In-Loop Simulation of Attitude Control of a Nanosatellite

Vatsal Jignesh Badami
Birla Institute of Technology and Science, Pilani
Rajasthan, India - 333031
vatsaljb.007@gmail.com

Tushar Goyal
Birla Institute of Technology and Science, Pilani
Rajasthan, India - 333031
tushargoyal21@gmail.com

Shubham Sharma
Birla Institute of Technology and Science, Pilani
Rajasthan, India - 333031
shubhamsh695@gmail.com

Saurabh Manish Raje
Birla Institute of Technology and Science, Pilani
Rajasthan, India - 333031
f2015045@pilani.bits-pilani.ac.in

Kushagra Aggarwal
Birla Institute of Technology and Science, Pilani
Rajasthan, India - 333031
kushagra.aggarwal01@gmail.com

Abstract*—Team Anant is a group of undergraduate students at the Birla Institute of Technology and Science, Pilani, working on the design and development of a 3U CubeSat. The proposed 3U CubeSat features a hyperspectral camera as its payload which requires stringent control maneuvers for imaging and generates a large amount of data. The On-Board Computer features a Zynq-7000 SoC consisting of two ARM Cortex A9 cores (PS) and a field programmable gate array (PL) on the same silicon. The PL is responsible for implementing a hyperspectral image compression algorithm while the PS is responsible for housekeeping and running attitude control and determination algorithms among other tasks. Verification of the design is a critical step in the development of a nanosatellite. Conventional in-loop simulations use testbeds as a means of simulating the space environment to verify the functional integrity of developed modules. In this paper, a simulation scheme is discussed which replaces some of the hardware test beds with satellite models running on a computer which allows reduced dependence on the attitude and control subsystem (ADCS). The simulation is done in order to test the functioning of the attitude control algorithm, with as much resemblance to the actual nanosatellite as possible and with minimum budget. The loop comprises of two Arduinos, Zedboard (an evaluation board for Zynq), DC-DC converter and satellite models running on a computer. One of the Arduinos is used to emulate the magnetometer which senses the magnetic field in space. The satellite simulator running on the computer sends the information to this Arduino, which is interfaced to the Zedboard via an I2C interface. The B-dot algorithm running on the Zedboard is used to calculate the required actuation and generates a PWM wave accordingly. This PWM wave is given as an input to a DC-DC converter which gives an analog voltage based on the width of the PWM wave. The other Arduino is used to emulate the actuator which takes an input from the converter circuit. This voltage is converted to current and sent to the actuator model running on the computer to determine the generated torque and hence its effect on the satellite is calculated, thus completing the feedback loop. Although the paper focuses on detumbling using B-dot, it can be generalized to the simulation of any subsystem of the satellite and can be used as a first step in the functional verification.

TABLE OF CONTENTS

1. INTRODUCTION…….. 1
2. SATELLITE ARCHITECTURE……..2
3. B-DOT ALGORITHM OVERVIEW……3
4. INTERACTION BETWEEN OBC AND ADCS..3
5. OBC CONTROL AND DATAFLOW ARCHITECTURE……..4
6. SIMULATION ARCHITECTURE……...5
7. SIMULATION COMPONENTS…….. 5
8. SIMULATION DATAFLOW……... 7
9. FUTURE WORK……8
APPENDICES……8
A. A88
B. 88
C. MATLAB PARTIAL LOOP……..... 9
ACKNOWLEDGEMENTS……...9
REFERENCES……... 9
BIOGRAPHY……10

1. INTRODUCTION

The miniaturization of the satellites in the form of CubeSat platform provides a low cost alternative for space agencies to test new technologies as well as students and amateur scientists to gather experience in the field of aerospace engineering. CubeSats have gained much attention in the past few years due to their ease of development, lesser lead time and manufacturing cost in comparison to bigger satellites.

Team Anant, a student satellite team at BITS, Pilani is working on the development of a 3U CubeSat with a hyperspectral imager. The satellite can be classified into six

978-1-5386-6854-2/19/$31.00 ©2019 IEEE

subsystems viz. Onboard Computer (OBC), Telemetry & Telecommand (TTC), Attitude Determination & Control System (ADCS), Electrical Power System (EPS), Payload, and Structure and Thermal Systems **[2]**.

The testing and verification of onboard software and satellite components for CubeSats still heavily relies on pre-existing test beds which have high procurement costs, often limiting the development process for budget restricted student teams **[1]**. Alternatively, entirely software based testing can be carried out having its own limitations like integration testing of various hardware drivers and interfaces becomes difficult. Further, the latency added by the interfaces used, especially if many sensors are connected on the same bus, becomes harder to measure in purely software based testing.

Satellite detumbling is a critical process that requires real-time capabilities and high performance from the OBC. This process decreases the angular velocity of the satellite to below a certain threshold. It forms an important part of any CubeSat mission requiring active control because a tumbling satellite has diminished power generation capability and increased the latency in pointing the satellite for certain tasks like data downlinking and remote sensing. In a centralized architecture (as the one proposed by the authors), it is possible that the ADCS control algorithms might be implemented on the OBC. This results in a huge interdependency between the OBC and ADCS. Given the importance of a low latency, high-throughput response from the processor during the control process (to achieve a predefined time to convergence), a rigorous methodology for a ground-test and evaluation is mandated. Conventional testing procedures employ hardware test beds comprising of air bearing and helmholtz-cage based magnetic field simulators **[4]**. This paper presents a novel hybrid hardware-software approach devised by the authors for testing the control algorithms that are employed onboard. This method allows us to test the latency of the processor and the interfaces used, however, it does not require building an actual hardware to simulate the space environment and the sensors/actuators.

2. SATELLITE ARCHITECTURE

Our satellite's functioning has been divided into six subsystems or modules. The functioning of these modules is distributed in a mutually exclusive and exhaustive way, covering all the aspects of satellite's mission and maintenance **[2]**. The functioning of these subsystems is interdependent. A brief description of the subsystems and their tasks is given below:

Onboard Computer (OBC)

The OBC controls, coordinates and monitors the functioning of the other subsystems of the satellite. Based on the satellite's status the OBC is responsible for switching its modes of operation **[3].** It does so by implementing a software called the Flightplan on the Processing System (PS) which is part of the Zynq-7000 System on Chip (SoC) comprising of a dual-core Cortex A9 processor. The Zynq-7000 SoC also contains an FPGA called the Programmable Logic (PL) on which a compression algorithm (CCSDS-123.0-B-1) is implemented, which is used to compress the hyperspectral image obtained from the payload. This is essential in making the downlink of the image feasible. As part of the control and coordination, the OBC processor is interfaced with most of the sensors and actuators on board via I2C, SPI and GPIO interfaces. The OBC PS has a linux based operating system called Petalinux running on it, hence all hardware level interactions happen via Linux kernel drivers. In each mode there are various tasks which might have to be implemented, these tasks are often executed as processes under a multi-processing paradigm in according to the kernel scheduler.

Telemetry & Telecommand (TTC)

The TTC subsystem is responsible for encoding the data packets using AX.25 protocol and downlinking them during a ground station pass. In addition to the data, the TTC periodically also sends small packets of information containing basic parameter of the satellite like its battery condition and call sign, this is called the beacon. The former requires larger power and uses the GMSK communication modulation technique, while the latter requires very low power and uses On-Off keying.

Attitude Determination & Control System (ADCS)

The ADCS is responsible for for controlling the attitude of the satellite and pointing it in the right direction during payload execution and data downlinking. It is also responsible for implementing an orbit propagator to predict the orbit of the satellite. A major part of controlling the satellite's attitude is the B-Dot detumbling algorithm which controls the angular velocity of the satellite and keeps it below a certain threshold. All the ADCS algorithms are executed on the PS of the OBC. Various sensors are a part of the ADCS to sense different parameters about the satellite. These are:

1) Three-Axis Inertial Measurement Unit: Used to sense the angular velocity and acceleration of the satellite.
2) Three-Axis Magnetometer: To sense the Earth's magnetic field around the satellite.
3) Sun Sensors: To sense the position of the Sun as seen from the satellite.

Based on the output of these algorithms, the amount of actuation required and the necessary action to provide that actuation is determined. The actuators, interfaced with the OBC, in our satellite are:

1) Magnetorquers: A combination of two torquer rods and a torquer coil, providing actuation in all three principal axes of the satellite. The torquers use the Earth's magnetic field to control the orientation of the satellite.
2) Reaction Wheels: Three individual wheels in the three principal axes of the satellite. The reaction wheels use their inertia to impact a control torque on the satellite.

The B-Dot requires the readings from magnetometer and outputs the current to drive the magnetorquers.

Electrical Power System (EPS)

EPS forms the lifeline of the satellite, and is responsible for power generation, distribution and storage. It collects housekeeping data from voltage, current, and temperature sensors and sends the data to OBC upon request. It implements the Maximum Power Point Tracking (MPPT) to optimise power generation from the solar panels and also monitors health of other subsystems using Over Charge Protection Circuit (OCPC).

Payload

Our satellite features a hyperspectral camera as its primary payload. It is interfaced with OBC using Universal Serial Bus (USB) which is used to transmit control signals from OBC to the camera and receive the image. The image taking constitutes the most crucial part of the mission and requires stringent attitude control.

Structure & Thermal System

The elements of this subsystem constitute the satellite structure and various thermal control mechanisms distributed over the body of the satellite. Electronic components work in a specific temperature range and often a temperature outside these ranges causes a permanent damage to them. This subsystem's role is vital in designing the structure to enhance satellite's longevity.

3. B-DOT ALGORITHM OVERVIEW

The B-dot algorithm is used to control the satellite's angular velocity due to its ease of implementation and easily measurable inputs. The on-board magnetometer measures the magnetic field vector in the body frame of the satellite. This magnetic field changes over time due to two reasons. The first is due to the satellite's revolution around the earth which has a varying magnetic field around it. The second is due to the changing orientation of the satellite about its centre of mass. The rate of change of the former depends on the orbital elements of the satellite. The rate of change of the latter then depends on the rate at which the satellite is rotating about its own axis, that is, the angular velocity of the satellite.

The change in magnetic field due to the satellite's revolution around the earth is typically very small. Hence, the change in magnetic field measured by the onboard sensors themselves provide a good indicator for the angular rotation. When the sampling time between two measurements is taken into account and the change in magnetic field is divided by it, we get a good indicator for the angular velocity. Hence, $\dot{B}$ becomes a good indicator for the angular velocity of the satellite, a small value of the B-dot means that the satellite is rotating slowly, and a large value means it is rotating fast.

Based on this observation the control mechanism takes two consecutive readings of the magnetometer and then divides it by the sampling time to get B-dot. This B-dot is then multiplied by a constant value **[5]** to obtain the necessary opposing torque required to counter the angular velocity. Based on the properties of the actuator (magnetorquer in our case), the torque value can be used to determine the current that needs to be provided to the Magnetorquers. This process of sampling the magnetic field and consequently calculating the required torque and supplying the necessary current is performed iteratively till the angular velocity goes below a pre-decided critical level.

4. INTERACTION BETWEEN ADCS AND OBC WHILE RUNNING B-DOT

As discussed earlier the B-Dot algorithm is running, like all other ADCS algorithms, is getting executed on the OBCs processing system. The purpose of the B-Dot algorithm is to detumble the satellite, and like any other control algorithm it is implemented continuously, with each iterations separated by a regular interval of time. In normal modes of operation, i.e. when the satellite is not tumbling, B-Dot is always

running in the background at a lower frequency. However, when the angular velocity drops below a certain critical threshold the entire focus of the OBC shifts towards implementing iterations of the B-Dot algorithm. When in this state, the satellite is said to be in detumbling mode, in which iterations of the algorithm are performed at a high frequency. In one iteration (say k^{th}) of the B-dot algorithm the following interactions need to take place between the OBC processing system and the ADCS components:

1) The magnetic field values in x,y and z directions i.e. $B_x(k)$, $B_y(k)$ and $B_z(k)$ are obtained from the magnetometer in iteration k. This is done via the I2C bus which is used for interfacing between the PS and the magnetometer. As mentioned in the explanation about OBC, the interaction takes place via the I2C driver subsystem which is part of the Linux kernel.
2) Based on values obtained in the previous iteration i.e. $B_x(k-1)$, $B_y(k-1)$ and $B_z(k-1)$ and the the samping time between two continuous iterations, the value of B-dot is calculated for all three axes.
3) The value of B-dot in all three axes is multiplied by the constants of the three axes which then gives us the required counter torque to be provided along each axis.
4) Based on known information about the magnetorquers and the torque along each axis we are able to calculate the required current value to be provided to the magnetorquers in the x, y and z direction respectively.
5) The zedboard has a maximum output current of a few milli ampere at most but the current to be provided will be in the range of amperes. To counter this, corresponding to each current value a PWM wave is generated, where the width is depends on the current value. Each PWM wave is fed into a separate driver circuit which is then able to provide the necessary current to the corresponding magnetorquer with help from the EPS.

5. OBC CONTROL AND DATA FLOW ARCHITECTURE

The Linux operating system is responsible for managing hardware resources and their allocation to software applications **[Linux Ref]**. It is bifurcated into two parts- user space and kernel space. The Linux kernel space comprises of subsystems, each subsystem managing a particular family of hardware and acting as an intermediate layer between the higher level user space applications and hardware modules. These subsystems are composed of various drivers which are a software service managing the hardware directly. The device drivers are organised as controller and slave drivers for a hardware bus. The controller drivers directly issue a call to the bus controllers which can communicate with the slave devices through the bus protocol. Each slave device is represented by a driver which in turn generates requests to the controller driver for any communication with the device. The hardware can only be directly accessed by these drivers such that any user space application requiring access to the hardware generates a request called trap to the kernel. Each trap is associated with a particular code in the kernel space. Whenever the kernels receives a trap, it transfers the flow of control to the associated device driver. This device driver based on the privilege level of the user space application (i.e. the permission to use a particular device) can respond to this request in two ways- if the user space application doesn't have the permission for requested access, the driver raises an exception to the processor which results in the termination of the user space application. In case the user space has permission for the access, the device driver generates a request to the controller driver which translates this software call to hardware. This procedure of translating software call to hardware introduces a latency between the request by the user space application and actual hardware operation.

The user space applications run in the context of an abstraction called process. Each process runs in its own allocated memory. The kernel, through a service called scheduler, is responsible for distributing the CPU time to each process. This distribution of CPU time is called scheduling of the processes. The processes do not run in an atomic context and could be interrupted in between their execution by the scheduler, which can then allocate the CPU to another process before current process's completion. In presence of multiple processes, the scheduler can prolong the time in which a process completes its task, thus inducing a latency.

The onboard software is divided into various processes with each process handling a specific task. The B-dot control algorithm which is one of these processes, requires access to the magnetometers through the I2C bus and generates a PWM signal through a GPIO. Its execution thus incurs delay introduced by the scheduler and also the delay due to hardware access through the I2C and GPIO controller via their drivers. The effect of this latency is an important part

of this simulation method to analyse the efficacy of the B-dot control mechanism.

6. SIMULATION ARCHITECTURE

The verification and validation of functional integrity of the onboard hardware and software is crucial since they have to operate without any human intervention to correct them. Hence procedures have to be developed to assist in the design process of the satellites, with the objective of assessing the correctness of intended operation. The latency of response by the system to an external trigger is an important parameter which needs to be verified before launch. In our system our primary purpose is to test the interfaces that the Zedboard uses to interact with the ADCS components and the latency that gets introduced due to the bus speed and multi-processing system in place. With this purpose in mind, we don't make any changes to the code running on the Zedboard, this means that the flow from the software application layer running on top of the OS via the driver layer and the hardware controller and bus are all as they would be on the actual satellite. However, we are posed with the following problems for the other parts of the system:

1) On board the satellite the magnetometers provide the magnetic field values which are as expected in the actual orbit.
2) On the ground these values are only available using simulations of orbit propagators run on softwares like MATLAB.
3) But it is not sufficient to have the values and supply them to the Zedboard, it is also important to use the right interface so that the Zedboard system needs not be changed.

The simulation setup described below helps us solve these problems simultaneously.

7. SIMULATION COMPONENTS

The setup consists of a computer running MATLAB simulations of the orbit on it, two Arduinos, a ZedBoard and a DC-DC converter. The Arduinos and ZedBoard are connected through serial ports to the computer. One Arduino has an I2C connection with the ZedBoard to replicate the connection between magnetometer and microprocessor on the satellite. The output PWM wave of the ZedBoard is sent as input to the DC-DC converter via GPIO pins and the output of the DC-DC converter is connected to the analog input pins of second arduino.

The following paragraphs indicate the role of various components used in the simulation:

Computer Running MATLAB

A detailed MATLAB model has been developed to simulate the complete kinematic and dynamic working of the satellite once deployed in orbit. It consists of ten modules, where each module models various aspects of the satellite when in orbit like forces acting on it, rotation of the satellite around its centre of mass, path of satellite in space, etc.
During each iteration, some of the state parameters which are calculated are:

- Position vector
- Velocity vector
- Latitude and longitude
- Angle rotated by satellite since start of simulation
- Angular velocity
- Moment of Inertia
- Torque acting on the satellite
- Magnetic field at the position vector
- Control torque to be produced

The simulation also includes a model of the magnetorquer, upon receiving the value of current that is given as input to the magnetorquer, the model is able to give the value of the corresponding torque which gets generated. This torque then gets used to calculate the new state of the satellite determined by the state parameters given above.

Arduino 1

The sole function of Arduino 1 is to provide an I2C interface with the ZedBoard and act as I2C slave, thereby modelling the magnetometer. Whenever ZedBoard requests for data via I2C bus, the Arduino 1 writes a byte to MATLAB to notify that it has to send data to Arduino 1. This is done via a serial port interface. The MATLAB simulation then obtains the most recent values of magnetic field in the three directions and sends it to Arduino 1 via the serial interface. Arduino 1 then writes these values to the ZedBoard via the I2C interface.

ZedBoard

The ZedBoard is used as it would onboard the satellite. That is, it also acts as an I2C master for Arduino 1 and requests magnetic field data from it using the actual I2C driver used to interface with the magnetometer. After receiving data from the Arduino 1, the ZedBoard first converts the data from bytes to float. It then runs one iteration of the B-Dot algorithm and outputs the PWM wave required to generate the current value.

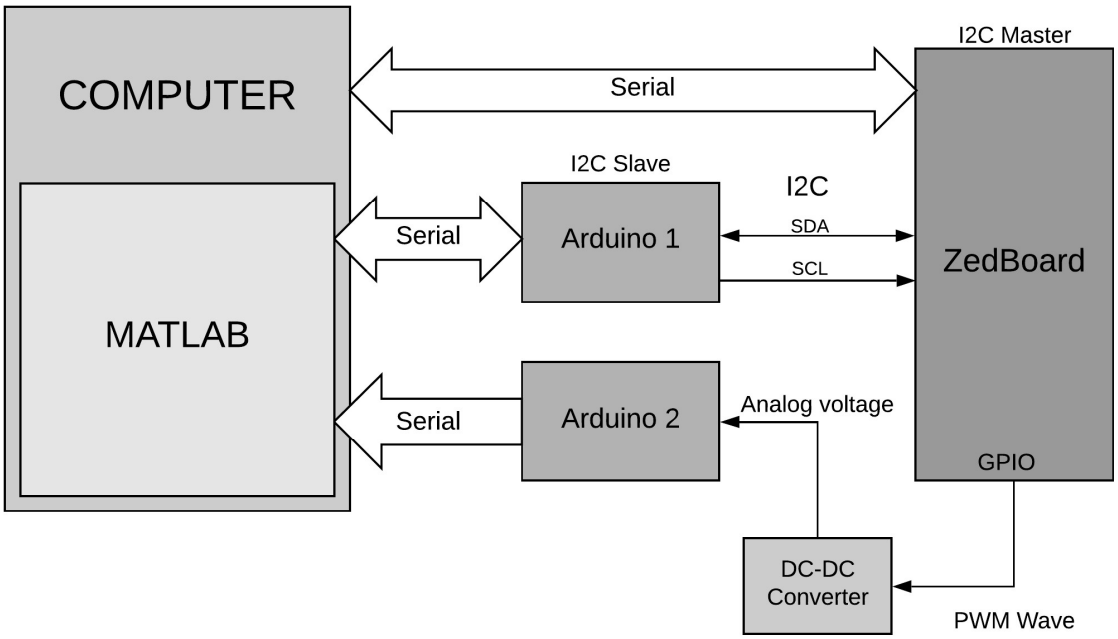


Figure 1. Block Diagram of Simulation setup

DC-DC converter

Arduinos cannot take in large values of current, so to make the Arduino 2 understand the value of the current in order to simulate the magnetorquer, we have used a buck DC-DC converter. One input of the DC-DC converter is fixed at 3.3 V. The MOSFET switch is powered by the PWM generated by the zedboard. As per the standard operation of the DC-DC converter, it produces an output voltage of $V_o = 3.3D$, where D is the width of the PWM wave between 0 and 1. This analog voltage can then be sensed by Arduino 2.

Arduino 2

This arduino models the magnetorquers. The analog pins therein are connected to the output of the DC-DC converter and the serial port is connected to the computer. It reads the analog signal and converts it back to the PWM width. Based on the PWM width it is able to calculate the current that the onboard driver circuit would have generated. Upon making sure that Arduino 1 has received the previous values, Arduino 2 sends the current value to MATLAB via the serial port. It keeps on averaging the values incoming from analog signal until they are to be sent to MATLAB.

Figure 1 describes the simulation setup devised in order to test the working of B-Dot algorithm (along one axis) on the OBC PS using actual interfaces as in the satellite.

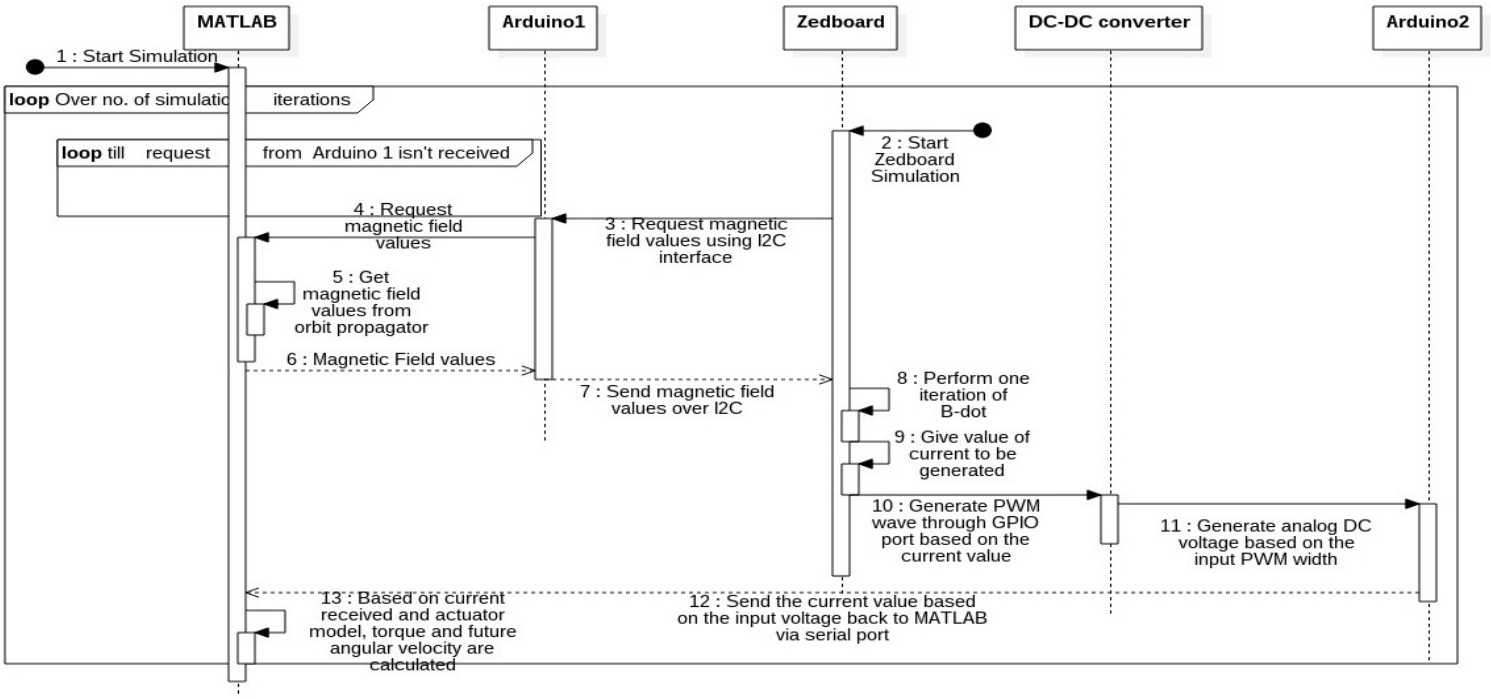


Figure 2. UML diagram depicting the flow of simulation

8. SIMULATION DATAFLOW

The sequence diagram in figure 2 tells us flow of control and data in the simulation system:

1. The Satellite Simulator starts running in MATLAB. It first initializes the various state parameters. Using a MATLAB function, the simulator loops till data is available on the serial port.
2. Meanwhile the ZedBoard also starts running its code by initializing the flightplan.
3. After initializing, the first section of the code running on the zedboard generates a request for the magnetometer values via I2C.
4. Upon receiving the request, Arduino 1 notifies the Satellite Simulator that it needs the magnetic field value, by writing a byte to the serial interface between itself and the computer.
5. Upon receiving the signal from Arduino 1, the Satellite Simulator exits the loop shown in point 1, obtains the magnetic field values from the orbit propagator, converts them to an appropriate data type.
6. The magnetic field value is sent to Arduino 1 via the serial port. After sending the magnetic field value to Arduino 1 a pause of 0.1 second is introduced, this is expected to be enough time for the voltage at the output of the DC-DC converter to get updated. After the pause a signal is sent to Arduino 2 telling it to send the analog value it reads to the computer via the serial port. In those 0.1 second steps 7 - 12 are expected to occur.
7. Arduino 1 then writes the value to ZedBoard via the I2C interface, which had been implicitly waiting for a response from its I2C slave. (due to the blocking nature of IO calls)
8. On receiving the value, the ZedBoard runs one iteration of the B-Dot algorithm, this basically involves taking the difference between the value of magnetic field received now and that received in the previous iteration which had been stored. The difference is then divided by the sampling time which is one second. The value of B-dot is then multiplied by a predetermined fixed constant **[5]** to find the torque value.
9. The torque value is used along with the magnetorquer model to generate the value of current to be supplied to the magnetorquer.
10. A PWM wave based on the value of current is then generated by the ZedBoard through the GPIO port.
11. The PWM wave is given as input to a DC-DC converter which generates an output voltage V_o= 3.3D, where D is the the width of PWM.
12. Arduino 2 reads this analog value, based on which it is able to determine the width of the PWM wave. Based on this width it is able to determine the current value which would have been given by the

driver circuit onboard. That current value is sent as input to the computer via the serial port to be used in the MATLAB code.

13. The actuator model in the MATLAB code uses this current value to determine the value of torque which would get generated. Using the value of torque and a sampling time of 1 second the new values of position, velocity, magnetic field, etc. are calculated and the simulator returns to the beginning of the loop, spin waiting for Arduino_1 to send signal to send the new magnetic field value.

9. FUTURE WORK

To test the latency introduced by multi-processing paradigm the different functionalities/modes of operation of the satellite which form part of the Flightplan can be integrated along with this simulation. Further, instead of taking the constant sampling time of 1 second, we can make the sampling time variable by measuring the time between two consecutive iterations, allowing us to get more accurate results. To further inculcate real hardware we could add more Arduinos to simulate various sensors to be interfaced. More complicated tasks like the processes involved in uplinking can also be simulated by using one of the Arduinos as a receiver emulator.**[2]**

APPENDICES

A. ARDUINO 1 CODE

```
#include <Wire.h>
#define SLAVE_ADDRESS 0x1E

byte B[4]={0};

void setup() {
  Serial.begin(9600);
  Wire.begin(SLAVE_ADDRESS);
}

void loop() {
    Wire.onRequest(requestEvent);
}

void requestEvent()
{
  Serial.write(10);

  if(Serial.available()>0)
  {
    Serial.readBytes(B,4);
  }
   Wire.write(B,4);
}
```

B. ARDUINO 2 CODE

```
byte* current_bytes;
float voltage = 0.0;
float current = 0.0;

#define RESISTANCE0 344.3234
int count = 1, number = 1;

void setup() {
  Serial.begin(9600);
}

void loop() {
  if (Serial.read()==5)
  {
    voltage = voltage / number;
    current = voltage / RESISTANCE0;

    current_bytes = (byte *)&current;
    Serial.write(current_bytes, 4);

    voltage = 0.0;
  }
  else
  {
    voltage = voltage + (analogRead(A0)
- analogRead(A1)) * 5.0 / 1023.0;
    number++;
  }
  count++;
}
```

C. MATLAB PARTIAL LOOP

```
while arduino.BytesAvailable == 0
end

flushinput(arduino1);

Field_X =
single(Mag_Field_Body(1,Counter))

MField_X = typecast(Field_X,'uint8');

fwrite(arduino1,MField_X,'uint8');

pause(0.1);

fwrite(arduino2,5);

pause(0.5);

temp = fread(arduino2,4,'uint8');

flushinput(arduino2);

current=typecast(uint8(temp),'single')

Current_Magnetorquer_BDot(:,Counter) =
0;

Current_Magnetorquer_BDot(1,Counter) =
current;
```

ACKNOWLEDGEMENTS

The authors thank the following:

- Dr. Kaushar Vaidya and student members of Team Anant, the nanosatellite team of BITS Pilani, for their constant support throughout.
- The administration of BITS Pilani for providing the funds necessary to procure components for the setup.

REFERENCES

[1] Ivanov, D.S., Karpenko, S.O., Ovchinnikov, M.Y. et al. J. Comput. Syst. Sci. Int. (2012) 51: 106. https://doi.org/10.1134/S1064230711060104

[2] Mr. Rutwik Narendra Jain, et. al., "Modes of Operation for a 3U CubeSat with Hyperspectral Imaging Payload", 69th International Astronautical Congress, 2018.

[3] Mr. Sourabh Raje, Mr. Abhishek Goel et. al., "Development of On Board Computer for a Nanosatellite".

[4] Goyal, Tushar. (2017). Design and Development of a three-axis controlled helmholtz cage as an in-house Magnetic Field Simulator for Cubesats.

[5] J. F. Kasper and V. Kasper, Attitude Determination and Control System for AAUSat3 (Master Thesis), Aalborg University , 2010.

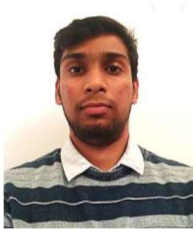
BIOGRAPHY

Vatsal Jignesh Badami *is pursuing B.E. (Hons) in Computer Science from Birla Institute of Technology and Science, Pilani, India. He has been a part of the On Board Computer subsystem (OBC) of Team Anant since January 2017 and is currently the subsystem lead of OBC. His research interest lies in the field of Computer Vision.*

Tushar Goyal *is set to receive a B.E. (Hons.) in Electrical and Electronics Engineering from Birla Institute of Technology and Science, Pilani, India in 2019. He has been with Team Anant since January 2016. He has been the student coordinator for the team since March 2017, managing the administrative and financial aspects of the team. Prior to that, he was the subsystem lead for Attitude Determination and Control System (ADCS) of the team. The ADCS is responsible for developing algorithms to determine the orientation of the satellite and then control it according to the requirements. His interests lies in Astrodynamics and Mission Analysis and he aspires to pursue research, in the same field, in the future.*

Shubham Sharma *has graduated with a B. Tech in Electronics and Instrumentation and an MSc. in Mathematics from Birla Institute of Technology And Science, Pilani. He has been a part of Team Anant in various capacities since 2013. He currently works for Toshiba Software India Pvt. Ltd, Bangalore.*

Saurabh Manish Raje*, a senior at BITS Pilani (majoring in computer science) has been a member of Team Anant for over two years. He has served as the Head of on-board computer subsystem and played a key role in the design and implementation of the software architecture of the satellite, published at the IAC-17. His area of research is High Performance Computing, with a focus on HPC for deep learning. He currently works as a research assistant at INRIA in Grenoble, France on the Rust language. He enjoys various physical pursuits such as hiking and powerlifting.*



Kushagra Aggarwal, *a student of Birla Institute of Technology and Science, Pilani, India is pursuing B.E. (Hons.) in Electronics and Instrumentation Engineering. He was a part of Team Anant from August 2016 to May 2018 contributing to the development of Onboard Computer subsystem. He also served as the System Engineer for the team and played an important role in developing the modes of operation of the satellite. He was a Research Intern for Summer 2018 at University of Tokyo and is currently a research scholar at IRISA, France. His interest lies in the field of device physics and quantum computing.*