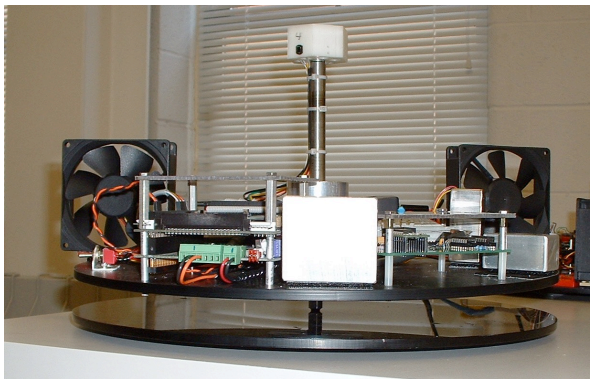
Aerospace Information Systems
Aero 552, Fall 2017

Issued: ***Wednesday, November 22, 2017***
Due (CTools & t1000 at 11:55pm):
Tuesday, December 12, 2017

Controlling the Tabletop Satellite (TableSat) Embedded System



TableSat is a 1-DOF satellite simulation platform driven by two computer fans that apply + and – torque, respectively. The primary objectives of this assignment are to write a program that uses the onboard rate gyro to control the rotation speed of TableSat, and to write a program that will store magnetometer and light (sun) sensor data for which you will develop calibrations to determine TableSat "heading". You will be working in groups of two to solve this assignment per the Google Drive spreadsheet. TableSat runs the QNX real-time operating system (www.qnx.com) on Diamond Systems PC/104-based data acquisition and computer systems, relying on the DSCUD driver software provided by Diamond Systems Corporation (www.diamondsystems.com) to interface with the hardware. (The DSCUD manual is available online at http://docs.diamondsystems.com/dscud/manual_Main+Page.html if you are interested.) We have two TableSat platforms available for this class, **TSat1** or TableSat I, and **TSat2** or TableSat II. Both use a Diamond Systems Athena processor.

You have been given as a reference separate programs for reading the gyros (`read_gyros`) and driving the fans (`command_fan`). You can find these programs on `t1000.engin.umich.edu`, under the `/home/public/tsat` subdirectory. As of Saturday (morning), you will all have accounts on **t1000.engin.umich.edu**. `t1000` also runs QNX which has a smaller footprint than Linux but a similar command-line interface. For this project `t1000` is required to compile your TableSat code since the embedded TableSat platforms do not have sufficient disk space for the compiler (they are old but have excellent data acquisition capability). Your default login password is "blah" on `t1000` – please change this by logging in and running the `passwd` command ASAP. Otherwise, you run the risk of other students and external hackers finding easy access to the machines and your files. As a native QNX platform, you have limited editor access on `t1000`. If you feel comfortable with `emacs -nw` (no window only) or `vi` you can use those on t1000; otherwise edit your code on another machine and transfer it to `t1000` for compilation. You can use `telnet` to login to t1000 and `ftp` to copy files to/from `t1000` from another machine. You can connect *from* any of the QNX machines (at the QNX prompt $) *to* a Linux machine with `ssh` to login or `scp` to copy files.

Each student team must submit to CTools a single report (`tablesat.pdf`) for this project on or before the submission deadline. Leave your code under your home directory on `t1000`. Make sure your report and code contain your name, and comment your code if there's anything unusual we might not

understand while grading. Below is machine information for TSat1 and TSat2. Each student will have access to both TableSats but I highly recommend you choose one and stick with it unless one is down (your gains may vary between machines). Don't forget to use the CTools chat room to coordinate Tsat queues!

TSat1 information:

To access TSat1, log into **maximilian.engin.umich.edu** (a Linux machine with 2 network cards) (use `ssh`). From maximilian, you can log into TSat1 as **192.168.1.100** (through a local wifi router network) (use telnet); always log into all machines with your user name; you will want to use root only to run your code on TSat (and then always stay in your home directory and delete extra data files after completing a data acquisition run).

TSat1's video (overhead) can be accessed via the web site **http://7of9.engin.umich.edu**. When you get to this site, you will need to enter username `tsatview` and password `viewer` to view the video. Click on the "single" tab at the top of the screen and try a lower-resolution format if the screen is too jumpy with the default resolution.

TSat2 information:

From maximilian (see above), you can also log into TSat2 as **192.168.1.131** (through a local wifi router network) (use telnet); always log into all machines with your user name; you will want to use root only to run your code on TSat (and then always stay in your home directory and delete extra data files after completing a data acquisition run).

TSat2's video (overhead) can be accessed via the web site **http://3of5.engin.umich.edu**. When you get to this site, you will need to enter username `tsatview` and password `viewer` to view the video. Click on the "single" tab at the top of the screen and try a lower-resolution format if the screen is too jumpy with the default resolution.

Please make sure you don't leave TableSat spinning, except when you're actively testing a program. A simple execution of the available program "command_fan" with a thrust level of 0 entered will turn off the fans. [Note that working versions of `command_fan` and `read_gyros` can be found directly on the TableSat disk in /home/public for each TSat platform – please copy the appropriate set of files to your home directory for your use.] As a final note of etiquette, please do not leave any data files on TableSat itself day-to-day. As an embedded system, the TableSats have a very limited hard disk size, and if we fill a disk, that TableSat will be unusable until we get the chance to clean the disk. Once you collect usable data, you can use `ftp` to move the data from TableSat to `maximilian`, where you can use `scp` to copy your data to any machine you'd like to use to plot your data. Once you have your data on maximilian, please delete (`rm`) the data files from TableSat.

Also, please note that only one executable can be run at a time on each TableSat platform. We highly recommend that when you are ready to test your program on your designated TableSat system, that you use the CTools chatroom to create and monitor a queue of users for TSat1 and TSat2, respectively, and check the process queue on the TSat machine with 'ps –ef'. Note that if a Tablesat process has been running for some time (over a couple of minutes), you are authorized to use the "kill" command *only

for student-created processes for this assignment!!!* (please be sure – don't kill processes you don't understand). The TableSat platforms will be online several hours every day from Saturday 3/19 until the assignment is due (Tues Mar. 29). Each day TableSat will be turned on at approximately 3pm unless otherwise noted by announcement and will stay on until the battery expires (probably around 10pm). If we have a chance to replace the battery mid-evening TSat may last longer, but no guarantees. Please complete the following tasks for this project and document your results in a formal report with sections for each problem. Your report should have a brief introduction to the project that provides an overview of the project goals and an outline. You should have one report section for each of the below tasks, describing in text, equations, and figures/tables your method, procedure to solve the problem, and results. Provide a brief conclusion summarizing your results and offering any suggestions you have to improve the project for the next class. We recommend you create a report shell now and populate content as you solve problems and collect data to avoid needing to write everything near the deadline.

1) **Rate Control:** Write a program `rate_control.cpp` that i) asks the user to input a desired TableSat rotation speed (in "gyro-Volt" units 0-5V), b) ramps to and maintains that speed for 30 seconds, and c) captures data during the "drift" ramp-down to near-zero velocity (gyro voltage at its 0 reading +/- 0.1 volts). Save time (since program start), thrust command, and rate gyro data in output data file `rate_data.txt` at 100Hz from when you issue the first non-zero fan command output (just after the user enters the desired speed) until TableSat ramps down to its "near-zero" velocity described above. Leave your working code in your home directory on `t1000`. We will be grading your code from there, not downloading it from CTools. Include in your report plots of time vs. fan command (in +/- Volts) and time vs. rotational rate (in gyro-Volts) for three different reference commands: 1 gyro-Volt, 3 gyro-Volts, and 4 gyro-Volts. Define the control law(s) you tested, the procedure by which you tuned gains, and the behavior observed in these plots in approximately one page of text not including your figures.

2) **Data Collection:** Write a program `daq.cpp` that slowly rotates TableSat through multiple revolutions using your rate controller. Capture time, rate gyro, magnetometer, and coarse sun sensor (CSS) data in data file `tsat_data.txt` at 200Hz. You will need to characterize orientation of the sensors based on collected. In your report, plot time vs. each magnetometer and CSS channel over a span of 2-3 platform revolutions. Compare and contrast different moving average filter window sizes (n) with respect to observed (white) noise and peak-to-peak magnitudes for representative magnetometer and CSS data channels; use this analysis to select an appropriate filter window for each data type.

3) **Gyro MKS Unit Calibration:** Create a linear calibration of the rate gyro from gyro-volts to degrees/second using unique data "patterns", e.g., passage by the "sun", from part 2). Specify your (linear) calibration (include a C code excerpt as well as a mathematical equation) for this conversion and plot the relationship between gyro voltage and calibrated degrees/sec.

4) **Magnetometer Calibration:** Specify a (nonlinear) calibration function (include your C code) relating magnetometer data to TableSat heading. Feel free to define "North" however you'd like – just be consistent. Your function can use "logic" and limited "data history" as you will not likely be able to define a well-behaved single calibration function relating a single channel of magnetometer data to a unique heading.

5) **CSS Calibration:** Specify a (nonlinear) calibration function (include your C code) relating CSS data to TableSat heading. Define the "Sun" as North. Note that you need not use the same North

reference for CSS Calibration as you have for the magnetometer, although if you would like to define a consistent North that is fine (of course).

6) **Pointing Control:** Select your "favorite" heading sensor type, CSS or magnetometer. Using your calibrations for rate gyro and your favorite heading sensor type (CSS or magnetometer), integrate the necessary data acquisition, calibration, and (PID) control code into code point.c that asks the user for a pointing direction input (in degrees from 0 – 359.999) and then moves TableSat to that heading. The grader will run this particular code to verify its functionality with different heading inputs. The grader will command North (0) first then reference the accuracy of your other headings relative to your choice of North. If your code does not work for all possible headings, indicate this in your report to guide the grader to test only values that will work well.

Example `ssh` login and other commands, working through maximilian (Tsat1 example only):

Open a terminal
ssh to your work computer
```
ssh -Y uniqname@maximilian.engin.umich.edu
```
Give your password for maximilian

Copy your code from maximilian to t1000 for compilation
```
ftp t1000.engin.umich.edu
```
Give your username (uniqname) and password for t1000
Copy the source code to t1000
```
put your_code.cpp
```
Log out of the ftp client
```
quit
```

Log in to t1000
```
telnet t1000.engin.umich.edu
```
Give your username (uniqname) and password for t1000
Compile your code
```
g++ your_code.cpp -o your_program
```
Copy your executable from t1000 to maximilian
```
scp your_program uniqname@maximilian.engin.umich.edu:~/your_program
```
Give your password for maximilian
Log out of t1000
```
exit
```

Copy your executable from maximilian to TSat1 for testing (you are currently logged in to maximilian)
```
ftp 192.168.1.100
```
Give your username (uniqname) and password for TSat1 (note: do NOT use root to access!)
Copy the executable to Tsat1
```
put your_program
```
Log out of ftp client (one of these two should work: exit, quit)
```
exit
```

Log in to TSat1 (you are currently logged in to maximilian)

```
telnet 192.168.1.100
```
Give your username (uniqname) and password for TSat1 (note: do NOT use root to access!)
(Check to make sure you are in your home directory, listed under `/home/your_uniqname`)
Elevate privileges to root / superuser
```
su
```
(Wait until no-one is using TSat1 – we recommend you use the chatroom on CTools to create a queue)
Test / Run executable program
```
./your_program
```
(Run program as many times as necessary)
(Watch the TSat1 camera while running to notice if things are working properly, see login instructions p.2)

When done, copy datafile(s) as necessary; from maximilian login, use ftp to get data file from Tsat:
```
ftp 192.168.1.100
```
Log into Tsat as yourself, not root, then "get" the data from Tsat to maximilian and quit/exit
```
get datafile1.txt
quit
```

Repeat other groups of steps above as necessary

Log out of maximilian
```
exit
```

Note: For the TSat2, the login address is `192.168.1.131.`

**Also a general note: t1000 doesn't have a working "gcc" command so please use "g++" as your only compiler, even if you choose to work with pure C code as is provided in the examples.