Design of a terminal node controller hardware for CubeSat tracking applications

IOP Conf. Series: Materials Science and Engineering **152** (2016) 012031 doi:10.1088/1757-899X/152/1/012031

Design of a terminal node controller hardware for CubeSat tracking applications

Y A Ahmad[1*], N J Nazim[2] and S S Yuhaniz[3]

[1]Kulliyyah of Engineering, International Islamic University Malaysia, Malaysia
[2]Faculty of Engineering, Universiti Selangor, Malaysia
[3]Advanced Informatics School, Universiti Teknologi Malaysia, Malaysia

[*]yasser.asrul@gmail.com

Abstract. CubeSats enable low-cost experiment and missions to be performed by universities and research institution in space. CubeSats for research use UHF and VHF communication for its tracking and telemetry applications. The current practice of a CubeSat communication is to modify radio amateur's Terminal Node Controller (TNC) to enable data to be received in the ground station. The objective of this research is to design a hardware specifically for use as a TNC for CubeSat tracking applications. A TNC is developed as an interface to the terminal and to serve as data packetization platform. The modem is integrated with a microcontroller unit (MCU) and an audio amplifier to enable the audio signals to be smoothened, amplified and interfaced with the radio. The modem, MCU and audio amplifier circuitry are designed and integrated to form a TNC platform suitable for CubeSat communication.

1. Introduction

CubeSat technology has enabled scientific research and development of space technology to be done with relatively low cost, which facilitates affordable space research and development at the university levels. In addition, many new space companies have also mushroomed based on the use of CubeSat concept for commercial applications of remote sensing, radio amateur and scientific research in space [1]. Essentially, CubeSat is a nanosatellite-class of satellites with fundamental weight of one kilogram and a volume of 10 cm x 10 cm x 10 cm. This standard is accepted worldwide and is known as 1U, where U denotes unit. The CubeSat is expandable into 2U, 3U, 6U and even 12U.

The CubeSat is basically a miniaturization of classical satellite, which is heavy, big and expensive in nature. It consists all subsystems in a satellite such as electrical power system, onboard computer, attitude determination and control system, communication system and mechanical system. CubeSats are launched into the Low Earth Orbit (LEO) as a secondary payload piggyback. Once deployed into orbit, CubeSat is tracked and communicated via a ground station.

2. CubeSat communication

CubeSat typically uses amateur radio frequencies and protocol for its packet data transmission [2, 3]. Based on the survey made by Klofas and Leveque [4], 70% of the total CubeSats launched into orbit between 2009 to 2012 used amateur radio frequencies as a part of their communication systems. Many CubeSats that are developed by universities [5, 6, 7] used slower data rates due to the ease of getting and implementing the hardware. These projects were mostly decided to build entire transceiver out of individual components that allow for tighter control of requirements and specification. Components of

these custom-built transceivers also include terminal node controller (TNC), transceiver and amplifier. TNC are typically built and embedded in the satellite On Board Computer (OBC) and transceiver, but in term of TNC in the ground station, the commercial-off-the shelf (COTS) components are typically used [2, 4].

The TNC is used in the CubeSat packet radio communication, where digital signals are enabled to propagate using radio waves. It is a device used by amateur radio operators to participate in AX.25 packet radio networks [8, 9, 10, 11]. The primary usage of a TNC is as a bridge between a computer or a programmable device to radio transceiver with packet network capability [9, 10]. TNC functions as a modem and ensures the data are formatted and packetized based on packet radio requirements such as AX.25. TNC system is separated by digital part, analog part and TNC software [11]. In the CubeSat system, the experimental data will be collected and downloaded to the ground station. The telemetry data is also downloaded to the ground station. Most of the TNCs used in CubeSat missions are mostly COTS based [4]. Though the COTS TNC are implementing AX.25 packetization, the performance of each COTS is not the same because of differences and compatibility issues arises due to factors such as implementation details of AX.25 protocol, properties of the software of the TNC microcontroller and hardware properties, which affect the performance of a TNC in terms of effective transmission speed and network efficiency [11]. The use of a COTS TNC in the ground station will result in the compatibility issues and poor communication link with a CubeSat in space. Reliance on the COTS components are making the CubeSat mission difficult because the software and hardware properties of COTS components cannot be changed as it is set by the manufacturer. Therefore, it is important to have access to TNC source code for modification and capability to design the TNC hardware. A self-developed TNC for CubeSat would enable compatibility issues to be fine-tuned during the CubeSat development and modified even in the satellite operations.

A common ground station consists of an antenna system, a receiver, a transmitter, a terminal node controller, computers and operation software**.** TNC consists of two major components: microcontroller unit (MCU) and modem. The main function of the MCU is to implement the AX.25 packet protocol in the embedded source codes. The output of AX.25 packet encoding process will be the packet frame, which is the data in digital signal (TTL). The digital modulation must then be performed to convert the digital signal into analog signals before it that can be propagated using radio waves. The modulation process will be performed by the digital modem. The AX.25 packet protocol is a derived version of X.25 protocol that is formerly used in amateur radio packet transmission [12, 13]. It is an amateur radio specification that describes how to encode digital data to transmit it over radio frequencies. The AX.25 specification mandates a bit rate of 1200 baud and uses audio frequency shift keying (AFSK) encoding to represent binary values 0 and 1 with audio tones of 1200 Hz and 2200 Hz, respectively [14, 15]. The modem will operate in AFSK for lower baud rate and in frequency shift key (FSK) and Gaussian minimum shift keying (GMSK) for higher baud rate. Both AFSK and also FSK/GMSK modulation methods are very common in CubeSat communications [2, 3, 4]. AFSK digital modulation is implemented to convert the digital signals into analog signals before it that can be propagated using the radio waves. Figure 1 illustrates the position of a TNC and functions of its components in satellite communications.

3. The AX.25 frame

Data link layer packet radio transmissions are sent in small blocks of data called frames. Each frame is made up of several smaller groups called fields. Each field is made up of an integral number of octets (8-bit binary) and serves specific functions. All fields except the Frame Check Sequence (FCS) are transmitted low-order bit (LSB) first while FCS is transmitted high-order bit (MSB, i.e. bit-15) first [14, 15]. Specifically, the Unnumbered Information (UI) frame format of AX.25 protocol is utilized in most CubeSat communications schemes.

AX.25 UI frame shown in Table 1 starts with a flag and followed by the frame header, information (data), CRC checksum (FCS) and ends with another flag. The packet frame header consists of the destination address, source (origin) address, control bits and a protocol identifier (PID). Both of the destination and source addresses are included with Secondary Station Identifier (SSID).

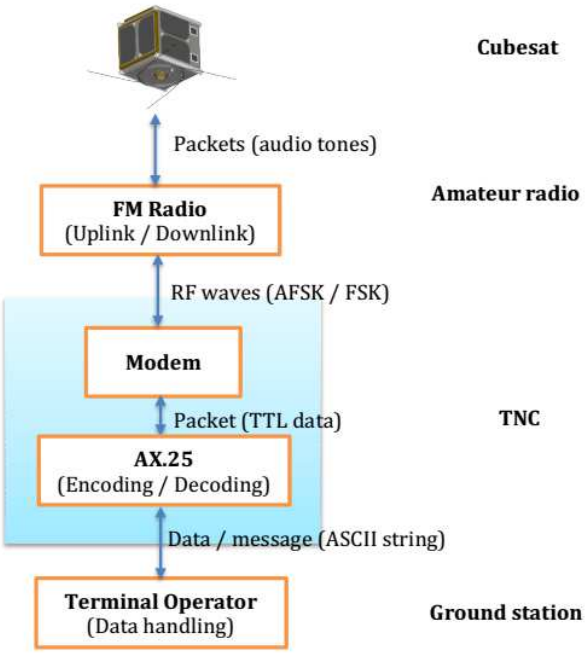

Figure 1. Application of a TNC in communication between ground station and CubeSat

Table 1. AX.25 Unnumbered Information (UI) frame [14]

Flag	AX.25 Transfer Frame Header (128 bits)				Information Field	Frame-Check Sequence	Flag
	Destination Address	Source Address	Control Bits	Protocol Identifier			
8	56	56	8	8	0-2048	16	8

3.1. Flag field

A flag is a specific number indicating start and end of a frame. It is one octet long, occurs at both the starting and the end of each frame. Two frames may share one flag, which denotes the end of the first frame and the start of the next frame. A flag consists of a zero, followed by six '1's and then another zero (b01111110 or 0x7E) [15, 16]. To ensure that the flag bit sequence mentioned does not appear accidentally anywhere else in a frame, bit stuffing is applied. The sender monitors the bit sequence for a group of five or more contiguous '1' bits. Any time five contiguous '1' bits are sent, the sending station inserts a '0' bit after the fifth '1' bit. During frame reception, any time five contiguous '1' bits is received, a '0' bit immediately following five '1' bits is discarded [15, 16].

3.2. Address field

The address field identifies both source of the frame and also its destination. The destination address consists of the callsign and the SSID of the destination, as shown in Table 2.

Table 2. Destination address frame field (56 bits) [14]

Callsign (48 bits)			SSID 8 bits
C1 (8 bits)	....	C6 (8 bits)	
X XXXXXX 0		X XXXXXX 0	0 1 1 SSID 1

The callsign is made up of six upper-case letters, numbers or space ASCII characters only (7 bits). The SSID is four-bit integer that uniquely identifies multiple stations using the same amateur callsign. The six characters of the callsign are placed in the first six octets of the field (C1 to C6). Each of the character bits are shifted one bit on the left and the LSB is set to '0'. The SSID is placed in bits 3-6, while other bits of the field are contained with fixed values [15].

Table 3. Source address frame field (56 bits) [13]

Callsign (48 bits)			SSID 8 bits
C1 (8 bits)	….	C6 (8 bits)	
X XXXXXX 0		X XXXXXX 0	0 1 1 SSID 1

The source address frame shown in Table 3 consists of the callsign and the SSID of the source. The callsign and the SSID configurations are similar to the destination callsign and SSID as stated earlier. The only bit that differs from the destination address field is the leftmost bit, which is set to '1'. This indicates that the octet to which it belongs is the last octet of the address fields [14].

3.3. Control field

The control field identifies the type of frame being passed and controls several attributes of the Layer 2 connection. For an AX.25 Unnumbered Information Frame, its value is always b00000011 (0x03) [14, 15, 16].

3.4. Protocol identifier (PID) field

The protocol identifier (PID) field appears in information frames (I and UI) only. It identifies the kind of Layer 3 protocol, if any, that is in use. The PID itself is not included as part of the octet count of the information field [16]. The definition of PIDs can be referred to Table 4. For this TNC application, the PID shall be 11110000 (0xF0).

Table 4. PID definitions of an AX.25 packet frame [15]

Hexadecimal	MSB LSB	Translation
**	yy01yyyy	AX.25 layer 3 implemented
**	yy10yyyy	AX.25 layer 3 implemented
0x01	00000001	ISO 8208/CCITT X.25 PLP
0x06	00000110	Compressed TCP/IP packet. Van Jacobson (RFC 1144)
0x07	00000111	Uncompressed TCP/IP packet. Van Jacobson (RFC 1144)
0x08	00001000	Segmentation fragment
0xC3	11000011	TEXNET datagram protocol
0xC4	11000100	Link Quality Protocol
0xCA	11001010	Appletalk
0xCB	11001011	Appletalk ARP
0xCC	11001100	ARPA Internet Protocol
0xCD	11001101	ARPA Address resolution
0xCE	11001110	FlexNet
0xCF	11001111	Net/ROM
0xF0	11110000	No layer 3 protocol implemented
0xFF	11111111	Escape character. Next octet contains more Level 3 protocol information
Escape character. Next octet contains more Level 3 protocol information	00001000	

3.5. Information field
The information field conveys the user data (message) from one end of the link to the other. The maximum (default) size of the field is 2048 bits (256 octets) [14, 15, 16].

3.6. Frame check sequence
The frame check sequence (FCS) is a 16-bit (2 octets) number calculated by the sender of a frame. It ensures that the frame was not corrupted by the transmission medium. The FCS is calculated in accordance with recommendations in the HDLC reference document, ISO 3309 [14, 16]. The FCS will be recalculated in the receiver. If the FCS does not match with the transmitted value, the frame will be aborted [16]. For this project, the FCS is a CRC calculated using polynomial x^16+x^12+x^5+1 or also known as CRC-CCITT.

4. Modulation
The term modem is a composite word that refers to the two functional entities that make up the device – a signal modulator and a signal demodulator [14]. Modulation refers to the process of transforming information of digital data into analog signals for transmission. Once the data reaches its intended destination, demodulation is performed to transform the analog signal into digital data and recovers back the original information [15]. Figure 2 illustrates the application of a modulator and demodulator at both transmitter and receiver ends.

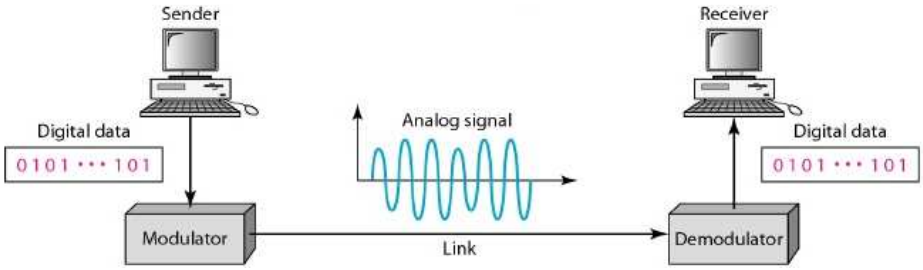

Figure 2. Digital to analog modulation [14]

A wireless modem as in the TNC converts digital data into radio signals at the transmitter and then converts the radio signal back to the digital data at the receiving end [14]. One of the most applicable digital modulation methods to transmit the radio signal using FM radio carrier is the frequency shift keying (FSK).

4.1. Frequency shift keying
In frequency shift keying (FSK), the system starts by generating a Non Return Zero (NRZ) baseband signals. The NRZ signal is where binary 1 is transmitted with square pulse of voltage +V and binary 0 is transmitted with square pulse of -V. Since FSK is based on frequency modulation, it is the carrier that varies in accordance with the information baseband signal. Since baseband signal takes only two values, it follows that frequency of modulated waveform also takes one of two values. The modulation is referred as a keying operation hence, it is known as FSK. In FSK, the carrier frequency is shifted or varied in accordance with the baseband signal. The signal frequency during each bit duration is constant and its value depends on the bit, either 0 or 1. Both peak amplitude and phase remain constant [14, 15]. The FSK signals can be characterized by feeding data into a frequency modulator. The FSK signals are represented by:

$$s(t) = A_c \left[2\pi f_c t + 2\pi k_f \int_{-\infty}^{t} m(t)\, dt\right] \quad (1)$$

where *m(t)* is the baseband signal. The serial data input is binary, hence the resulting FSK signals are called binary FSK signal. The baseband signal is bipolar 0 (-V) or 1 (+V). The integration guarantees the phase is instantaneous. The instantaneous frequency is given by:

$$f_i t = f_c \pm k_f V \quad (2)$$

The maximum frequency deviation Δf is $k_f V$. If the data signal is defined as $d_i(t)$ depending upon either 0 or 1 is being sent, the frequency of the FSK waveform is then given by:

$$f_i t = f_c \pm \Delta f d_i(t) \quad (3)$$

Figure 3 visualizes the concept of an FSK signal. FSK overcome most of the problems caused by noise that might happen in another modulation method such as in amplitude shift keying (ASK) [15].

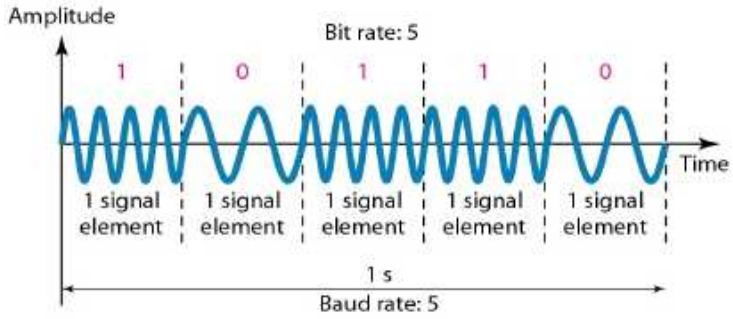

Figure 3. Frequency Shift Keying (FSK) signal [5]

4.2. FSK detection

In terms of demodulation, the non-coherent detection is preferred which does not require estimation of carrier phase. The implementation of FSK leads to orthogonal signaling where each tone is separated and cannot interfere with each other.

$$2\pi (f_1 - f_2)T = 2k_f \pi \quad (4)$$

The frequency spacing of FSK is thus k_f/T with minimum spacing of $1/T$. Simple implementation of the detector is done by using two bandpass filters tuned to each frequencies carrying bit 0's and 1's. The output of the detector is envelope-detected and then baseband-detected. This type of detector simply evaluates which of the two possible sinusoids is stronger and the receiver. The implementation of the detector is shown in Figure 4.

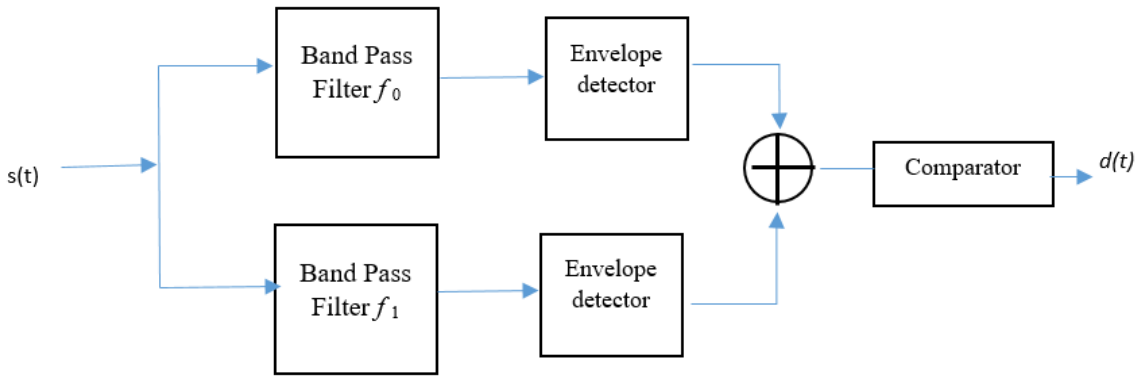

Figure 4. AFSK Non-coherent demodulator

5. Design

The integrated TNC consists of two modules: MCU module and modem module, has been designed as shown in Figure 5. The design of the TNC consists of an Atmega328P microcontroller, power supply circuitry, external clock and USB interface to terminal and PC. The modem module consists of the AFSK receiver and transmitter.

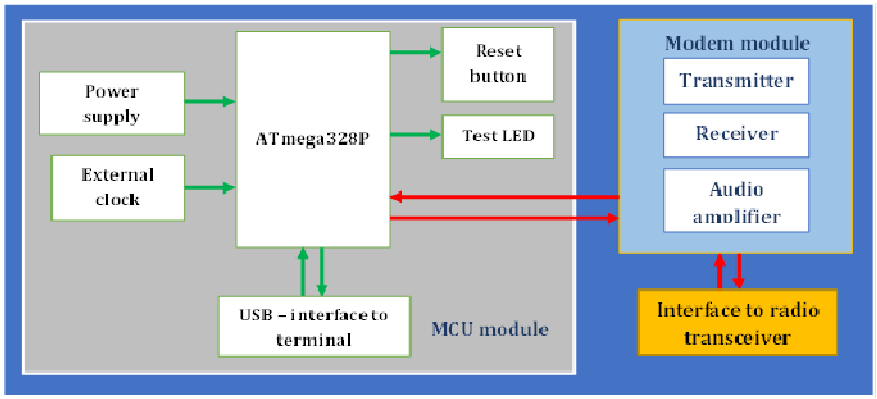

Figure 5. TNC block diagram

After the completion of the hardware design, a printed circuit board is design for the complete TNC board. The PCB is fabricated and use for testing of the modules. Figure 6 shows the printed circuit board of the TNC.

Figure 6. TNC PCB board

5.1. MCU module

The MCU module is powered by a power supply circuitry providing 5V voltages to all the circuit. The system requires a +12V DC input, which will be regulated to +5V. The MCU module interfaces with a USB to TTL board for interfacing with the terminal. The TNC MCU module prototype is connected to the terminal via USB Type-A cable. The interface enables hex file to be uploaded to the ATmega328P IC via the USB – TTL serial connection. The board pins must be correctly connected to TXD, RXD, VCC, +5V and ground pins of the ATmega328P IC. In addition to that, reset button of ATmega328P must be pressed while uploading the program's hex file from the IDE to the microcontroller IC. Note that the ATmega328P microcontroller is pre-programmed with the bootloader such that direct upload of program's hex file to the IC is possible. The external clock is a 16 MHz crystal oscillator circuitry.

The MCU module is interfaced with the modem module through the digital Pulse Width Modulation (PWM) output for modulation and the analog input for receiver modulation. Figure 7 provides the detail of the MCU module.

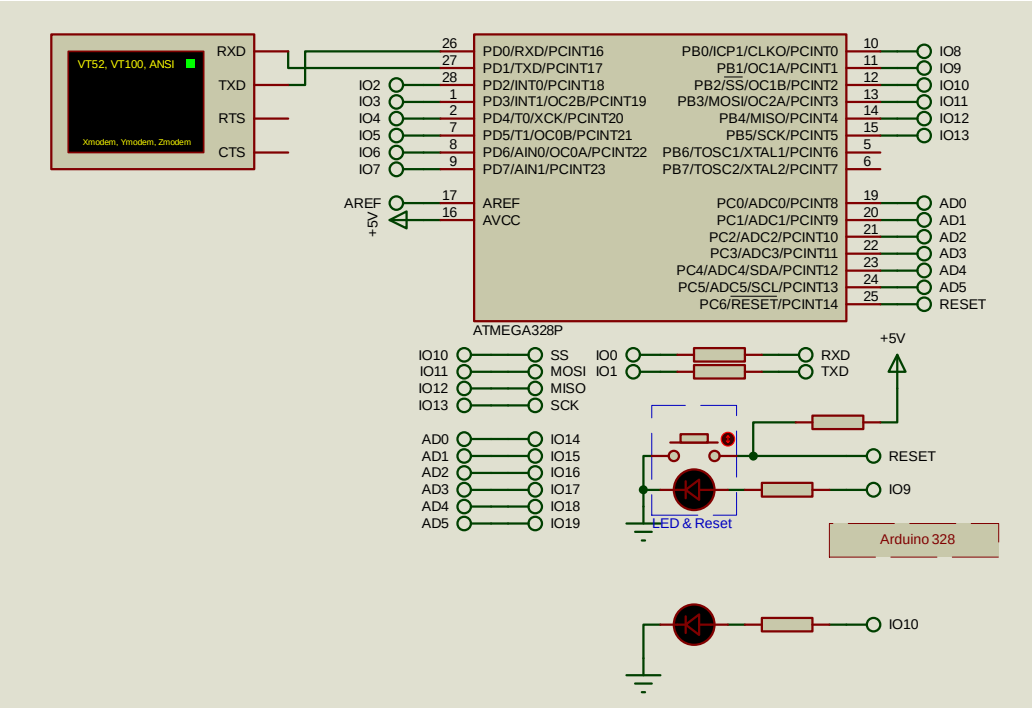

Figure 7. MCU module

The MCU module is validated by testing the software uploading process of the hex file into the Atmega328p via the USB – TTL serial connection. This is done successfully and the TNC software is able to run on the MCU module as shown in Figure 8. A serial monitor software is used to monitor data, which are able to be viewed from PC and the MCU module corresponds to the request made to the TNC.

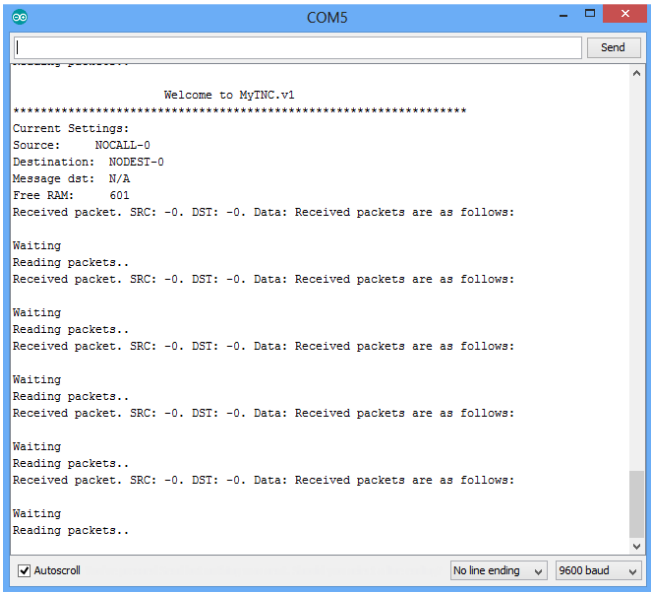

Figure 8. Serial data output from MCU module

5.2. Modem module

The modem module is made up of two parts: transmitter circuit (for modulation) and receiver circuit (for demodulation). The transmitter digital to analog conversion (DAC) circuit is shown in Figure 9 includes several resistors and capacitors.

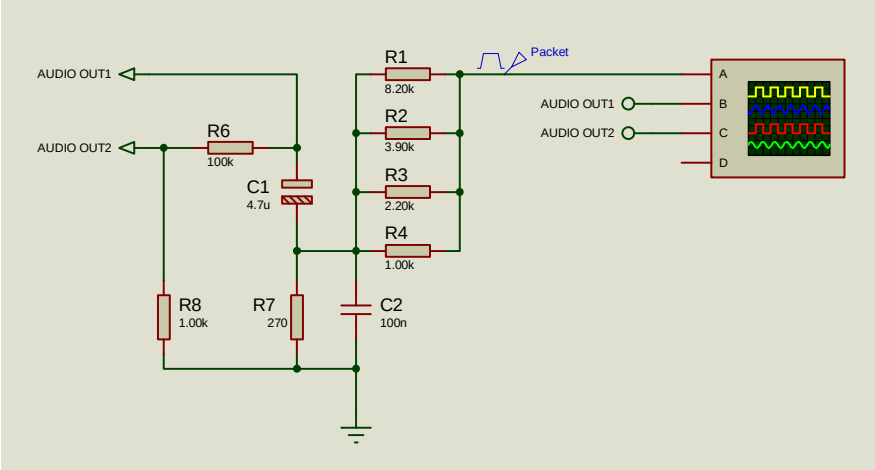

Figure 9. DAC for AFSK modulation

Software modem, which is also known as soft-modem, is implemented in the design. The software is housed in the MCU module. One of the reasons for this is to simplify the hardware modem, hence minimizing the dimensions of TNC board, leading to reduction of the overall costs. Several functions involved in the soft-modem, including digital-to-analog (DAC) in modulation module, and analog-to-digital (ADC) in demodulation module. In Figure 10, the input the digital form is compared with the DAC output.

Figure 10. DAC output

Based on Figure 10, the based band data are converted into analog signals. However, the DAC is not smooth due to the small number of taps implemented. In order to smoothen the signals, the audio amplifier with bandpass filter is developed to be included in the transmitter circuit. The purpose of the amplifier is to boost up the weak signal generated at the audio output and smoothen the conversion effect for a clean conversion of AFSK signals. The band-pass filter functions to allow the AFSK signal to be transmitted in the range of band-pass frequencies from 1200 – 2200 Hz. Figure 11 provides the audio amplifier circuitry.

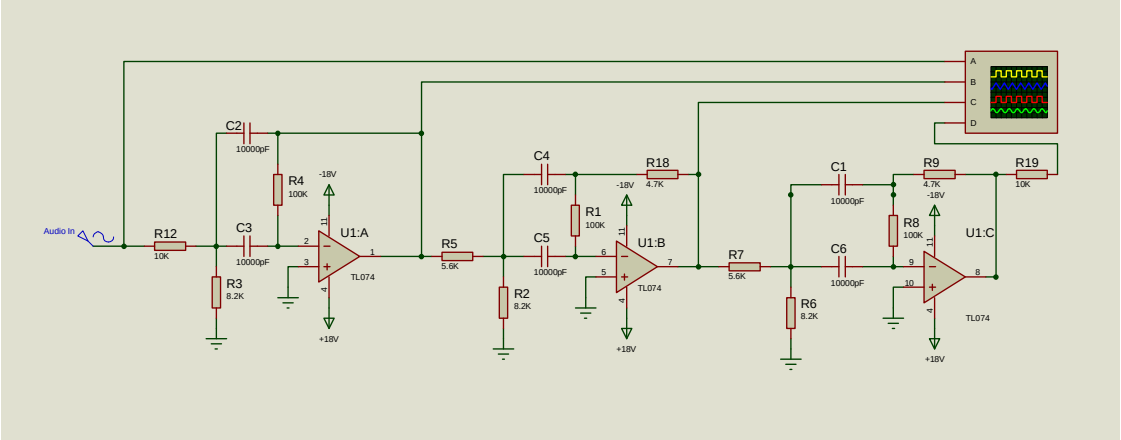

Figure 11. Audio amplifier

Figure 12 shows the output of the modem. The audio amplifier output is connected to oscilloscope to verify the output. It is observed that the modulated data has properly been smoothening by the implementation of audio amplifier and bandpass filter. The two tones of AFSK signals are observed as well.

Figure 12. Modem output

6. AX.25 implementation in the TNC

The A.X25 program functions are divided into two modules: transmitter (packet encoder) and receiver (packet decoder). The source codes for each module are developed based on AX.25 UI frame format structure shown earlier. The transmitter module performs encoding messages (information) into the AX.25 packet forms while the receiver module decodes the AX.25 packets to retrieve the original message. The procedures for both modules are illustrated in the flow chart of Figure 13 and Figure 14, respectively.

6.1. Packet encoder

AX.25 transmitter module implements packet encoding functions like start and end flags insertion, bit stuffing and FCS generation for the CRC checksum. Based on Figure 13, packet encoding procedures are performed as follows:

- While the transmitter module is not reading data from input line, the flag insertion function will continue asserting flags
- The callsigns of both source and destination are asserted to start the encoding process. The SSID of the last callsign input indicates the addresses are complete and ready for the next step.

- The control bits and the PID bits are then asserted to the input line.
- The data is ready to be inserted into the input line, shifted out bit-by-bit on CLK rising edges.
- While data is being asserted, the CRC function calculates the FCS and the bit-stuffing function performs bitstuffing process.
- Once the stop signal is asserted, CRC function appends the generated 16-bits CRC into the input line and append the end flag.
- The complete packet is stored in the transmitter buffer register - queued for the next modulation process.

6.2. Packet decoder

AX.25 receiver module performs packet decoding functions including flag detection, zero un-stuffing, and CRC checking. Based on Figure 14, procedures for packet decoding can be elaborated as follows:

- The flag detection function updates the receiver module. Once a flag is detected, the receiver bit count is reset and the module is ready to read the packet header (128 bits).
- While the data reception is still going on, the zero unstuffing and the CRC functions are activated. The CRC function calculates the FCS using the same polynomial as applied in the transmitter module.
- The zero unstuffing function monitors the input data in the receiver buffer register. Once a zero that was inserted by bit stuffing process is detected, the shift register and the bit count processes are halted for one clock cycle. This is how a bit-stuffing zero is deleted from the incoming data [16].
- The procedure continues until the end flag is detected. Once detected, the calculated 16-bit (2-octects) FCS is compared with the last 2-octects that were received. If both two octets are different, the RESEND signal is asserted to appeal for a new packet.

7. Conclusion

This paper has provided a useful method in developing a terminal node controller (TNC) for CubeSat. Most of the TNCs implemented in the ground stations are COTs based and this creates an issue of data incompatibility. The incompatibilities are due to the manufacturers are using a different way of AX.25 implementation, different software properties of the TNC microcontroller and the hardware properties of TNC. As the COTs TNC are mostly designed for general radio packet network, a TNC specifically for CubeSat use in the ground station is required to overcome the issues arises with COTS TNC. The AX.25 protocol is discussed with an insight of the UI frame implementation. The fundamental of FSK modulations implemented in the design, as well as overall design and integration of microcontroller with the modem, are also discussed. Test results of the TNC development are presented and discussed. The implementation of AX.25 in the TNC is discussed at the end to complete the overall function of the TNC. Future research should address data integrity, increasing the data rates for higher bandwidth mission requirements, and focus on in-house radio development for an improved data integrity and communication performance.

Acknowledgments

The authors wish to acknowledge financial assistance from the Ministry of Science Technology and Innovation (MOSTI) for the funding of the grant no: E0620C1214 and the research facilities and opportunity provided by Jalinan Teknologi Solutions.

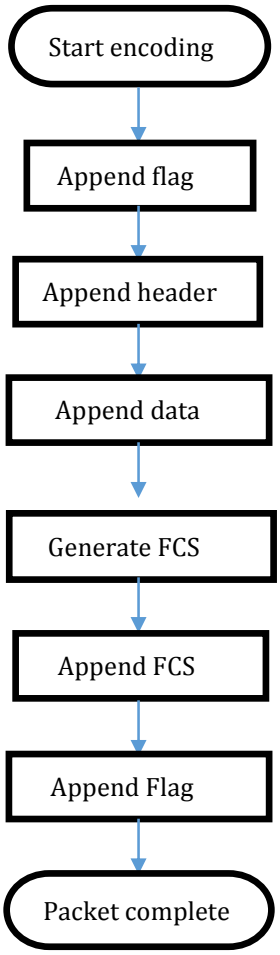

Figure 13: Program flow chart for AX.25 packet encoding

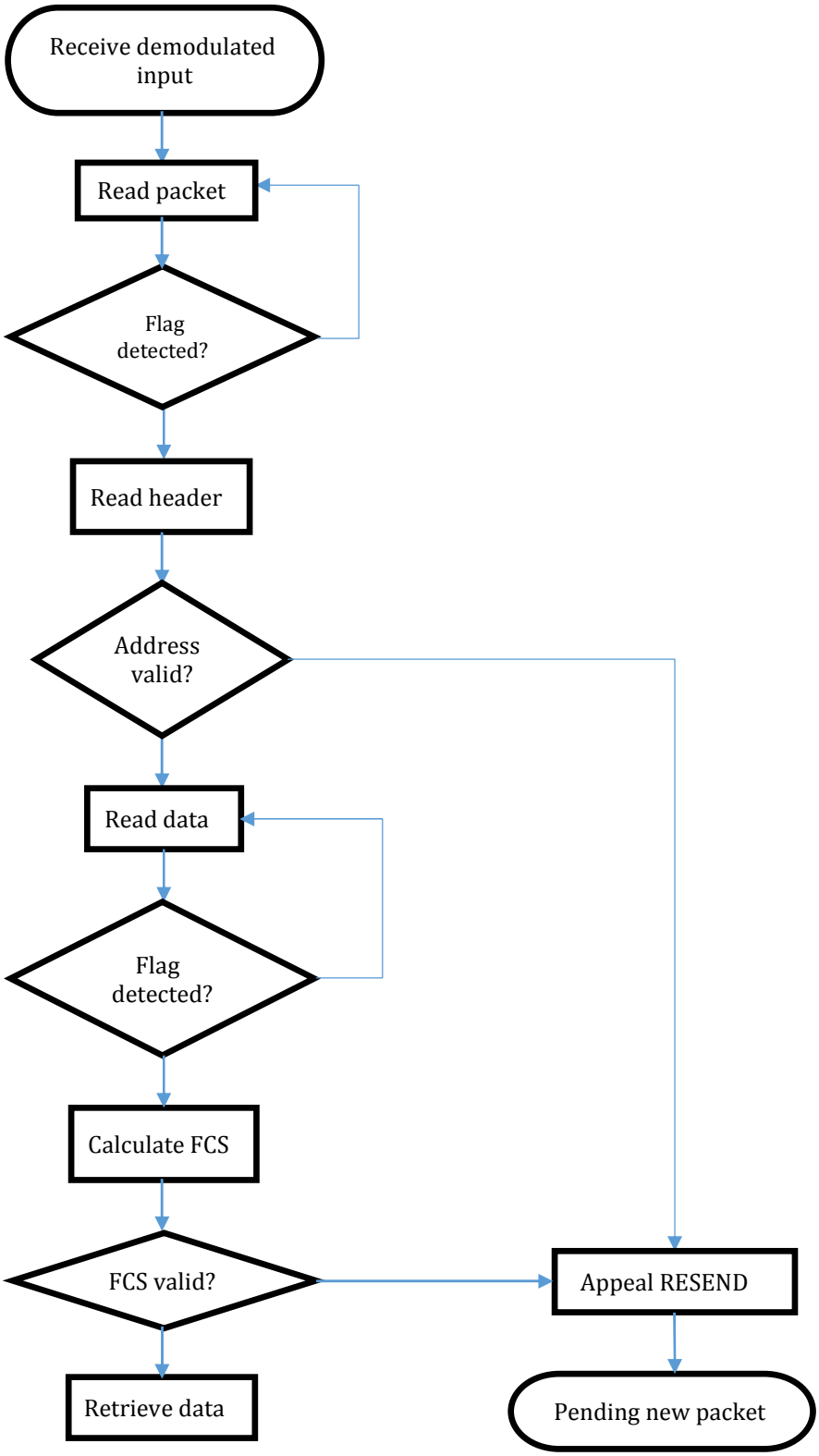

Figure 14. Program flow chart for AX.25 packet decoding

References

[1] Kramer H J and CracknellA P 2008 *Int. J. Remote Sens*. **29** 4285–337
[2] Klofas B and Anderson J 2008 *5th Annual CubeSat Developers Workshop (California)* p1-25
[3] Muri P and McNair J 2012 *J. Commun.***7** 290–308
[4] Klofas B 2013 *10th Annual CubeSat Developers Workshop (California)* p1-12
[5] Perez S, Jarrix S, Roche N J-H, Boch J, Vaille J-R, Penarier J-R, Saleman J-R, and Dusseau L 2009 *23rd Annual AIAA/USU Conference on Small Satellites (Logan)*
[6] Baumann F, Briess K and Kayal H. 2009 *Proc. of the 7th IAA Symposium on Small Satellites for Earth Observation*
[7] Cappelletti C, Martinotti G and Graziani F 2011 *Proc. of the 62nd International Astronautical Congress (Cape Town)*
[8] Dean Straw R 2006 *The ARRL handbook for radio communications* American Radio Relay League. pp. 9.14–9.15
[9] Welwarsky M *6PACK: a "real time" PC to TNC protocol* Retrieved online 2016-08-19
[10] Phil K and Chepponis M 1990 *ARRL 6th Computer Networking Conference (Redondo Beach)*
[11] Zieliński B M 2009 *Proc. Of 9th IFAC Workshop on Programmable Devices and Embedded Systems (Czech Republic)* pp 80-85
[12] Beech W A, Nielsen D E and Taylor J 1998 *AX.25 Link Access Protocol for Amateur Packet Radio* Tucson Amateur Packet Radio Corporation
[13] William A B, Douglas E N and Taylor J 1998 *AX.25 link access protocol for amateur packet radio, version 2.2* Tucson Amateur Packet Radio Corporation
[14] Parry R R 1997 *IEEE Potentials* **16** 14-6
[15] George F and Billeter S 2013 *AX.25 telemetry and telecommand transfer frames format* Swiss Space Center
[16] Huang Y 2012 *Cubesat Ground Station Implementation and Demonstration* Master's Thesis Cranfield University