeller Pin 4 on Pg2
Propeller Pin 27 on Pg3
Propeller Pin 26 on Pg3
Propeller Pin 25 on Pg3
Propeller Pin 24 on Pg3
Propeller Pin 23 on Pg3
Propeller Pin 22 on Pg3
Propeller Pin 21 on Pg3
Propeller Pin 20 on Pg3
Propeller Pin 19 on Pg2
Propeller Pin 18 on Pg2
Propeller Pin 17 on Pg2
Propeller Pin 16 on Pg2
Propeller Pin 15 on Pg2
Propeller/Brd Gnd to Pg2
Propeller +3.3V to Pg2
Propeller +5V to Pg3
TB1
Propeller VDD
TB2
Propeller VDD
Zuni Maritime Foundation
Propeller I/O Board with RS-485
W. Stefan
Rev 1.0
3/31/2011
Page 1 of 3



Zuni Maritime Foundation

Propeller I/O Board with RS-485

W. Stefan	Rev 1.0 4/14/2011	Page 2 of 3


+ 3.3v
U5
NTE3221
Input 17
Input 18
Input 19
Input 20
Input 21
TB1
R17 1k
R18 1k
R19 1k
R20 1k
R21 1k
Anode
Cathode
Collector
Emitter
U9
NTE3098
Propeller Pin 20 on Pg1
Propeller Pin 21 on Pg1
Propeller Pin22 on Pg1
Propeller Pin 23 on Pg1
Propeller Pin 24 on Pg1
To U4 Pin11 on Pg2
To U4 Pin10 on Pg2
U8
ULN2003A
1B
2B
3B
4B
5B
6B
7B
COM
GND
1C
2C
3C
4C
5C
6C
7C
TB2
Oputput 15
Output 16
Oputput 17
Oputput 18
Oputput 19
Oputput 20
Oputput 21
X24 SIO1
X26 SIO1
X25 SIO2
+5v
+3.3v
R44 4.7K
R43 4.7K
R45 4.7K
R46 4.7K
Q1
Q2
NTE491
Propeller Pin 27 on Pg1
Propeller Pin 25 on Pg1
Propeller Pin 26 on Pg1
U10
LTC485
VCC
DI
DE
/RE
RO
A
B
GND
R47 1K
R49 10K
R48 120
C1
0.1uf
JP1
JP2
TB3
RSS-485 Gnd
RSS-485 A
RSS-485 B
RSS-485 Gnd
RSS-485 A
RSS-485 B
Propeller/Brd Gnd to Pg2
Propeller +3.3v to Pg2
Propeller +5V to Pg1
Zuni Maritime Foundation
Propeller I/O Board with RS-485
W. Stefan
Rev 1.0
4/14/2011
Page 3 of 3