Digital PID Control Of A Space Station
Simulated and Scale Model
Part I

By John Blankenship And Samuel Mishal

The Project

The field of control is a very challenging engineering discipline that can also be extremely satisfying once mastered. The aim of any control mechanism is to manipulate parameters of a system so as to achieve a desired state and to resist external disturbances that act to deviate the system from this desired state. Additionally, the mechanism should allow an operator to command changes and to reposition the system accordingly in as an efficient manner as possible.

The way a control mechanism achieves the above goals is the objective of the designing engineer. The design process can be very mathematical and is normally quite complicated. However, the ultimate aim is to determine parameters for the control mechanism that will allow it to achieve the desired characteristics of the system response to external changes as well as commanded changes. These parameters can sometimes be determined by using trial and error if a scale or simulation model is available.

In some situations, no matter how rigorous the design process is, it will still be vital to model the system and test the design on the model before building the actual system. The modeling process can be one or both of two approaches:

1. Build a scale model of the system
   This can be quite an involved engineering process. Considerations have to be made to the characteristics of the model and how they relate to the actual system. Any insight gained from the model will have to be considered in the light of this relationship.
2. Simulate the system on a computer
   Simulating a system using a computer model is not an easy process and may even be a more involved engineering project than building a physical model. However, the final result can be more versatile and more amenable to modifications and experimentation than a physical model. Additionally, in some situations, the computer model can be more representative of the physical system than a scale model and can be readily used to train operators in the future.

As an illustration of all the above, we are going to design a computer simulation as well as a scale model of a Space Station and control them using a control system that will allow us to change the orientation (heading) of the station. The system will allow for manual as well as automatic control.

Controlling A Real Space Station

A real Space Station in orbit would have numerous sub-systems. Each sub-system would have an independent microcontroller to monitor and command the sub-system's various mechanisms (each of which may need a sub-microcontroller). All the microcontrollers would be networked to

a master controller that receives information from the various sub-controllers and commands them.

The master controller is the point of interface between the Space Station and a control system that allows a human operator to command the station. The control system would have a user-friendly interface that allows the operator to control the station using ergonomic controls and displays. The human interface system may be connected to the master controller directly or via a wireless link to facilitate controlling the station from an earth command center as well as from onboard the station.

The control system receives data from, and sends commands to the master controller. The master controller collects the requested data from the various sub-controllers and distributes the commands to the intended sub-controllers. This process goes on during the service life of the station (see Figure 1).

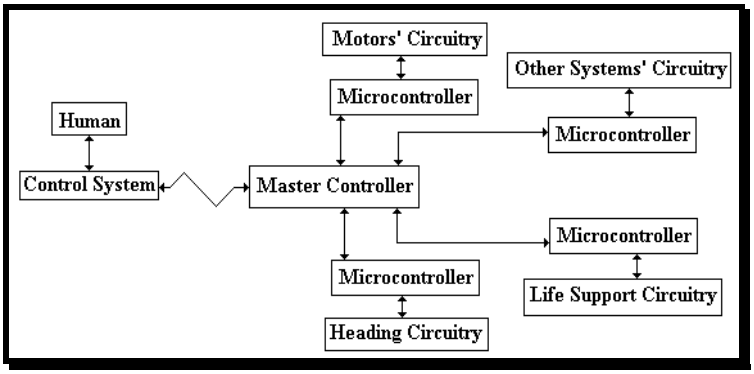

Figure 1: Control Hierarchy

Controlling the orientation (heading) of the station is not an easy matter. From Newton's laws we know that when a force is applied to a mass the mass will experience acceleration. This acceleration will change the velocity of the mass. Once the force is removed the acceleration stops and therefore the change in velocity will also stop. However, in space there is no friction and whatever velocity the mass has when all forces desisted will remain. Therefore to stop the mass (zero velocity) an appropriate force has to be applied to decelerate the mass until it stops.

Controlling the heading of a Space Station can be achieved using a pair of rockets in opposing orientations. One is fired to rotate the station for a certain duration. The other is then fired until the station stops. When, for how long, and at what burn-rate to fire the rockets so as to effect a particular heading change is the aim of the controller. Another consideration while controlling the heading is how to determine what the heading is. This can be done using a pair of stars, or a signal from earth, or even from other artificial satellites.

The Scale Model Station

We are going to build a scale model of the Space Station and control it the way we would the real station. We will model the space station with a thin disk and fans (see Figure 2). However there are three major problems to consider:

1. Zero Gravity

No matter what we might be able to do down here on the surface of earth, we would not be able to effectively simulate zero gravity (at least for an effective period). Therefore we will need to support the disc in some manner to simulate a free-floating station.

2. Friction Free Movement
   The rotation of the real station in space is frictionless. Since the model is supported by some mechanism, it will not be frictionless.
3. The Propulsion Mechanism
   Obviously for our limited resources we will not be able to model a rocket system. To achieve propulsion on the model we will use electrical fans. Also since the model is not in a vacuum there will be an effect on the station due to air resistance.

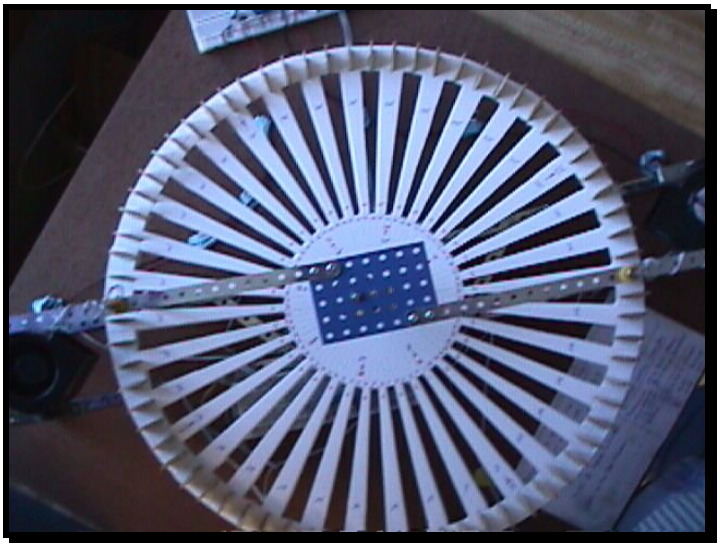

Figure 2: Scale Model. Notice the fans on the left and right.

The Computer Simulated Station

In addition to the scale model, we are also going to design a computer simulation model of the station. The simulation will use mathematical formulas to calculate the force, acceleration and rotational speed of the station as well as the heading. The mathematical model will be used to calculate the response of the station in reaction to commanded rocket firings.

To simulate a real space station we will need the properties of the real station as parameters for the mathematical model. However, since we are going to use the control system to control the scale model as well as the simulated model, it would be nice if we had the simulated model's parameters be the properties of the scale model rather than an actual full station. This way we can have some comparison between the two systems.

The Control System

The control system will interface with a human operator to receive commands as to what the desired heading is and will display information back that will indicate the pertinent parameters of the system.

The control system will constitute three sections:

1. User Interface

This will allow the user to specify various parameters and will display the current status of other parameters of the station. Some of the parameters may not be relevant in the case of controlling the scale model. The operator will be able to command heading changes as gradual or quantum steps.

2. Manual Control
   This will allow the user to control the firing of the rockets to move the station manually. The operator is responsible for controlling the station's heading.
3. Automatic Control
   This will control the firing of the rockets automatically to reposition the station to the commanded heading.

The control system will be used to control the simulated as well as the scale model. Thus there are four modes of operation:

1- Manually Controlling the scale model.
2- Manually Controlling the simulated (mathematical) model.
3- Automatically Controlling the scale model.
4- Automatically Controlling the simulated (mathematical) model.

Switching between the four options will be achieved by a user-friendly interface.

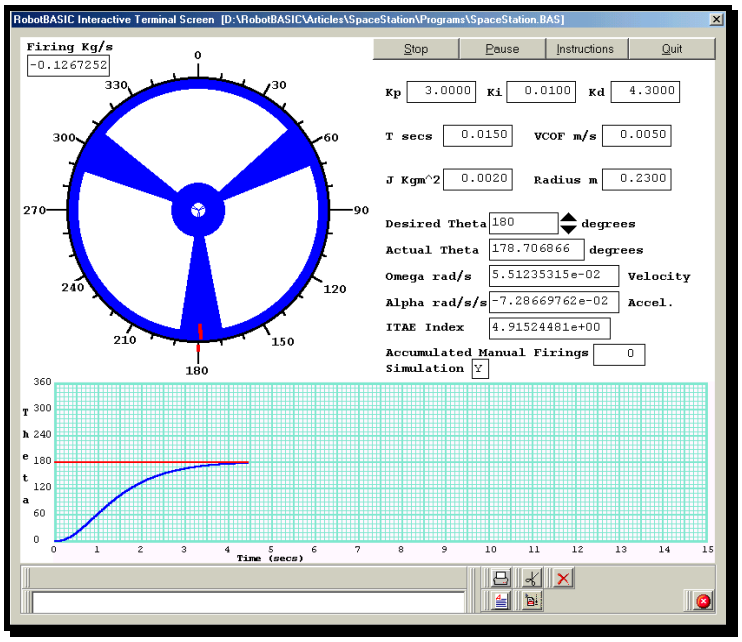

Figure 3: Control System (You can also see the elements of the simulated system)

Automatic Control

To automatically control the station (scale or simulated model) we will use a negative feed back control mechanism along with a PID (proportional, integral, differential) controller (see Figure 4).

In the case of this project the parameter under control is the heading of the station. Negative feedback will generate an error amount that is the difference between the desired heading and the actual heading. With the error value as an input, the PID controller will generate a firing rate for

the rockets. A negative rate will produce counter-clockwise rotations and a positive rate will cause clock-wise rotations. The rate will control the fuel burn rate in the rockets (speed of the fans).

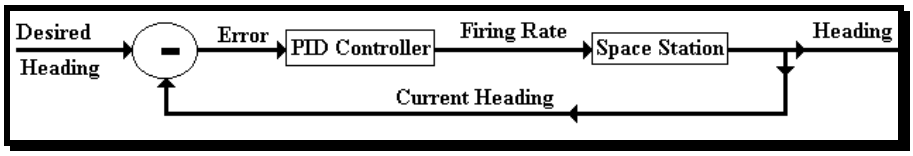

Figure 4: Negative Feed Back System.

PID means that the error amount will be considered in three ways:

1- The amount of error (Proportional).
   The firing rate will be in proportion to the error amount. The further the current heading from the desired value, the higher the firing rate.
2- The accumulated error so far (Integral).
   The firing rate will be in relation to the accumulation of the errors so far. The sum of the errors so far will affect the firing rate. The more the errors persist the more the rate of firing. In other words the firing rate depends on the history of the errors as well as their amount.
3- The rate of change of the error (Differential).
   The rate of change of the error will affect the firing rate. In other words the rate of firing will depend on the speed with witch the station is approaching the desired value.

Measuring the error is a straightforward process. However, measuring the rate of change of the error and the accumulated sum of the errors is not so simple. Before the advent of fast and capable computers, a PID controller was achieved using Operational Amplifiers (***Op-Amps***) that where configured to create ***Amplifiers, Integrators*** and ***Differentiators***. These functioned quite nicely and as long as there were appropriate transducers that could translate physical characteristics into voltage levels and actuators that could translate voltage levels into forces, these ***analog controllers*** were very effective and capable.

With the advent of fast and capable computers, microcontrollers, and dedicated digital systems, the field of ***Digital Control*** became a viable alternative to the analog systems. At first digital controllers were designed as a direct replacement for the analog systems, but soon the field of digital control took on a character all of its own. Now with digital control using computers we can implement more effective methods such as ***Adaptive Control***, ***Fuzzy Control***, and other sophisticated systems like ***neural networks***.

The field of control dates back to the 19th century (not counting DaVinci and Archamedes). Mechanical control systems like the steam engine fly wheel where replaced with electrical (analog) systems in the 20th century. In the late 20th century digital control systems almost totally replaced analog systems. The body of knowledge developed by engineers for designing the mechanical and electrical systems is still fully relevant and indispensable for designing the digital systems. Engineers used techniques developed by mathematicians and engineers like

Nyquist, Laplace, Fourier, Bode and many more to devise mathematical and graphical methods for analyzing a system to determine the parameters of the controller systems.

Considerations for things like the settling time, maximum overshoot, steady state error, dampening factor, and many more were the parameters that indicated and dictated the quality of the resulting systems. Engineers deployed methods like Zero-Pole plots, Bode plots, and many more graphical shortcuts to be able to solve the complicated ***differential equations*** that resulted from the mathematical analysis of the ***physics*** of the system.

The final outcome of all the analysis is to determine the three parameters that affect the PID controller's output. The three factors are K_p, K_i, K_d. These factors determine how the PID controller will make the system behave. Will the system will achieve the desired final value quickly or slowly. Will it oscillate then settle, continue oscillating at a constant level or will it be unstable and oscillate increasingly until it destroys itself.

The time it takes and the way the system reaches the desired state is of importance as well as how it responds to external disturbances that may cause it to deviate from the desired state after it has settled. In the case of a space station, characteristics like the power available from the rockets, the number of rockets, the radius of the station, the mass distribution (shape) of the station, the maximum allowable g-forces, the maximum allowable error, and the allowable amount of oscillations are all factors that will affect how the system will respond and how we desire it to respond.

For digital control systems the rate of sampling of the system is of paramount importance. If our digital controller samples the heading values of the system slower than the rate of change of the heading then the controller will give erroneous values that can be misleading. This is called the ***Nyquist criterion***. Simply stated, this criterion limits the minimum sampling rate the controller must have in relation to the maximum harmonic of the system.

In a digital controller we do not have a continuous measurement of the parameter we wish to measure. Rather we use an Analog to Digital Converter (ADC) that converts the level of the parameter to a digital value. ADCs are not able to follow the continuous variation of the value under measurement. Rather, they take a snap shot of the value at a regular rate called the sampling rate. From these regular snap shot values, you can approximate the analog values using various mathematical tools. The details are not of importance here. All that matters is to know the sampling rate.

Imagine that the heading error is oscillating at such a rate that every 0.1 seconds the error is zero, but it is oscillating between two values above and below zero. If our controller sampled the error level every 0.1 seconds then it will always sample an error of zero and will think that the desired value has been reached and thus will react incorrectly. If the sampling rate was, say, every 0.01 seconds then the controller will be able to catch intermediate error levels and thus react accordingly.

The Nyquist criterion simply states that the minimum sampling rate must be no less than twice the frequency of the maximum harmonic of the system under consideration. So if there are

changes in the value to be sampled that occur at the rate of 100 changes per second, then our controller must sample the value at a rate no less than 200 times per second (the more the better).

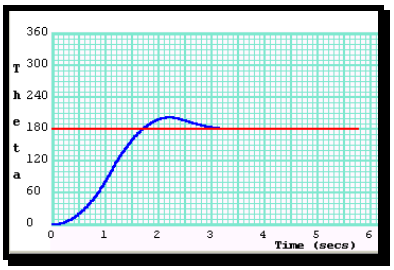

Figure 5a

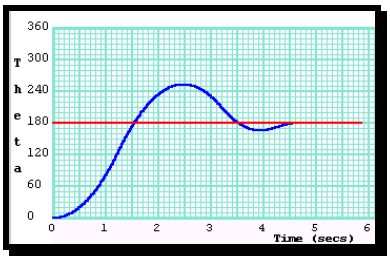

Figure 5b

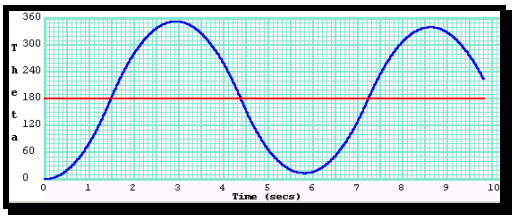

Figure 5c

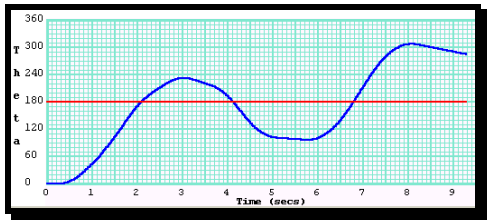

Figure 5d

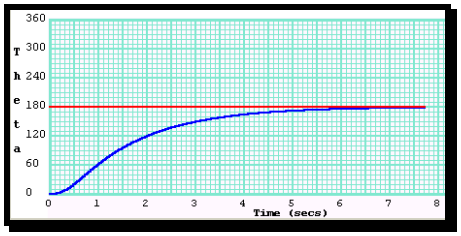

Figure 5e

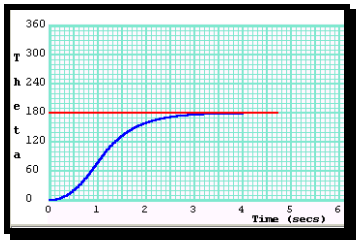

Figure 5f

The plot in Figure 5a shows the heading of the station along a time scale after a heading change has been commanded. This type of plot is called the ***Step Response***. You will notice that the system is quite good in arriving at the desired heading. It gets there quickly but overshoots a little before settling down to the desired value. This type of response may be unacceptable if overshooting is unacceptable. Also the amount of overshoot has to be considered. Another consideration is that even though the station responded very quickly, the accelerations that resulted (i.e. g-forces) were too large and the occupants of the station (humans and equipment) were subjected to intolerable g-forces.

Figure 5b shows another possible response. In this situation the station oscillates a little around the desired value with decreasing amplitude. This response is rarely acceptable.

Figure 5c is similar to Figure 5b except the oscillations never die out the oscillation will remain forever. Unless your aim is to create a joy ride in space this response would be unacceptable. Figure 5d is an example of an unstable system where the station will eventually be in deep trouble.

Figure 5e shows a system that may be acceptable in certain situations, but is usually not the most desired response due to the long time it takes the system to eventually arrive at the desired setting. Figure 5f is usually the profile of the most ideal response. The response time is adequate

and not too fast for physical comfort. Also there is no overshoot with the resulting sudden changes in headings.

Figures 5a to 5f were derived using the simulation. Had we not had the computer simulation it would have been hard to deduce the response profiles from the differential equations that describe the physics of the system. That is why traditional control systems design involved many graphical approximations and transformations like the Laplace and Fourier transforms along with Pole-Zero analysis and various other *tricks of the trade* to enable visualization of the responses you see in the figures. We are lucky to have the computational power that enables us to solve the differential equations using numerical analysis methods to be able to get a visual plot of the step response of the system.

Another advantage in using the computer to solve the formulas is that a solution for the best-behaved system can be arrived at using trial and error in place of comprehensive analysis. Of course to be able to ***intuit*** a good trial and error guess you need experience in how the various parameters affect the results. Nevertheless, it is definitely easier for a less mathematically capable person to arrive at a solution. Additionally, due to the ease of changing parameters, you can experiment with many *what-if* situations and gain a lot of experience and insight that would be difficult to gain from the traditional (and cumbersome) analysis methods.

Next Month

Next month we will go into the details of how to implement the simulated station using mathematical formulas. Also we will show how to build a scale model with a master controller and control it using the same program that controlled the simulated model.

Digital PID Control Of A Space Station
Simulated and Scale Model
Part II

By John Blankenship And Samuel Mishal

Last month we analyzed the requirements for building a computer simulation as well as a scale model of a space station attitude control system. This month we are going to implement the project. The project consists of four parts. The Control Center PC based program, the Master Controller program, the Quadrature Encoder program and associated electronics, and finally the Motor Controller and its associated electronics. Additionally there is the building of the scale model's mechanical system.

Implementing The Control System

The Control Center system will be written for an MS Windows based PC using a language called RobotBASIC that can be downloaded for free from www.RobotBASIC.com. The language is easy to use while having powerful graphical, mathematical, user interface, and serial communications tools that will be necessary for the successful completion of this project with ease.

The program will have the following tasks to achieve:

- Allow the user to specify whether Automatic or Manual control is desired.
- Allow the user to specify whether the control is to apply to the Scale Model or the Simulation Model
- Allow the user to define the physical characteristics of the space station to be used for the mathematical simulation model.
- Allow the user to specify the values for the PID controller as well as the sampling rate for use during automatic control.
- Allow the user to specify the desired heading and to command step changes or gradual changes during automatic control.
- Allow the user a means to commanding the firing rate of the rockets during manual control.
- Display to the user the current values of the Heading, Rotational Speed, Rotational Acceleration, Firing Rate and other pertinent data that indicate the instantaneous state of the station.
- Display a Graph of the system response along a time scale.
- Generate a real-time animation of the system.
- Manage the communications between the control system and the master controller on the scale model during control of the scale model.

The user interface is shown in Figure 3. All the elements detailed above are clearly visible. The program is listed in Figure 7 below. The listing is not a complete one due to space. A full listing is available for download from www.RobotBASIC.com. The listing in Figure 7 shows some of the control related subroutines.

The subroutine *CalcResponse* is used during the automatic and manual control of the simulated model. It calculates the acceleration and speed from the burn rate (see below) then calculates the heading change.

The *CalcStationResponse* subroutine is used during the automatic and manual control of the scale model it sends the burn rate to the master controller on the scale model and then receives a number that enables calculation of the heading (see later). Then it calculates the speed and acceleration from the heading change.

To model the space station using a mathematical model, the following formulas are used:

- Burn Rate (Kg/s) x VCOF (m/s) = Applied Force (Kgm/s^2 or N)
  - Burn Rate is the rate of fuel flow through the rockets.
  - VCOF is a factor that gives a measure of the Velocity gained from burning one Kg of fuel per second. In other words this is an energy factor for the fuel.
  - Multiplying these values gives the value of the force exerted by the rocket.
- Force Applied (N) x Radius (m) = Torque (Nm)
- Torque (Nm) ÷ J (Kgm^2) = Alpha (radians/s^2) (α = Rotational Acceleration)
  J is the rotational inertia. It depends on the shape and mass distribution of the Station. If the station is a disc with mass M (kg) and radius R (m) then J = ½MR^2
- $\int\alpha$ = Omega (radians/s) (ω = Rotational speed)
- $\int\omega$ = Theta (radians) (θ = Heading)

Thus to find out the current heading due to the commanded burn rate the block diagram in Figure 6 will substitute the block for the Space Station in Figure 4 (last month). The block 1/S represents the Laplace transform of an integrator.

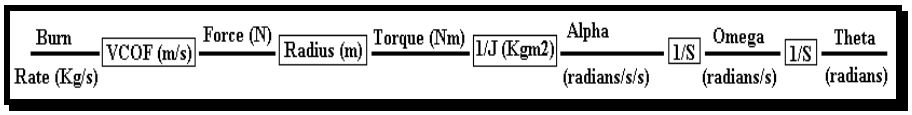

Figure 6: Substitute Block For the Simulation Model Of The Station.

To implement an integrator using digital systems we will use the formula:

Current Output = Previous Output + (Current Input + Previous Input) x T ÷ 2

Or

Y = YZ1 + (X + XZ1) x T ÷ 2

Or

$Y = Y \times Z^{-1} + (X + X \times Z^{-1}) \times T \div 2$

T is the sampling period (seconds). Basically this is the ***trapezoid rule*** for ***numerical integration***. In the program listing you will notice that instead of saying current input and previous input we say Input and InputZ1. InputZ1 stands for Input x Z^{-1} which is the Z-Transform method of saying previous input. InputZ2 means Input x Z^{-2} which means the previous-previous input value. The Z-transform is the digital method of analyzing systems and is related to the Laplace-transform. Both transforms are mathematical tools to facilitate solving differential (difference) equation.

The solutions will result in plots like the ones in Figure 5. These plots make it easy to visualize how the system will respond to commanded or disturbance changes.

Since this is a digital discrete system, inputs and outputs occur at discrete time intervals T seconds apart. The previous input is the input that occurred T seconds ago. Previous-Previous input is the input that occurred 2T seconds ago. These values are initialized at zero and are stored in variable in the program as time moves forward.

The program will use digital PID controller. The formula for the PID controller is:

$$\textbf{PID\_output} = \textbf{PID\_output} \times Z^{-1} + A0 \times \textbf{error} + A1 \times \textbf{PID\_error} \times Z^{-1} + A2 \times \textbf{PID\_error} \times Z^{-2}$$

Where

$$A0 = K_p + (T \times K_i \div 2) + (K_d \div T) \quad , \quad A1 = -K_p + (T \times K_i \div 2) - (2 \times K_d \div T) \quad , \quad A2 = K_d \div T$$

For manual control, the PID block will be replaced with the burn rate value as commanded by the operator.

During the control (automatic or manual) of the scale model station the values for **ω** and **α** are calculated by numerically differentiating the change in **θ** once and twice. The formula for numerical differentiation is:

$$\textbf{Output} = (\textbf{Input} - \textbf{Input} \times Z^{-1}) \div T \quad \textbf{OR} \quad \text{Output} = (\text{Current Input} - \text{Previous Input}) \div T$$

```
  Cx           = 205
  Cy           = 197
  GraphColor   = 13693056
  IsSimulation = true
  CommPortNo   = 2
//===========================================================
//===========================================================
MainProgram:
  gosub SetUp
  gosub Instructions
  gosub MonitorInputs
  gosub FinishUp
End
//===========================================================
//===========================================================
CalcPID:
   PID_output = A0*error+A1*PID_errorZ1+A2*PID_errorZ2+PID_outputZ1
   PID_outputZ1 = PID_output
   PID_errorZ2 = PID_errorZ1
   PID_errorZ1 = error
   BurnRate = Limit(PID_output,-5,5)
Return
//===========================================================
CalcError:
   error = desired -actual
   if abs(error) > Pi() then error = error - sign(error)*Pi(2)
Return
//===========================================================
CalcResponse:
   //----Calc Theta, dTheta (W), ddTheta (Alpha)
   Torque = BurnRate*VCOF*Radius
   ddThetaZ1 = ddTheta
   dThetaZ1 = dTheta
   ThetaZ1 = Theta
   ddTheta = Torque/J
   dTheta = dThetaZ1+T/2*(ddTheta+ddThetaZ1)
   Theta = ThetaZ1+T/2*(dTheta+dThetaZ1)
```

```
   Alpha = ddTheta
   W = dTheta
   actual = frac((iactual+Theta)/Pi(2))*Pi(2)
   if actual < 0 then actual = actual+Pi(2)
   aTheta = round(rtod(actual))
Return
//=====================================================================
CalcStationResponse:
   if within(BurnRate,-0.3,0.3) then BurnRate = 0
   if BurnRate < 0
      CS_Speed =round(BurnRate/5.0*55-65)
   elseif BurnRate > 0
      CS_Speed =round(BurnRate/5.0*70+50)
   else
      CS_Speed = 0
   Endif
   CS_Sp = CS_Speed \ if CS_Speed < 0 then CS_Sp = abs(CS_Speed) | 128
   serout char(CS_Sp)
   serbytesin 2,CS_dAngle,CS_NoIn
   if CS_NoIn < 2
      n=ErrMsg(Msgs[10],Msgs[0],MB_OK|MB_ERROR)
      GoSub StopStation
      CommsError = true
      return
   endif
   CS_HB = ascii(substring(CS_dAngle,1,1))
   CS_LB = ascii(substring(CS_dAngle,2,1))
   CS_dAngle = (CS_HB<<8)+CS_LB
   if CS_HB & 128
      CS_HB= ~CS_HB & 255
      CS_LB = ~CS_LB & 255
      CS_dAngle = -((CS_HB<< 8)+CS_LB+1)
   endif
   ddThetaZ1 = ddTheta
   dThetaZ1 = dTheta
   ThetaZ1 = Theta
   Theta = CS_dAngle*dtor(2.5)
   dTheta = (Theta-ThetaZ1)/T
   ddTheta = (dTheta-dThetaZ1)/T
   Alpha = ddTheta
   W = dTheta
   actual = frac(Theta/Pi(2))*Pi(2)
   if actual < 0 then actual = actual +pi(2)
   aTheta = round(rtod(actual))
Return
//=============================================================
CalcITAE:
   s_time = s_time+T
   ITAE_error = abs(error)*s_time
   ITAE = ITAE_outputZ1+T/2*(ITAE_error+ITAE_errorZ1)
   ITAE_errorZ1 = ITAE_error
   ITAE_outputZ1 = ITAE
Return
```

Figure 7: Control Program Listing (most subroutines are not listed for brevity)

Master Controller

The Central Control system on the PC will need to communicate with the Master Controller on the Scale Model of the space station. The PC system requires to know the current heading of the station and to command a burn rate (voltage level) for the rockets (fans).

The Master Controller's job is to (Refer to Figure 8):

- Receive the burn rate (fan speeds and direction) value from the PC system.
- Command the Motors' controller with this new speed and direction.
- Interrogate the Quadrature controller to obtain a value for the current quadrarture count.

- Send the count value to the PC system (the PC system will use this value to calculate a heading).

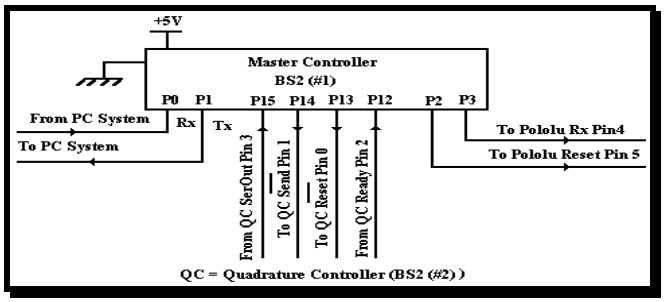

Figure 8: Master Controller Connections.

The Master controller will be a Basic Stamp 2 microcontroller (www.Parallax.com). The program to implement the above is written in PBasic and is shown in Figure 9. The BS2 is a very adequate controller for the tasks above. The only shortcoming of the BS2 is that it does not have a serial buffer and it does not implement serial communications with interrupts. So while it is sending or receiving it is incapable of doing other tasks. Also if an input is received while it is not listening the data will be lost. However, the BS2 is still very suitable if the PC program is written with consideration to these shortcomings. If you study the sequence of actions, the shortcomings are not a factor as long as the PC program does not send data until the BS2 is expecting it. Additionally the BS2 can be programmed using PBasic, which is easy to understand and use.

```
' {$STAMP BS2}
' {$PBASIC 2.5}
'=============    Variables         ==========================
'================================================================
    ReceivePin    PIN 0
    SendPin       PIN 1
    MotorReset    PIN 2
    MotorPin      PIN 3
    QuadratureRx  PIN 15
    QuadratureSt  PIN 14
    QuadratureRSt PIN 13
    dAngle        VAR byte(2)
    MotorSpeed    VAR Byte
    Direction     VAR Nib
'================================================================
Main:
  GOSUB Initialize
  DO
    SERIN ReceivePin,84,[MotorSpeed] 'receive motor speed and direction
    IF MotorSpeed = 255 THEN GOTO Main  'reset code is recieved
    GOSUB SetMotors  'command the motor speed
    GOSUB Quadrature ' interrogate the Quadrature controller
    SEROUT SendPin,84, [dAngle(1),dAngle(2)]  'Send the data back
  LOOP
END
'================================================================
Initialize:
  QuadratureSt   = High
  QuadratureRst  = Low
  QuadratureRst  = High 'Reset Quadrature
  LOW MotorReset
  HIGH MotorReset  'reset motor
  PAUSE 100
  SEROUT MotorPin,84,[$80,0,0,0] 'motor 0 brake
```

```
  PAUSE 20
  MotorSpeed  = 0
  Direction = 0
  dAngle = 0
RETURN
'=============================================================
SetMotors:
  IF MotorSpeed.BIT7 = 1 THEN
     SEROUT MotorPin,84,[$80,0,0,MotorSpeed & 127] 'backwards (ccw)
  ELSEIF MotorSpeed = 0 THEN
     SEROUT MotorPin,84,[$80,0,0,0] 'brake
  ELSE
     SEROUT MotorPin,84,[$80,0,1,MotorSpeed & 127] 'forward (cw)
  ENDIF
RETURN
'=============================================================
Quadrature:
    QuadratureSt = Low  'tell the quadrature controller to send data
    QuadratureSt = High
    SERIN QuadratureRx,84,[dAngle\2]  'receive the data
Return
```

Figure 9: Master Controller Program for the BS2

Motors Controller

The scale model has two fans that will be used to model the rockets for controlling the rotation of the space station. The fans are powered by DC motors. To control these motors we need:

- A controller to generate the PWM (pulse wave modulation) signals required to control the speed of the motors.
- Opto-Isolation to isolate the Motors' power supply from the microcontroller's supply.
- Darlington power transistors to be able to sink sufficient current to drive the motors.

For the PWM controller we will use the Pololu module(www.Parallax.com). This module can be commanded using asynchronous serial communication. It is able to drive two separate motors. We will use it in the single motor mode because of the way we are going to design the driving circuitry.

The circuit diagram is shown in Figure 10. The Master Controller will send the serial data to the Pololu commanding the desired direction and speed of the motor. The Pololu sets the two lines that go to the opto-isolators and sets either the left or the right isolator to be on and off at the PWM rate. The isolator will in turn activate the darlington transistor to drive the motor with the PWM rate which sets the fan on at the desired speed.

Rarely do motors have the exact same characteristics. Different motors will have different minimum starting voltage and different speeds for the same voltage. Thus you will need to tune the driving circuitry so that the motors have roughly the same starting voltage and roughly the same speeds. This can be achieved via the variable resistors that control the level of the base voltage applied to the darlington transistor. Also the PC based control system will send voltage values that are corrected to form a minimum starting voltage that may not be zero and is different for each fan.

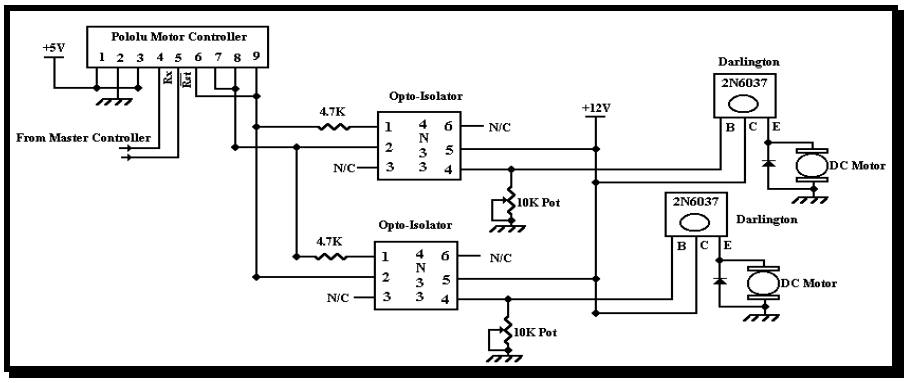

Figure 10: Motors Control System

Quadrature System

The quadrature controller has been described in a previous article and is exactly the same as described there. The master controller in this project will use the control lines (see Figure 11) to ask for the quadrature states transition count and then send that value to the PC controller. The value is then used to calculate the stations heading as shown in the *CalcStationResponse* subroutine (lines after **`SerBytesIn`** command)

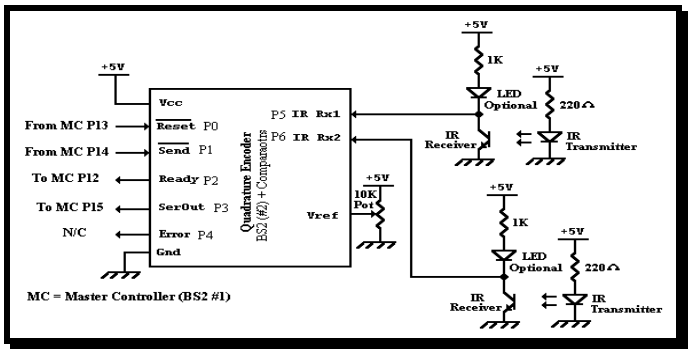

Figure 11: Quadrature Encoding System

Using The System

Figure 3 (last month) shows the user interface of the PC based Control Center. The user can select whether this is a simulated system or not. If it is simulated then all the control action (manual or automatic) will be used to control the simulated model. If it is not a simulated system then the control actions (automatic or manual) will be sent to the master controller on the physical model to control the fan speeds and to read the quadrature encoder from the scale model.

In the manual mode the user will command the direction and firing force of the fans. For ease of control the firing force can be only full or half force. This can be done using the arrow keys or the mouse buttons. For half force the *PageUp* and *PageDown* keys are used. During manual control the heading, speed and acceleration of the model are calculated and shown. Also the screen animation and graphs are maintained up to date.

In the automatic mode the system takes over and calculates the firing rate using the PID controller. The user commands the desired heading by clicking on the desired heading field and entering a heading value or by pressing the mouse on the arrows next to the field. This allows for gradual or step heading changes (see Figures 12 and 13). If during the automatic control of the physical model you (by hand) change the heading of the station the control system will act to restore the heading to the value indicated in the desired heading field (see Figure 14).

A field called ITAE is calculated during the manual as well as the automatic control (simulated or scale model). This value is the integral over time of the absolute value of the error multiplied by time and is an indicator of the effectiveness of the control. The larger the number the less effective the control has been. This number can be used while fine-tuning the PID parameters to compare which combination is best. Also it can be used to compare manual control against automatic control and also to compare two manual controllers against each other. So you can actually use the system as a kind of game to see who can control the station best.

Shortcomings And Improvements

As you can see from Figures 12 and 13 the scale model behaves in a very similar manner to the simulation model. The scale model has some jerkiness in the plot, but this is due to the fact that the model is far from being ideal. It is far from being frictionless. The wheel is not well balanced despite some pains to make it so. The support frame of the wheel and the wheel's imbalance also created areas where the wheel would prefer to settle and thus creating unsymmetrical forces that had to be overcome by the already unsymmetrical fans

The delays in starting (see Figure 13 seconds 0, 4.5 and 10) and too early stopping (see Figure 13 seconds 0.5, 5.5, and 11.5) of the wheel in the scale model are due to the fact that static friction is higher than dynamic friction. Also friction causes the station to stop quickly when forces are removed.

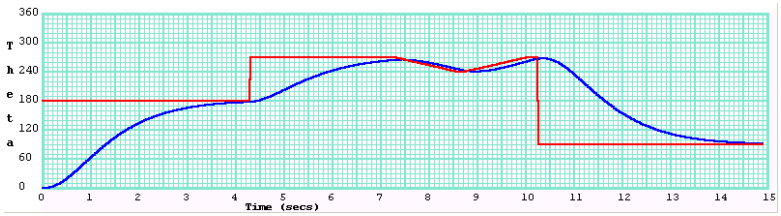

Figure 12: Automatic control of the simulated model station. Notice the step and gradual commanded heading changes.

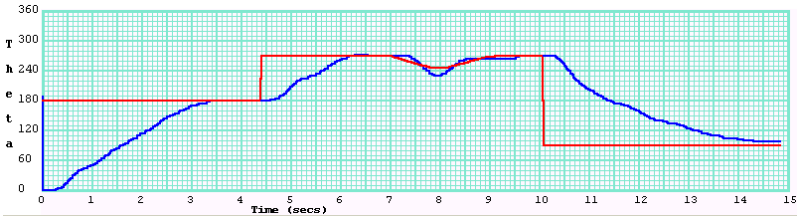

Figure 13: Automatic control of the scale model station. Notice the step and gradual commanded heading changes.

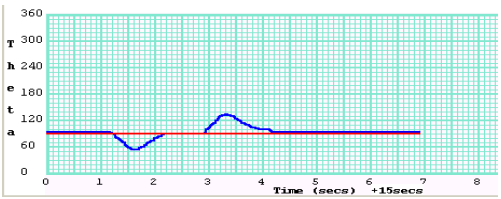

Figure 14: Disturbance rejection during automatic control of the scale model station.

The digital PID controller (*CalcPID* subroutine) works real well. Notice how on the simulated station the actual heading follows the gradually changing desired heading. Also on the scale model the following is quite acceptable given the delays due to friction. Also notice how in Figure 14 the disturbance was rejected quite nicely.

The physical properties (J, VCOF, and R) were calculated for the scale model. So the simulation model would simulate the scale model not a real full sized station. R was measured, and J was calculated by weighing the wheel and assuming that the wheel is a disc. VCOF was an educated guess.

Final Thoughts

Notice how the K_p, K_i and K_d values work well for both the simulated and physical models. These values were not what was calculated using the mathematical tools and formulas. The calculated values produced a system that responded as in Figure 5b (last month). The values shown in Figure 3 (last month) are close to the calculated values but have been changed by trial and error using the scale model until an acceptable response was obtained. The simulated model also responded well. Compare the responses in Figures 12 and 13 to the ideal response in Figure 5f (last month).

This project is quite an involved one. There are programs for the PC and for two separate BS2 controllers (master and quadrature). Also there is a lot of electronic circuitry. The Control Center system (PC) is a very involved program but was not really difficult to create. The principles of digital control boil down to adding and multiplying in the end. There is no real hard math.

The PC system is what you would expect from a professional system. There are ergonomic displays of the status of the system as well as ergonomic controls to allow the user to change the parameters and the state of the system in an intuitive manner.