A Method for Using a TX2i for On-Board Computer via Serial Connection

Multi-view Onboard Computational Imager (MOCI)

S. Godfrey Hendrix[*1], Eric Miller[#1], David L. Cotten[^1,2], Deepak Mishra[1,2]

[*]sgh33519@uga.edu, [#]ericnmiller99@gmail.com, [^]dcotte1@uga.edu,

[1]Small Satellite Research Laboratory, University of Georgia,

[2]Center for Geospatial Research, University of Georgia

Abstract

Current generation CubeSats are very computationally light. They typically only have enough computational power to run basic flight software and rely on computers on the ground to do the bulk of the data processing. The Nvidia Jetson platform presents an opportunity to execute high performance computing tasks on orbit rather than relying on these ground-based systems. This allows for two main improvements to current mission capabilities:

1. Significant decrease for data downlink requirements
2. The possibility of real-time data processing

Traditional CubeSat components operate using limited serial-type connections for data transfer and control as well as not using standard commercial connectors such as USB or Ethernet, thus making control of a Jetson via a direct shell infeasible. We propose a system architecture that effectively abstracts all of the primary functions of the Jetson platform to a simple serial connection such that it can be controlled fully via a space-grade flight computer while still providing the aforementioned HPC advantages.

System Architecture

The control program created by this project centers around a master-slave relationship between the On-Board Computer (OBC) and Nvidia Jetson TX2i, respectively. Under this system, the OBC can invoke a predetermined list of commands via the serial interface on the TX2i, each of which has predetermined arguments and specified acknowledgment systems. These commands are received and processed via a Python based program running on the TX2i. Such commands range from a simple ping, where the OBC will check that serial communication is still running, to “setProgrammaticState”, which instructs the TX2i to transition into one of its operational states. The system also has the ability to command bulk file transfers to and from the Jetson. Finally, the system includes commands that can set the power state of the TX2i via hardware based switches. The programmatic states are the functions that drive the bulk of the HPC functionality and are defined within separate python scripts. These are executed by name and are commanded via the main program through interprocess communication.

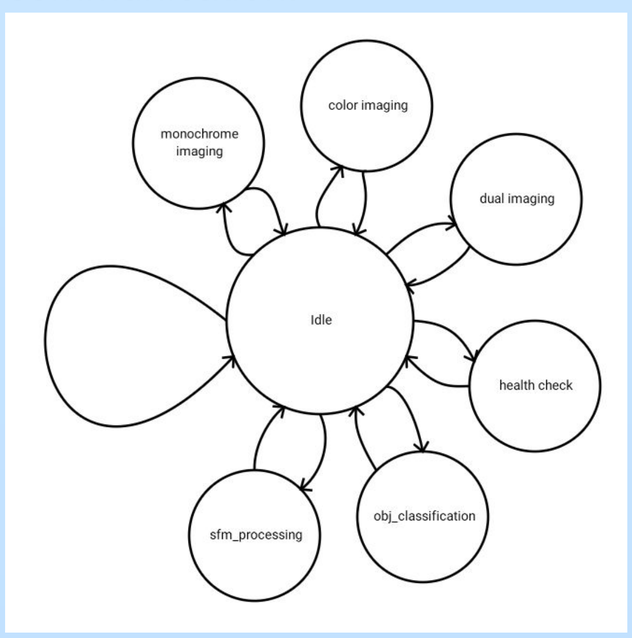

Figure 1: Illustrates the different programmatic states present for the MOCI mission

Current Progress

Version 1.0 of the control system is almost complete at the time of creating this poster. Currently, the OBC has the ability to do the following:

1. Ping TX2i
2. Query telemetry
3. Set programmatic state of TX2i
4. Power off TX2i

The primary area of work at the moment is to allow data transfer. The TX2i has demonstrated the ability to send and receive data via the serial interface; however, the flight software running on the OBC currently is unable to store the incoming byte stream.

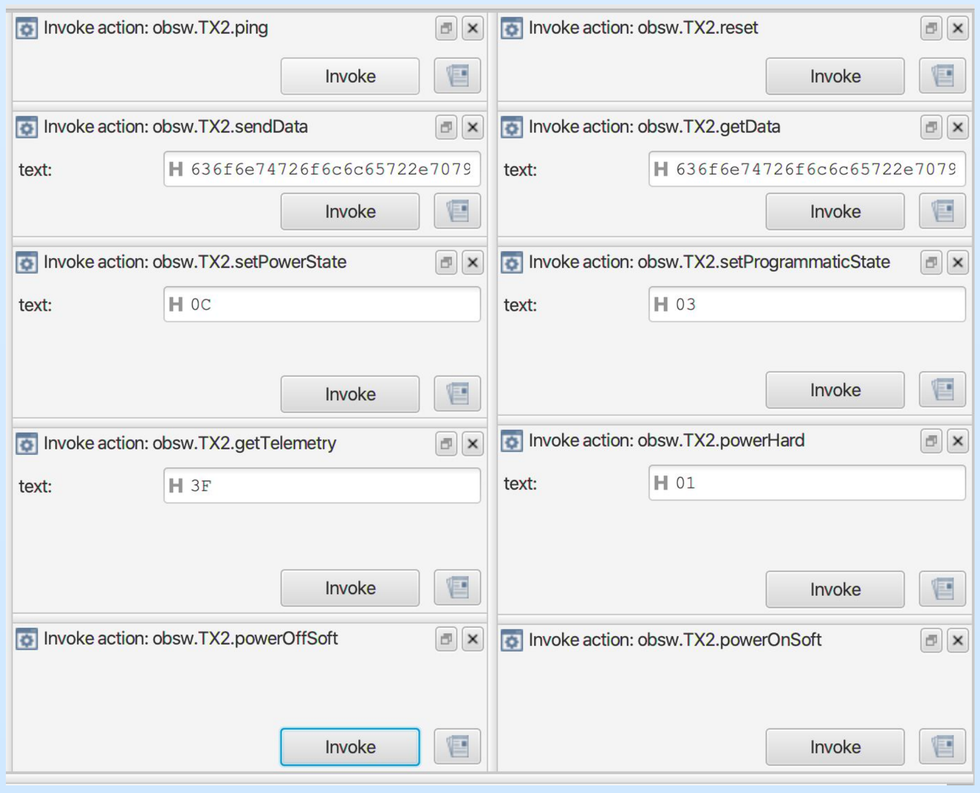

Figure 2: Illustrates the system view from the OBC perspective

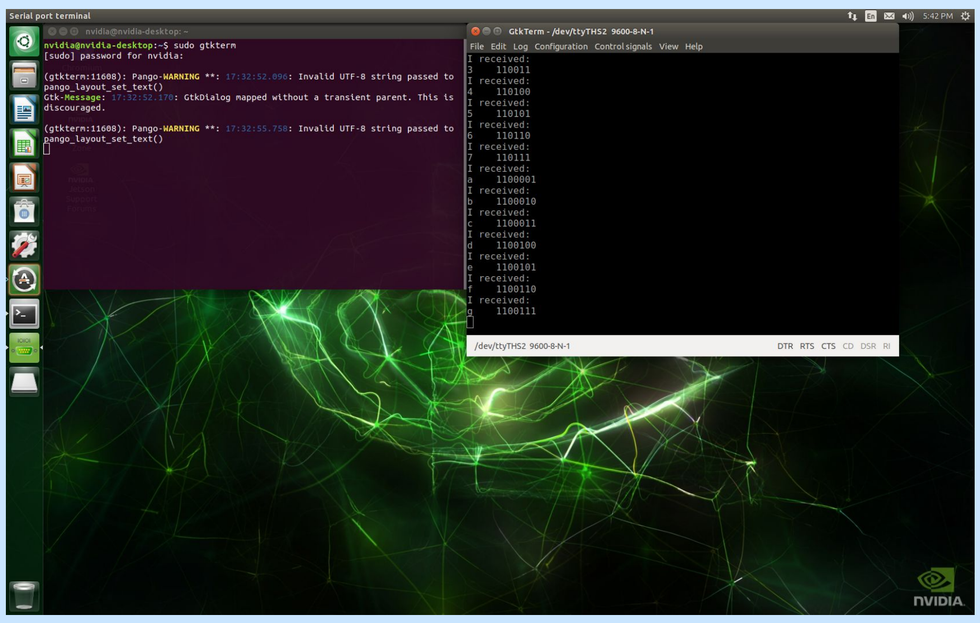

Figure 3: Illustrates the TX2i receiving raw serial data

Future Work

After finalizing version 1.0 of the control program, work will begin on implementing all the programmatic states required by the MOCI mission. These will be designed to be as small as possible in terms of algorithmic complexity. This has two major benefits:

1. Reduces execution latency
2. Allows more frequent checkpointing of pipelines

Beyond the MOCI mission, this control system will be applied to future missions running on the SpaceGPU project currently in parallel development at the SSRL.