A Generic Subsystem Simulator - Applied to the Antenna Subsystem of the MIST Student Satellite

Louise Fischer and Mustafa Al-Janabi

***Abstract*—Functional testing is a vital process when building a satellite. However, often using flight-ready hardware for testing is not feasible. The work in this project has been to construct a flight representative model of the antenna deployment system for the KTH student-built MIST satellite. Specifically, the focus has been on creating a physical simulator for the antenna system. The purpose of the simulator created is to achieve the correct behavior, but without the need to use the real flight hardware.**

The challenges mainly concern establishing communication between the on-board computer of the satellite and the microcontroller on the created antenna deployment system, via the I^2C bus, and ensuring that physical responses occur in a useful manner. Further, the simulator needed to implement software with the same functionality as the real system. The microcontroller used in this project was an Arduino Due that represented the antenna deployment system's microcontroller.

All the functions, e.g. temperature sensor and LEDs, were put together on a custom-made add-on circuit for the Arduino. Moreover, a 3D-printed model has been made for the deployment mechanism of the antenna elements. A simulation of the antenna system has been produced, determining whether a custom-built simulator can be used for functional testing of the antenna deployment system. The simulator can later be used for functional testing of the MIST satellite and also be the base for testing the deployment of the solar panels.

***Index Terms*—MIST, CubeSat, Antenna deployment system (AntS), Burn wire, I^2C bus, Arduino**

I. Introduction

For a satellite to be considered small, the mass has to be no more than 300 kg. The main purpose of CubSats is to reduce the cost of a satellite project. Constraints on for example shape, size and weight make make it possible for companies to mass-produce components and reduce the launch cost of such a satellite remarkably [1]. CubeSats are built together with standardized CubeSat units with the size of 10 cm x 10 cm x $11,35$ cm, also called 1U. 1U have the maximum weight of 1.33 kg [2]. The KTH Royal Institute of Technology (KTH) student-built CubeSat Miniature Student Satellite (MIST) has the size of 3U, which gives the CubeSat the standard dimensions of 10 cm x 10 cm x 34 cm.

When sending up satellites to space, everything must work; fixing hardware failure later is not usually an option. To ensure the success of the satellite mission, all the subsystems of the satellite must work together. A so-called functional testing thus becomes necessary. Functional testing can be done on all the components in the satellite, in several ways. One of the systems that will be tested is the antenna deployment system (AntS). If the antennas do not deploy, the connection between ground and MIST will be lost, rendering the mission a failure.

The command to tell the antennas to deploy will be sent from the On-Board Computer (OBC) to the microcontroller on the antenna system via an I^2C bus. For the functional testing of the AntS, a physical model will show if the antennas are deployed or not. The goal of this project is to create a simulation device for the AntS.

The project addresses the following five questions:

1) Can one or two Arduinos along with associated custom-designed add-on circuit boards represent the microcontrollers in the AntS?
2) Can the Arduinos be powered from the electrical power system (EPS) of the satellite, and will it consume similar current?
3) How should the OBC be simulated (with an I^2C) during development?
4) How should the physical representation of the antenna elements be designed?
5) Which components should be used in the simulated physical design?

II. Background

A. Actual flight hardware

The actual flight hardware, see Figure 1, is delivered from the company Innovative Solutions In Space BV. (ISIS). The antenna system consists of four rolled-up antennas, each with one burn wire and two resistors, one of which is used as a backup. The antennas are made out of a NiTi-alloy based shape-memory alloy. Upon deployment, the resistor heats up and melts the burn wire after which the antenna pushes the lid open and extends itself. In order to prevent the resistors from becoming too hot, there is a safety time limit. To prevent the antennas from deploying unintentionally, there is an arm and disarm state for the system. This means the deployment is only possible when the armed command has been sent. Each antenna is provided with a switch that will be switched on when the antenna is deployed, this ensures the knowledge of a successful deployment. A successful deployment is estimated to take approximately three seconds [3].

Commands for deployment and for the switches are sent via I^2C between the OBC and the microcontroller on the AntS [3].

B. Requirements

The AntS made by ISIS has specification given in [3] from which the following requirements are derived. The typical deployment current for each antenna is $0.56\ A$ when the voltage is $3.3\ V$, and for a voltage of $5.0\ V$ the current is $0.3\ A$. There is also a safety time limit, set to 30 seconds. This is to make sure that the resistors do not overheat. The antenna system has 6 functionalities, these are given in the ISIS Manual:

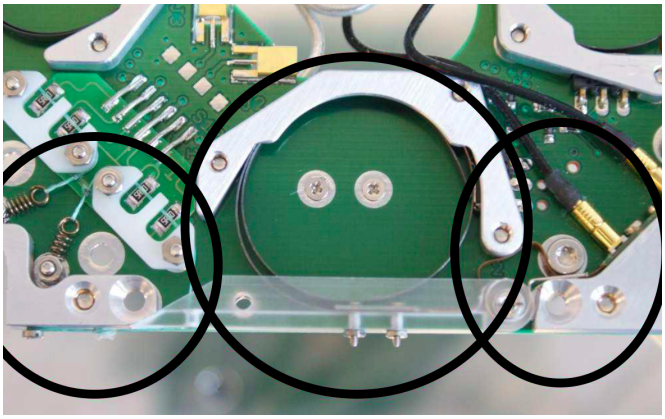

Fig. 1. Figure from [4] showing one of the four antennas in the antenna system. The left circle shows the two resistors and the burn wire that keeps the lid closed. The middle circle surrounds the stowed antenna and the lid, and in the right circles the antenna radio communication cable.

1) Reporting antenna deployment status
2) Arming and disarming the antenna system
3) Deployment of individual antennas
4) Automated sequential antenna deployment
5) Storage and reporting of the activation count and total activation time
6) Reporting system temperature

To deploy the antenna or to use another function of the antenna system, the OBC sends a command in form of a binary code called a Command Code. Different cases for the antenna system have different binary codes to send, that includes commands and device addresses. The real AntS uses an analog temperature sensor.

C. Microcontroller

A microcontroller is a micro sized computer that can be used to control different systems. On the antenna provided by ISIS there is a microcontroller on the device [3]. Unlike the real flight hardware, in this project an Arduino Due is used as microcontroller. The Arduino Due is based on 32-bit ARM core microcontroller. The Arduino has 12 analog inputs and 54 digital inputs/outputs pins and is equipped with a DC-jack as specified in [5]. The Arduino Due has a limited current output of 130 mA. Also used is an Arduino shield that is placed on top of the Arduino. This shield will create extra functionality to the Arduino [6].

Arduino integrated development environment (IDE) can be used to write and upload code to the Arduino board [7]. In the IDE the language that is used, called APL, is based on C++ and C [8]. In APL a so-called class can be used. A class is a place to collect functions and variables. Classes can be public or private; with public it means that other people can use it through a library [9]. One library can be used to make an interface to an Arduino OLED [10]. Another function that can be useful is called Boolean, a bool, can be true or false. This makes it possible to make a functions output vary between just two values [11].

D. FlatSat

A FlatSat is an assembly of different connected components of the satellite in two-dimensions. It is necessary to know that all the subsystems will work together. If the components do not work on the FlatSat they probably will not work in stacked mode [12]. In MIST, the FlatSat is where satellite components are put together and “connected but not fully integrated” connected but not fully integrated”, according to [13].

E. I^2C

The serial communication is via the I^2C bus. I^2C bus stands for Inter-Integrated Circuit bus. The system has two communication lines, one Serial Data Line (SDA) and one Serial Clock Line (SCL). The advantage of using I^2C is that the speed of the communication does not have to be set in advance [14]. To use an I^2C communication a library is required. To allow I^2C communication the Wire library is used. In the Wire library the command *Wire.onReceive(handler)* can be used to call a function when the slave Arduino receives a command from the master Arduino. Also used is the *Wire.onRequest(handler)* command, that instead call a function when a request from the master Arduino is sent to the slave Arduino for data [15]. Interrupt Service Routine (ISR) function that does not return anything, but can handle requests and sends them further [16] [17].

F. Serial Peripheral Interface

Serial Peripheral Interface (SPI) is a bus for serial communication between master and slave microcontrollers. There are three lines for the communication, Master In Slave Out (MISO), Master Out Slave In (MOSI) and Serial Clock (SCK). MISO is the line for the slave to send data to the master, MOSI is the line for sending data to the peripherals and SCK is a clock generated by the master in order to synchronize data communication [18].

G. Components

To produce all the functionalities for the AntS simulator, other components are used. These are described in this section. Using a burn wire is an easy and reliable way to deploy components out in space. When a current passes through resistors, the resistors heat up and break the wires. Burn wires in the antenna and solar panel deployment systems for CubeSats are common [1].

The Metal Oxide Semiconductor Field Effect Transistor (MOSFET) is a component that can switch and amplify electronic signals [14].

The actual AntS uses the so-called J1 connector, which is a female Omnetics Bi-Lobe dual row connector of 9 pins (reference A29100-009 [3]. A similar connector is the D-subminiature (D-sub), where the name comes from their D-shaped shield [19].

When 3D-printing, first of all a design of the object is necessary. This design can be made in different computer programs, for example Solid Edge or Shapr3D. And after the drawing is done, it is transmitted to the 3D-printer and then printed.

H. Ohm's law

To calculate the resistance of the resistor and getting the right current in the circuit, Ohm's law can be used. This gives a relation between the voltage, current and resistance.

$$U = R \cdot I \quad \Rightarrow \quad R = \frac{U}{I} \tag{1}$$

III. Hardware Design

A. Deployment circuit

The centerpiece of the entire circuit is the set of four $15\,\Omega$ resistors used to burn off the burn wire holding the antenna lid in place. Power to the resistors is supplied by the EPS. Since the Arduino's current output was limited to 130 mA [5], it becomes necessary to control the current flow to the resistor by an external component which can be controlled by the digital pins of the Arduino. For that purpose, surface mounted MOSFET-transistors (of type IRL6244) are used for each resistor. In series with each transistor, a generic light emitting diode (LED) with a unique color is connected to indicate current flow.

To make the AntS as user-friendly as possible for testing purposes, an OLED screen displaying the current state of the AntS is also connected. The OLED screen communicates with the Arduino via the SPI bus.

Power to the burn wire resistors is supplied by the EPS. The Arduino representing AntS is powered separately by a lab power supply from which the Arduino takes 7 to 12 V. The Arduino is connected to the lab power supply via its built in DC-jack.

Connection between the OBC, EPS and the Arduino simulating AntS happens through a 9-pin D-sub connector. Only four pins of the D-sub are utilized. Two for the I^2C communication (pins 2 and 7 in Figure 4) with the OBC and two for the $5\,V$ DC-power (pins 1 and 3 in Figure 4) from the EPS to power the burn wire resistors. Lastly, the simulator also includes a digital temperature sensor.

All the components are assembled as shown in the block diagram, Figure 2.

The detailed description of each component is described in Section III-B. A photograph of a working prototype is shown in Figure 3.

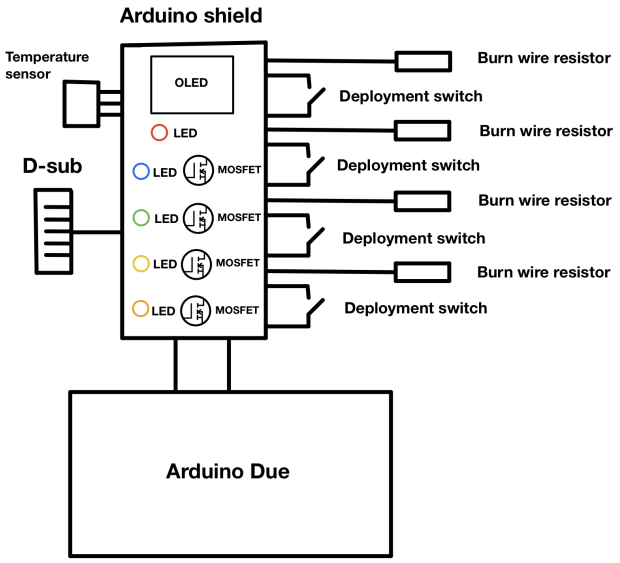

Fig. 2. All the parts in the electrical part of the AntS simulator. The LED represented by a red ring indicates arm/disarm status. The blue, green, yellow and orange LEDs indicate ongoing deployment of antenna element 1, 2, 3 and 4 respectively.

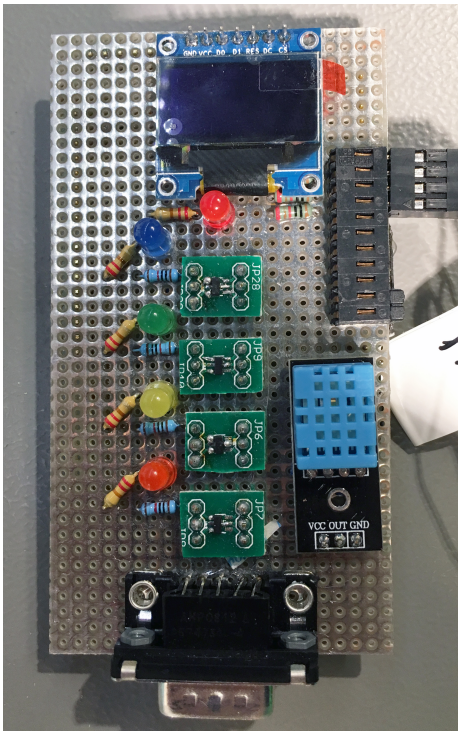

Fig. 3. The custom-made Arduino shield with all the parts assembled.

B. Electrical components

1) Arduino: The Arduino used in this project is the Arduino Due [5].

2) Burn wire resistors: The resistors choice of a set of four $15\,\Omega$ resistors was based on Ohm's law, Equ. 1. The ESP supplies the AntS with $5\,V$. Further it was specified in [3] that, during deployment, the AntS should consume 300 mA of current.

3) MOSFET-transistors: The IRL6244 MOSFETs are used.

4) OLED Screen: The screen that is used is the OLED LCD Display with SPI Serial SSD1306.

5) Temperature sensor: The VMA311 DHT11 digital temperature humidity sensor module for Arduino is used [20].

6) Antenna deployment switches: To indicate that an antenna element is deployed, a physical button is used. With the button being lightly pressed, the antennas are to be considered stowed. When released, the antennas are to be considered deployed, meaning that the burn wires burnt off and the lid is open.

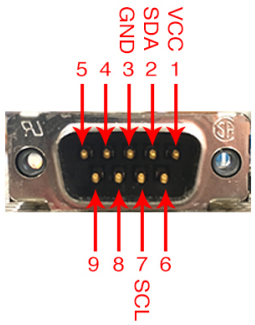

Fig. 4. The 9-pin male D-sub mounted on the AntS simulator with the pin numbering shown. VCC is the input for 5V, GND is ground, SDA is the serial data and SCL is the serial clock.

Fig. 5. The 9-pin female D-sub attached to four wires with pin headers for the four inputs of the male D-sub on the AntS simulator

7) Connectors: A 9-pin male D-sub is used for the connector on the AntS simulator, see Figure 4 and Figure 5. Only pins 1, 2, 3 and 7 are used. Numbering of the male 9-pin D-sub is done as for the real connector according to Figure 4. A female connector with a 9-pin female D-sub in one end and four pin headers in the other is also made (Figure 5) to connect to the FlatSat.

C. Physical deployment mechanism

The physical model of the deployment mechanism must fulfill criteria. Firstly, it must be easy to reload with new burn wires after a deployment test. Secondly, it needs to give a physical response to the deployment of each antenna separately. Lastly, there has to be a deployment switch for every antenna to indicate successful deployment digitally.

The initial sketch of the deployment mechanism is made in Solid Edge, and is later finalized for 3D-printing in an iPad application called Shapr3D (Figure 6).

As can be seen in Figure 6 the antenna elements consist of a solid cube with a notch and a cylinder. The notch matches the main section of the 3D-printed parts and keeps the antenna element straight during deployment. On the cylinder a spring is mounted, using the lid as a way to keep the antenna element stowed with the burn wire tied to hold the lid closed. The

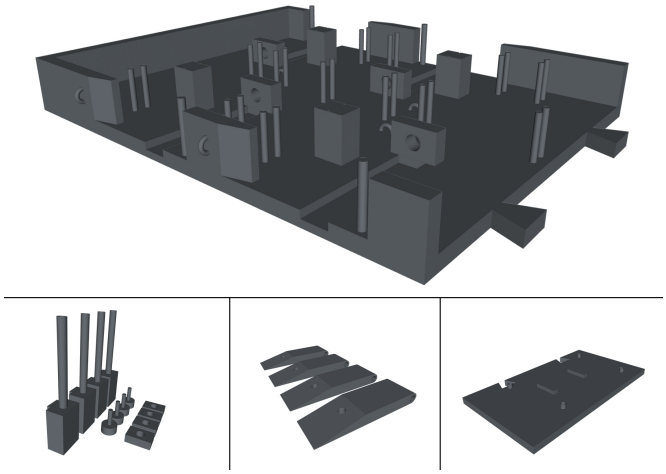

Fig. 6. On the top: the 3D-model of the main part of the physical model. Bottom left: The antenna elements with their parts. Bottom middle: the lids to keep the antenna elements in place. Bottom right: Arduino Due mount, the part is mounted with the main section.

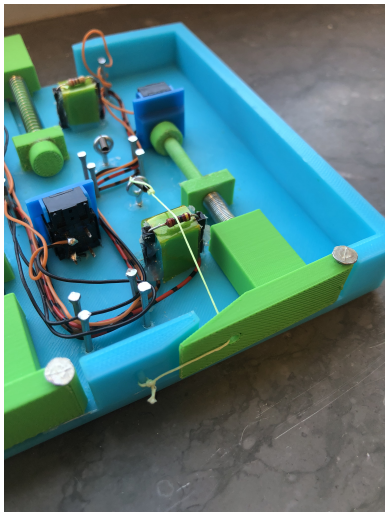

Fig. 7. The 3D-printed model with a stowed antenna held in by a burn wire.

burn wire resistors are mounted beside each antenna element adjacent to the holes in the lids (Figure 9). When the resistor burns the burn wire, the spring pushes the antenna element outwards, meaning that the blue buttons in Figure 9, are released and the antenna element is deemed to be deployed.

Some parts of the model had to be glued or nailed, so that the part did not break or fall out. The resistors are placed on a cube covered in a tape that isolates the heat from the 3D-printed part made of plastic. The burn wires are tied to the 3D-printed part and to a spring to ensure no slack occurs (Figure 7).

Reloading the antenna element is a matter of pushing it back until it pushes the switch, then closing the lid and tying a new burn wire string to the mounting points. If the burn wire resistors get burnt out, they can easily be replaced by new ones. The burn wire that is used is a type of fishing wire, Strike Wire X12 0,38mm/39kg.

IV. Software design

A. I^2C communication

As part of the first step in designing the simulator, ensuring communication between two Arduino Dues via the I^2C bus is essential. This was done using the *Wire* library. Upon connecting two I^2C devices via their designated pin header for the SDA and SCL, it was immediately possible to establish communication. The library was utilized in the AntS simulator both to receive commands from and respond with data to the OBC. All the binary codes for the commands and device addresses are put in a header file (.h) that is included in the main AntS simulator code.

B. Implemented commands

The deployment sequences are implemented according to the two deployment scenarios specified in the AntS manual; for nominal deployment and deployment with override [3]. Further, the following commands are implemented (all according to the AntS manual [3]) in the AntS simulator:

1) Arm: Only when armed, the AntS is able to execute deployment of the antenna elements. Implemented as a Boolean-changing class method with the Boolean variable indicating the current armed status. The *arm* command sets the Boolean variable to true.

2) Disarm: Disarms the antenna and terminates any ongoing deployments. Changes the same Boolean indicating the armed status to false.

3) Cancel: Only when armed, this command terminates any ongoing antenna deployments.

4) Deploy antenna: Deploys the specified antenna element using the nominal scenario. The commands are sent in two bytes, one for the command and the other specifies the maximum activation time. If that value is zero or longer than the safety timeout of 30 seconds, the safety timeout is used to limit the deployment system, i.e. the time for which the burn wire resistors are active.

5) Deploy antenna with override: Works exactly like the *deploy antenna*-command, except it skips checking the value of the deployment switches. Deploying with override is useful when the readings from the deployment switches are considered unreliable.

6) Sequential automated deployment: Attempts deployment of each antenna element one at a time. The implementation of it uses the class methods for the regular deployment with the maximum activation time being the same for each antenna element.

7) Measure temperature: Upon receiving this command the AntS simulator takes a temperature reading from the DHT11 sensor and converts it to raw data using the conversion formula of the temperature sensor used on the real AntS [20]. That data is then sent back to the OBC as a two-byte message with only ten significant bits.

8) Report status: A two-byte message is constructed and sent to the OBC. The message is based on Boolean variables describing the current state of the antenna system and are ordered in two bytes according to the data sheet [4].

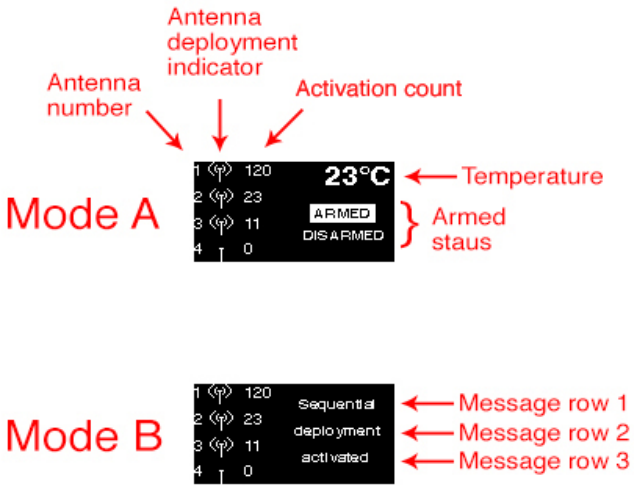

Fig. 8. The layout of the graphical interface for the OLED screen. Mode A is the default screen shown and mode B is shown for only one second when there is a message to show.

9) Report activation count: Each time the deployment system of an antenna element is activated a counter is increased by one. When reporting the activation count to the OBC, the AntS simulator returns a byte containing the counted number of activation of the specified antenna element.

10) Report activation time: Calculates the total accumulated activation time in seconds of the specified antenna, it then takes the number of seconds and multiplies it by $\frac{1}{0.05} = 20$ to return the elapsed time in 50 millisecond steps.

All the commands are checked by a switch case, which controls the command upon receiving it, the parameter (when available) is then checked. All command-functions are written as class methods belonging to the class AntennaSystem. Some of the methods that only change a Boolean value (like *arm* and *disarm*) are called immediately within the *onReceive*-event function. However, functions that require time measurements (like the deploy commands) are activated in the Arduino's loop function. Once a time-measuring command is received by the AntS simulator, its corresponding Boolean becomes true and is thus activated in the Arduino loop cycle. The reason for that is because the *onReceive*-event is an interrupt request, which stops the time tracking of the Arduino while it is executed.

The *Reset*-command is omitted since the Arduino already has a physical reset button that can be used if resetting the AntS becomes necessary during testing.

C. Screen code and design

The screen is designed according to the layout shown in Figure 8. To design the interface for the screen, the Arduino OLED library [10] is used. Given different occasions in the testing cycle of MIST using the AntS simulator, different messages (see Table I) are displayed to make interactions with the simulator more user-friendly.

D. OBC-simulator code

During the development of the software and hardware of the AntS simulator, an Arduino was used as a simulator of

TABLE I
EACH POSSIBLE MESSAGE WITH THE CORRESPONDING OCCASION AT WHICH THEY ARE EXPECTED TO BE DISPLAYED.

Message	**Occasion**
”Armed”	Antenna system is armed.
”Disarmed”	Antenna system is disarmed.
”Sequential deployment activated”	A sequential deployment starts successfully.
”Deploying antenna ##”	A deployment is successfully initiated of the antenna number ##.
”Deploying ## with override”	A deployment with override is successfully initiated of the antenna number ##.
”Reporting status”	The OBC requests a status report from the AntS simulator.
”Antenna deployment successful”	The time limit is passed or the antenna deployment switch is released.
”Antenna already deployed”	The antenna deployment switch indicates the antenna is deployed.
”Antenna deployment ongoing”	A deployment of another antenna is already ongoing.
”Must be armed”	Attempting to initiate an antenna deployment before arming the Ants Simulator.
”Sequential deployment canceled”	An ongoing deployment is canceled.

the OBC. This code consisted of functions callable from the serial monitor of the Arduino IDE. The functions also utilize the *Wire*-library and are a combination of functions that send a command code for the AntS simulator to be executed as well as functions that send a request for information from the AntS.

V. RESULTS

The result shows that a flight-representative response has been produced. This response comes from a command sent by the OBC via an I^2C bus and is represented by LEDs that light up, see arrows labelled by A in Figure 9.

Also, distinct physical responses are created with antenna elements, see circles at C in Figure 9, that get released during commands. For this, the burn wire and resistors are used, the resistance calculated is:

$$\frac{5V}{300mA} = 16.6666...\Omega. \tag{2}$$

Testing a resistor with the resistance of $15\ \Omega$ proves to be viable, by which it is meant that the given voltage and current supplied to the resistor heats it enough to burn off a burn wire within approximately ten seconds. An $18\ \Omega$ resistor is also tested, but did not get nearly as hot as required.

The simulator provides temperature readings, and presents it in a user-friendly way on an OLED screen, seen in circle B in Figure 9. The whole simulation is tested together with the OBC, and it is fully working as desired. It takes approximately seven seconds for the antenna element to deploy.

VI. DISCUSSION

With the obtained results it is possible to make the conclusion that successful simulations of the antenna deployment of MIST can be achieved using the produced AntS simulator. However, these tests are performed in an environment, here on earth, where the temperature is at normal room temperature. In space, the temperature will differ substantially from the one tested which is a test case that is not covered by the AntS simulator.

A. Choice of components

1) Arduino Due: The main reason to use the Arduino Due was because of the Arduino Due's ability to work with $5\ V$, which is similar to the AntS. However, this proved unnecessary since the Arduino ended up only being used as a switch for the MOSFET-transistors. On the other hand, the large amount of pins was found to be useful, as compared to a smaller Arduino.

2) Burn wire resistors: The choice of resistors was one of the limiting factors in this project, because off-the-shelf resistors require high power to get hot.

Since a $15\ \Omega$ resistor consumed more current than $300\ mA$ and a $18\ \Omega$ resistor did not get hot enough, one can conclude that a resistor with resistance in between $15 - 18\ \Omega$ would have been optimal. This is confirmed by Ohm's law (Equation 1). However, such resistors were not available. It is a possibility to connect two resistors in series and have a more tailored resistance, which does, on the other hand, infer some limitations. Firstly, the heat dissipation would be spread over two resistors, which might impede the resistor's ability to get hot enough for burning off the burn wire. Secondly, it would have made the 3D-designed physical model bulkier and more cluttered.

Although the current consumption of the $15\ \Omega$ was more than specified, the extra current needed was deemed negligible for the testing phase of MIST. Burn wire resistors with lower power rating than the ones used would have been better.

3) Burn wire: The choice of the Strike Wire X12 as the burn wire was based on its availability. Qualitatively, it looks similar to the real burn wires on AntS. However, choosing a thinner wire might have decreased the deployment time by a few seconds, and thus making the deployment simulation more realistic to the real AntS, which is expected to finish deployment within 3 seconds.

4) MOSFET-transistors: The surface mounted MOSFET-transistors where used due to their small size and their more than sufficient current and voltage ratings. In the testing phases of the project they proved to work flawlessly and were used in the final design too.

5) OLED screen: Because the AntS Arduino was already in slave mode, when communicating via the I^2C bus, the screen had to use a different communication protocol. Thus, an OLED screen that uses the SPI communication protocol was chosen. Further, it already had an Arduino library specifically for it, which made the design of the graphical interface less cumbersome. A larger screen with colors would have improved the legibility and user friendliness of the AntS simulator.

6) Temperature sensor: The fact that the temperature sensor in the real AntS has an analog sensor makes the DHT11 in the simulator a poor choice due to it being a digital sensor. Converting the digital reading to an analog one similar to what the real sensor would have given is not a future proof

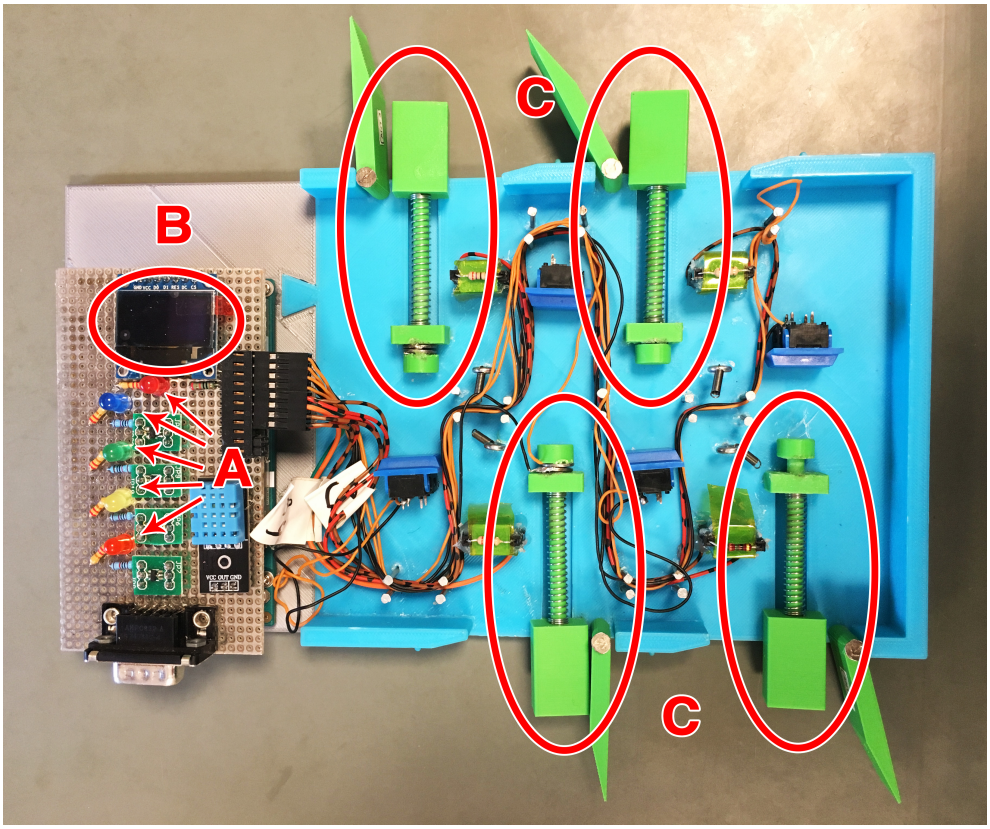

Fig. 9. A top view of the final 3D-printed and assembled model. The arrays from A represent the LEDs that light up when deployed. The circle B surrounds the OLED screen and the four C circles the antenna elements.

method. That is because the theoretical conversion formula could, in practice, be inaccurate. Furthermore, there is a chance that the implementation could be wrong since there is no real way to test whether the converted value is correct. Thus, an analog temperature sensor would have been a better alternative. On the other hand, the temperature reading is not essential for the function of the antenna simulator during functional testing. The disadvantage of the conversion method is, thus, insignificant for the scope of this project.

7) Antenna switches: The switches in the physical model where buttons that released during deployment, opposite to the way the real flight hardware would have worked. To make it more like the real hardware it could, be done with buttons that get pressed in when deployed. As antenna switches go, the buttons installed behind each antenna element (see Figure 7) work as anticipated. However, they are not a sustainable solution. Using buttons decreased the reliability due to the fact that ensuring they are properly pushed in when the antenna element is stowed cannot be fully guaranteed. Moreover, if the burn wire itself is tied somewhat loosely the antenna element would be partially deployed and, thus, the antenna switch would give the wrong reading. Giving the limited time in the project it wasn't possible to implement a more robust solution. A possible improvement is to use a reliable sensor, instead of buttons as deployment switches, for instance a Hall effect sensor with a magnet attached to the backside of the antenna elements.

8) Connectors: Due to the price and availability of such a connector a female 9-pin D-sub is used for the AntS simulator. Because the FlatSat setup itself doesn't use the real connectors during the early stages of functional testing, where the AntS simulator are mainly intended to be used, having a connector doesn't actually affect the functionality of the simulator. The D-sub was used to make a tidier simulator and a symbolic similarity to the real AntS.

B. 3D-printed model

One of the criteria for the simulator's physical model was that it should be easy to reload. The model fulfilled that criterion as reloading the antenna elements required only a simple push to stow it. The difficult is making sure that the burn wire is tied tightly without slack. Using a clamp whilst retying the wire could be a possible solution. Although it would require a strong clamp as the fishing wire is slippery.

When planning and sketching the physical model, Solid Edge was used in the beginning, but half way through it was changed to an iPad app. It would probably have been easier to stick to one program and from that export it to the 3D-printer. The 3D printing made it possible to get the exact shape and dimensions of the model, but the plastic broke easily and melted. This made it difficult to work with when everything was supposed to be assembled. Sticking to Solid Edge and ensuring to keep enough margins in the design would have been a solution to make the structure more robust.

C. Simulator compared to the real AntS

Clear differences are observed between the simulated and the real AntS. First, the simulator only concerns the deployment of the antennas, while the real AntS also utilizes the radio communication. Second, the simulator did not have the same resistors or burn wire as the real flight hardware. This was the main reason the deployment time difference between the two systems. In addition, the temperature sensor differed from the one on AntS. For the scope of this project these differences have a negligible result on the ability to preform functional testing.

VII. Conclusions

As can be seen in the results, a simulator of the antenna deployment system of MIST was achieved in this work.

With a custom-designed add-on circuit board, the Arduino Due proved to be a viable choice to represent the microcontrollers in the AntS. It can however be stated that the Due-model is not necessary and any other Arduino with I^2C and SPI communications, along with a sufficient amount of pins would have been satisfactory. That is because the current to the burn wire resistors was supplied by the EPS directly and not through the Arduino itself. The Arduino Due was initially considered a better choice due to the fact that it has a $5\ V$ output as opposed to just $3.3\ V$.

It was found that an Arduino is also a good choice to simulate the OBC during development, considering a correct implementation similar to the real OBC and with the same device codes for the I^2C communication as the flight hardware.

Using only off-the-shelf resistors for the purpose of burning off a burn wire was tricky given specific current and voltage limitations. However, the current consummation achieved by the AntS simulator which used off-the-shelf resistors is deemed close enough to the specifications of the real AntS, at least for the primary phase of functional testing of MIST.

To review, the created AntS simulator has a plausible physical response with the software part working much like the real AntS. Adding a screen and LED-indicators makes it user-friendly for testing purposes. The current consumption comes close to the real specifications. Thus, the AntS simulator created in this project will be sufficient for the functional testing of MIST, thereby fulfilling the aim of this project.

VIII. Future work

Given the nature of the project, the software part is almost optimal since it follows the data sheet [3] in the manual given by the supplier of the flight hardware. However, there are many possible improvements that can be done the hardware part of the project. When using the AntS simulator for demonstration purposes, one could add a mode selection switch to the custom-made Arduino shield. When in demonstration mode, the current could be supplied by a commercial $9\ V$ battery and a different set of resistors to achieve the desired results when burning off the burn wire. In that case it is assumed that the OBC simulator is used to send the commands to the AntS simulator.

The AntS simulator could be designed to have real antenna elements with the ability to transmit and receive data. This would be beyond the aim of this project. The reloading mechanism is still somewhat fiddly and could be made more efficient with an improved design. Lastly, when it comes to following the data-sheet [3] a different set of resistors which dissipate more heat and with a resistance that would get the current consumption closer to $300\ mA$.

Most importantly, a method for using the AntS simulator in the functional testing of the MIST satellite needs to be devised. At the time of writing the AntS simulator has yet to be used for actual testing of the satellite. Tests have only been performed to ensure that the AntS simulator fulfills its criteria.

The simulator can also be used for the functional testing of the deployment of the solar panels, since they have a similar deployment mechanism.

Acknowledgment

First and for most, we like to thank every member of the MIST team for great technical support throughout the project. Further we thank our project manager Sven Grahn, without whom this project would not have been possible, for his invaluable insights and enthusiasm. Last, but not least, we're very grateful to have had Francesca Capel as our supervisor. We're thankful for her passionate guidance, encouragements and continuous follow-ups, which helped us reach our goals smoothly.

References

[1] *CubeSat 101, Basic Concepts and Processes for First-Time CubeSat Developers*, NASA - National Aeronautics and Space Administration, Oct. 2017. [Online]. Available: https://www.nasa.gov/sites/default/files/atoms/files/nasa_csli_cubesat_101_508.pdf

[2] Swedish National Space Agency. (2018, Mar.) Vad ar en cubesat? SNSA, Solna, Sweden. [Online]. Available: https://www.rymdstyrelsen.se/upptack-rymden/bloggen/2017/03/vad-ar-en-cubesat/

[3] *Antenna System User Manual*, ISIS Innovative Solutions In Space, Oct. 2014, issue 1.44.

[4] *AntS Datasheet. Innovative, deployable upon command and compact Antenna System especially designed for CubeSat missions.*, ISIS Innovative Solutions In Space, May 2011, issue 2.0.

[5] Arduino . (2019, Apr.) Arduino due. Code: A000062. [Online]. Available: https://store.arduino.cc/due

[6] Arduino. (2019, Apr.) Shields. [Online]. Available: https://www.arduino.cc/en/Main/arduinoShields

[7] Arduino . (2019, Apr.) Software - download the arduino ide. [Online]. Available: https://www.arduino.cc/en/Main/Software

[8] Arduino. (2019, Apr.) Arduino software. [Online]. Available: https://www.arduino.cc/en/main/FAQ

[9] Arduino. (2018, Apr.) Writing a library for arduino. [Online]. Available: https://www.arduino.cc/en/Hacking/LibraryTutorial

[10] Adafruit industries. (2019, Apr.) Adafruit ssd1306. [Online]. Available: https://github.com/adafruit/Adafruit_SSD1306

[11] Arduino. (2019, Apr.) Data types - bool. [Online]. Available: https://www.arduino.cc/reference/en/language/variables/data-types/bool/

[12] AMSAT-UK, FUNcube. (2010) Flatsat meeting - what is a flatsat meeting? . [Online]. Available: https://funcube.org.uk/working-meetings/flatsat-meeting/

[13] M. Gruber, "Functional testing of the mist satellit," Master's thesis, KTH, Stockholm, 2018.

[14] P. Lindahl, R. Meyer, H. Johnsson, M. Grimheden, W. Sandqvist, and M. Paulson, *Elekroteknik*. Instutionen for Maskinkonstruktion, Mekatronik, 2006, Stockholm.

[15] Arduino . (2019, Apr.) Wire library. [Online]. Available: https://www.arduino.cc/en/reference/wire

[16] Arduino. (2019, Apr.) attachinterrupt(). [Online]. Available: https://www.arduino.cc/reference/en/language/functions/external-interrupts/attachinterrupt/
[17] Pramoth Thangavel. (2019, Feb.) Arduino interrupts tutorial. [Online]. Available: https://circuitdigest.com/microcontroller-projects/arduino-interrupt-tutorial-with-examples
[18] Arduino . (2019, Apr.) Spi library - a brief introduction to the serial peripheral interface (spi). [Online]. Available: https://www.arduino.cc/en/reference/SPI
[19] PC. (2019, Apr.) Encyclopedia - d-sub connectors. [Online]. Available: https://www.pcmag.com/encyclopedia/term/55439/d-sub-connectors
[20] *LM94022/-Q1 1.5-V, SC70, Multi-Gain Analog Temperature Sensor With Class-AB Output*, Texas Instruments, 2015.