ME218C 2017 Communications Protocol

1. Definitions

Table 1: General Definitions

Term	Definition
FARMER	A controller capable of interfacing with any of the DOGs. Required inputs are: pairing action, fwd/back, left/right, digital braking and digital special action.
DOG	A remote-controlled hovercraft capable of moving forward/backward, left/right and optionally braking and/or performing a digitally controlled special action.
Pairing action	A command input that will initiate a pairing sequence between the DOG and the FARMER. After this sequence they will be considered “paired”, or actively engaging in targeted addressing, 2-way conversation.

2. Communications Overview

A game for DOGs and FARMERs will consist of multiple rounds lasting 218 seconds each. Each FARMER must be able to interface with any DOG. Successfully pairing with a DOG allows a FARMER to control the DOG for the remainder of a game. This pairing will be disconnected early (before the game ends) if there is a lack of transmission for 1 second. In this case, the FARMER will have to attempt to pair again.

Communication between DOGs and FARMERs will be accomplished using XBee radio modules in API mode. DOGs and FARMERs may not issue messages to each other at a rate greater than 5 Hz. A FARMER will broadcast a pairing request in order to take control of a DOG with a specified tag number. If a DOG is already paired, it will ignore all incoming requests to pair. If it is unpaired, the DOG will then respond with an identification packet, acknowledging the pairing. After this, the FARMER will send a packet containing a 32 byte encryption key to the DOG. All subsequent messages from the FARMER to the DOG will be encrypted control packets. In the event that the DOG decrypts a message that results in an error, it will send a request to the FARMER to reset the encryption index (particular to the specific encryption algorithm used). The DOG will also send report packets to the FARMER containing data from an onboard IMUl. In the event that no transmissions are received for 1 second, the DOG and FARMER will unpair.

The FARMER manages inter-message timing, by a 300 ms timer with regular expiries. Although a 200 ms timer would provide the compliant 5 Hz bandwidth, previous years of ME218 expected message delays of approximately 200 ms. Thus the inter-message delay is set to 300 ms to prevent possible transmission conflicts (e.g. FARMER Xbee transmission jamming a DOG response transmission). Both the DOG and the FARMER manage their own 1 s timers to determine whether communications have been lost. These timers are depicted in more detail in Section 6: State Charts.

3. Hardware Requirements

a. DOG Tag
   The three DOGs in each round are identified by a DOG Tag denoting ID number 1, 2, or 3.
   i. Each FARMER must have an input for the user to select which DOG it will pair with.
   ii. A DOG with no DOG Tag attached should ignore all attempts to pair.

b. IMU Implementation and Axis Geometry
   i. Each DOG will be equipped with an LSM6DS3 6 DOF IMU. The sensor reports linear acceleration along the x, y, and z axes as well as angular rate (in degrees per second) about the x, y, and z axes.
   ii. The orientation of the IMU shall be standardized between all DOGs, as it corresponds both with the camera orientation (relevant to gameplay but not to communications protocol) and how the DOG should interpret commands from the FARMER.
      1. The +x axis runs normal to the camera lens and is interpreted to be the same direction as a command to move forward.
      2. The +z axis is normal to the floor of the game surface.
      3. In forming a right-handed coordinate system, the +y axis is, therefore, defined to point toward the camera's left. This corresponds to a command to move left (>128), described in more detail later.

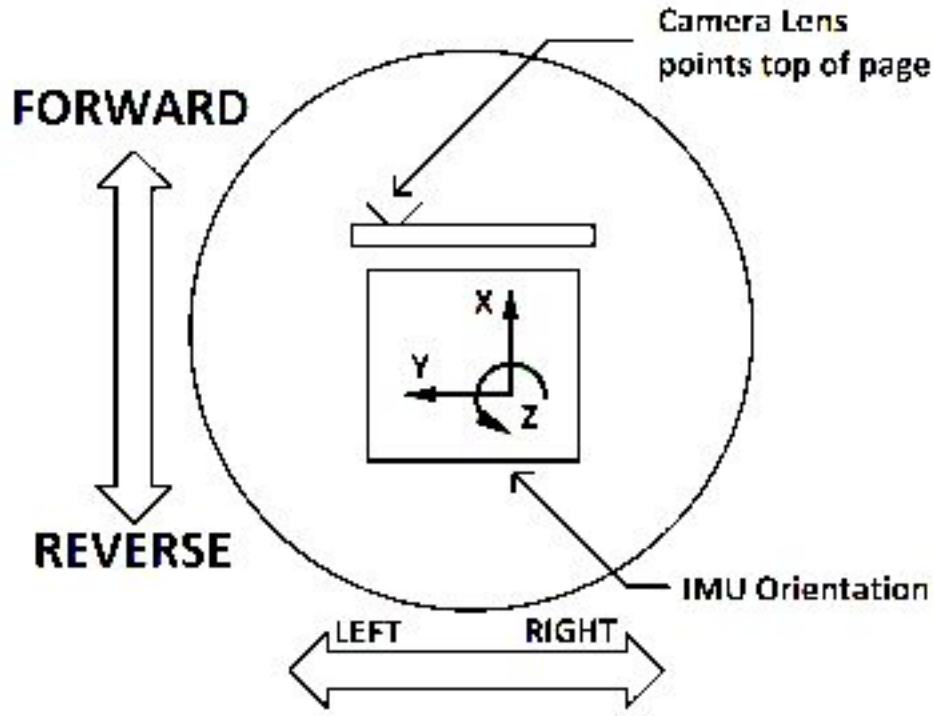

Figure 1: Geometry Axis Definitions

4. Packet Structure

a. Xbee API Structure

The figure below provides reference for the structure of the XBee Transmit (TX) packet used. All the packets described throughout section 4 are contained within the RF Data section of the transmission. The Frame ID must be a nonzero value to assure that the receiving Xbee provides an acknowledgement of receipt to the transmitting Xbee. This acknowledgement is intended for use in individual teams' transmit functionalities, such that a software transmit module will attempt to transmit a message several times until an acknowledgement is generated. In the case of heavy RF traffic, this implementation ensures transmission robustness. The options byte must be set to 0x00 so that no special options are enabled during transmission. Due to these software configurations, an Xbee will provide the sender (Tiva) with an acknowledgement that an RF message at least made it to the destination Xbee. While implementation of this functionality is not explicitly required in terms of this protocol, it is recommended for redundancy in checking message receipt during development.

Transmit request (16-bit address):

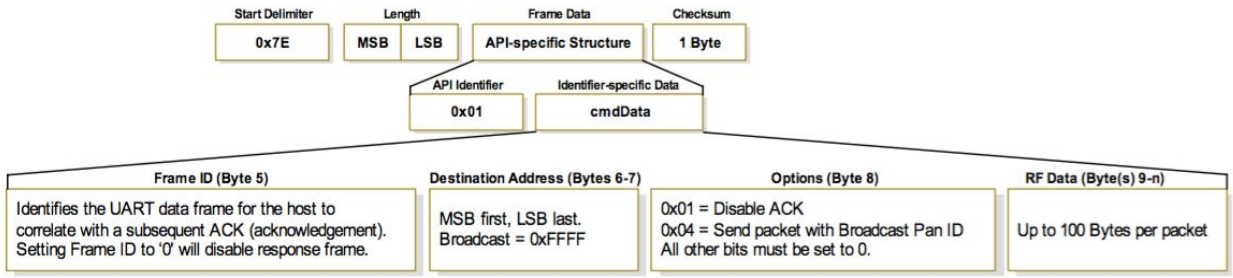

Figure 2: Transmit Request Structure

An outline of the receive packet has also been included below for reference. When the receiving XBee gets an RF packet, the information is output from the XBee's UART module to a microcontroller in the following format.

Receive packet (16-bit address):

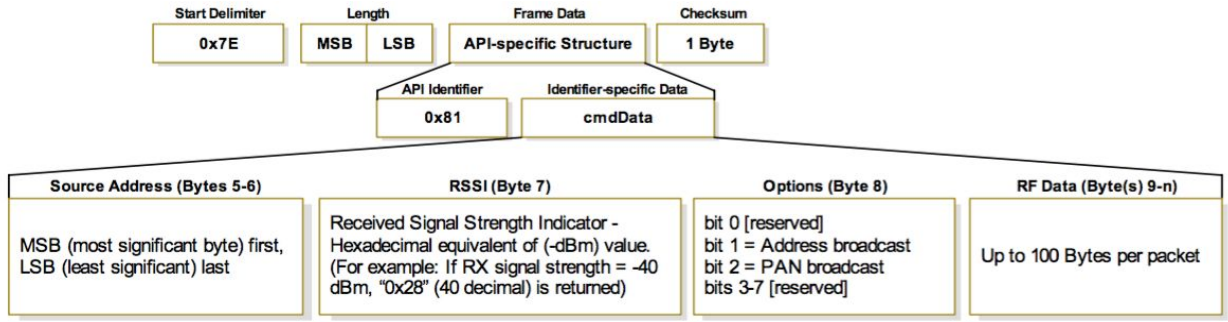

Figure 3: Receive Packet Structure

RF Data structure:

All subsequent packets will be sent in bytes 9-n of the transmission. The number of content bytes sent will vary depending on the packet type. Packet types and their corresponding number of content bytes are outlined in section 4.2.

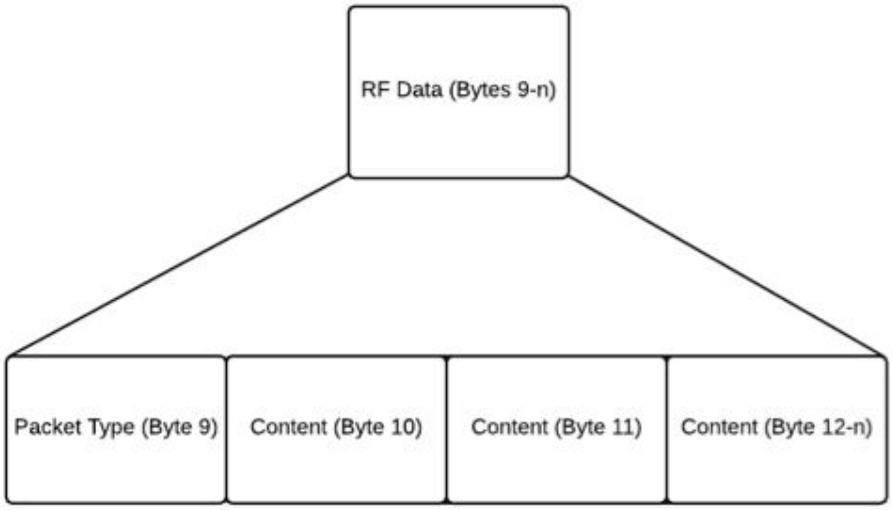

Figure 4: RF Data Structure

b. <u>Class Packets</u>

In the figure above, packet type refers to the specific header (hex value) that denotes each particular type of message (packet type). The first byte in the RF Data portion of every message will be the header, followed by packet-specific content as noted below (if there is any).

Table 2: Packet Types and Contents

Packet Type	**Direction**	**Encrypted**	**Header**	**Data Byte Contents**
REQ_2_PAIR	FARMER → DOG	N	0x01	[DOG_TAG]
PAIR_ACK	DOG → FARMER	N	0x02	- Nothing
ENCR_KEY	FARMER → DOG	N	0x03	[KEY0][KEY1].....[KEY30][KEY31]
CTRL	FARMER → DOG	Y	0x04	[CTRL0][CTRL1][CTRL2]
ENCR_RESET	DOG → FARMER	N	0x05	- Nothing
STATUS	DOG → FARMER	N	0x00	[STATUS0][STATUS1]....[STATUS11]

REQ_2_PAIR

- A broadcast message to all DOGs, containing the desired DOG Tag number (as an 8 bit unsigned integer) to pair with.

- Length: Header + 1 byte = **2 bytes**

Purpose:
The purpose of this message is for a FARMER to announce to all the DOGs that it wants to pair with a specific DOG identified by the DOG Tag number. The address of the FARMER's radio is contained within the Xbee API format. Therefore, the desired DOG, if unpaired, can respond with a PAIR_ACK message directed at the FARMER who sent the REQ_2_PAIR.

Table 3: REQ_2_PAIR Packet

HEADER	DOG_TAG
0x01	0x01, 0x02, or 0x03

Byte 0: [HEADER]
REQ_2_PAIR = 0x01

Byte 1: [DOG_TAG]
The DOG_TAG data indicates the specific DOG to pair with.
- 0x01 = DOG with DOG Tag 1
- 0x02 = DOG with DOG Tag 2
- 0x03 = DOG with DOG Tag 3

PAIR_ACK

- Directed message from an unpaired DOG to FARMER after receiving a REQ_2_PAIR with its DOG tag number
- Length: Header = **1 byte**

Purpose:
The purpose of this message is for the FARMER to receive an acknowledgement back from the appropriate DOG after sending a REQ_2_PAIR message. The DOG will only reply if it is unpaired. Otherwise, the DOG will ignore the REQ_2_PAIR message and not respond with a PAIR_ACK.

Table 4: PAIR_ACK Packet

HEADER
0x02

Byte 0: [HEADER]
PAIR_ACK = 0x02

ENCR_KEY:

- Directed message from the FARMER to the DOG after the FARMER has received PAIR_ACK
- Length: Header + 32 bytes = **33 bytes**

Purpose:
The purpose of this message is for the FARMER to pass an encryption key to the DOG so that all subsequent CTRL messages can be encrypted and properly decrypted using the rotating XOR cipher (see Section 5). The encryption key is 32 bytes of randomly generated integers (uint8_t) that will be constructed by the FARMER and sent to the DOG via the ENCR_KEY message.

Table 5: ENCR_KEY Packet

HEADER	**Encryption Key**				
0x03	Key[0]	Key[1]	Key[2]	.....	Key[31]

Byte 0: [HEADER]
ENCR_KEY = 0x03

Bytes 1-32: [Encryption Key]
The 32 byte encryption key (**least significant byte first**)

CTRL:

- Directed message from a FARMER to its paired DOG.
- This message will be encrypted using the rotating XOR cipher (Section 5).
- Length: Header + 3 bytes = **4 bytes**

Purpose:
The purpose of this message is for the FARMER to pass control data to the DOG with which it has paired. All bytes of the associated Xbee Transmission's RF Data portion will be encrypted, including the header.

Table 6: CTRL Packet

HEADER	**FORWARD/BACK**	**LEFT/RIGHT**	**DIGITAL**
0x04	1 byte	1 byte	1 byte

Byte 0: [HEADER]
CTRL = 0x04

Byte 1: [FORWARD/BACK]
An unsigned 8-bit integer for analog forward and backward control. A value of 255 corresponds to maximum control effort forward. A value of 0 corresponds to maximum control effort backwards. A value of 127 corresponds to no effort.

- 255 = Maximum forward control effort
- 0 = Maximum backward control effort
- 127 = No effort

Byte 2: [LEFT/RIGHT]
An unsigned 8-bit integer for analog left and right control. A value of 255 corresponds to maximum leftward control effort. A value of 0 corresponds to maximum rightward control effort. A value of 127 corresponds to no effort.

- 255 = Maximum leftward control effort
- 0 = Maximum rightward control effort
- 127 = No effort

Byte 3: [DIGITAL]
An unsigned 8-bit integer with bit fields reserved for different functions.

Table 7: Digital Control Byte Register

Digital Control Byte							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
unused	unused	unused	unused	unused	unused	brake	peripheral

- Bit 0: Peripheral
    - Any time a control packet is received with this bit set HI, the DOG shall toggle its peripheral action (if it has one). The DOG shall monitor the status of its peripheral and may initialize it in any way, as long as the DOG responds appropriately to receipt of the toggle bit
    - 1 = Toggle peripheral
    - 0 = Do nothing
- Bit 1: Brake
    - A DOG shall apply its brake (if it has one) when this bit is set. When this bit is cleared, the DOG shall disengage its brake (if it has one).
    - 1 = Brake applied
    - 0 = Do not apply brake

ENCR_RESET

- Directed message from DOG to FARMER

- Used to reset the encryption index for DOG and FARMER if the DOG detects an error in the encrypt/decrypt process.
- Length: Header = **1 byte**

Purpose:
The purpose of this message is for the DOG to initiate a reset of the encryption index if it detects an error in encryption synchronization. In the paired state, the DOG will only receive CTRL messages from the FARMER. Therefore, if the DOG decrypts a message to yield a packet with a header that is not 0x04, the DOG will assume an error in the encrypt/decrypt process and will send an ENCR_RESET message to the FARMER. Upon sending the message, the DOG resets its encryption index to 0. When the FARMER receives an ENCR_RESET message, it will reset its encryption index to 0 as well.

Table 8: ENCR_RESET Packet

HEADER
0x05

Byte 0: [HEADER]
ENCR_RESET = 0x05

STATUS:

- Directed message from the DOG to FARMER after having successfully paired and transferred the encryption key
- This message is not encrypted
- Length: Header + 12 bytes = **13 bytes**

Purpose:
The purpose of this message is for the DOG to transfer the IMU data to the FARMER. It also serves as an acknowledge to CTRL messages received by the DOG. The DOG will send the STATUS message after every CTRL message that it receives from the FARMER. In the register diagram below, MSB denotes “most significant byte”.

Table 9: STATUS Packet

HEADER	**Accel x**		**Accel y**		**Accel z**		**Gyro x**		**Gyro y**		**Gyro z**	
0x00	MSB	LSB	MSB	LSB	MSB	LSB	MSB	LSB	MSB	LSB	MSB	LSB

Byte 0: [HEADER]
STATUS = 0x00

Byte <1:2>: [Accel x]
An unsigned 16-bit integer containing the IMU data for acceleration in the x-direction. MSB is sent first.

Byte <3:4>: [Accel y]
An unsigned 16-bit integer containing the IMU data for acceleration in the y-direction. MSB is sent first.

Byte <5:6>: [Accel z]
An unsigned 16-bit integer containing the IMU data for acceleration in the z-direction. MSB is sent first.

Byte <7:8>: [Gyro x]
An unsigned 16-bit integer containing the IMU data for rotation about the x-axis. MSB is sent first.

Byte <9:10>: [Gyro y]
An unsigned 16-bit integer containing the IMU data for rotation about the y-axis. MSB is sent first.

Byte <11:12>: [Gyro z]
An unsigned 16-bit integer containing the IMU data for rotation about the z-axis. MSB is sent first.

5. Encryption Protocol

Every Control Packet (0x04) message sent from the FARMER to the DOG must be encrypted using the encryption key. Encryption will be applied to the header byte and all subsequent data bytes in the packet. To get an encrypted data byte, simply XOR the encryption key byte [i] with the data byte, where [i] is a tracker variable that initializes to 0 at the beginning of a game. After encrypting one byte, the FARMER will increment i. Once the index reaches [i = 31] (the end of the encryption key), the next index needs to roll over from [i = 31] back to [i = 0]. To decrypt a message, the DOG will simply XOR the encrypted data byte with encryption key byte [i] to get back the original data byte, and will then increment its own index so it is ready to decrypt the next byte in an orderly fashion.

For example, if a message from the FARMER to the DOG is 9 bytes long (byte 0 - byte 8), the FARMER will XOR the first byte of the message with byte 0 of the encryption key and so on until byte 8. The FARMER will then begin encrypting the first byte of the next message it sends to the DOG with Byte 9 of the encryption key, until it has incremented its index to 31, then the index will roll over to 0.

Once the DOG is paired, if it receives a message with a header that is not 0x04, then encryption synchronization has be lost and the DOG needs to send a “reset index packet” to the FARMER (0x05). In this case, the index of both the DOG and the FARMER need to be reset back to 0.

To establish encrypted communications follow the steps below. This sequence is also described visually by Sequence Diagrams immediately following the numbered list and by Section 6: State Charts.

Farmer to DOG:

1. Broadcast Request to Pair Packet (0x01)
2. Upon receipt of DOG ID Packet (0x02), send Encryption Packet (0x03) to the DOG that send the ID Packet
3. Upon receipt of a Standard DOG Report (0x00), send an Encrypted Control Packet (0x04) to the currently-paired DOG

DOG to FARMER:

1. Upon receipt of a Request to Pair Packet containing current DOG Tag address, send DOG ID Packet (0x02) to the FARMER that sent the Request to Pair Packet
2. Upon receipt of an Encryption Packet (0x03), send a Standard DOG Report (0x00) back to the FARMER that sent the Encryption Packet
3. Decrypt incoming Encrypted Control Packets (0x04) as they are received, and respond with standard DOG Reports (0x00)

Control Packets (0x04) from the FARMER to the DOG are always encrypted using the method described above. No other message types are encrypted.

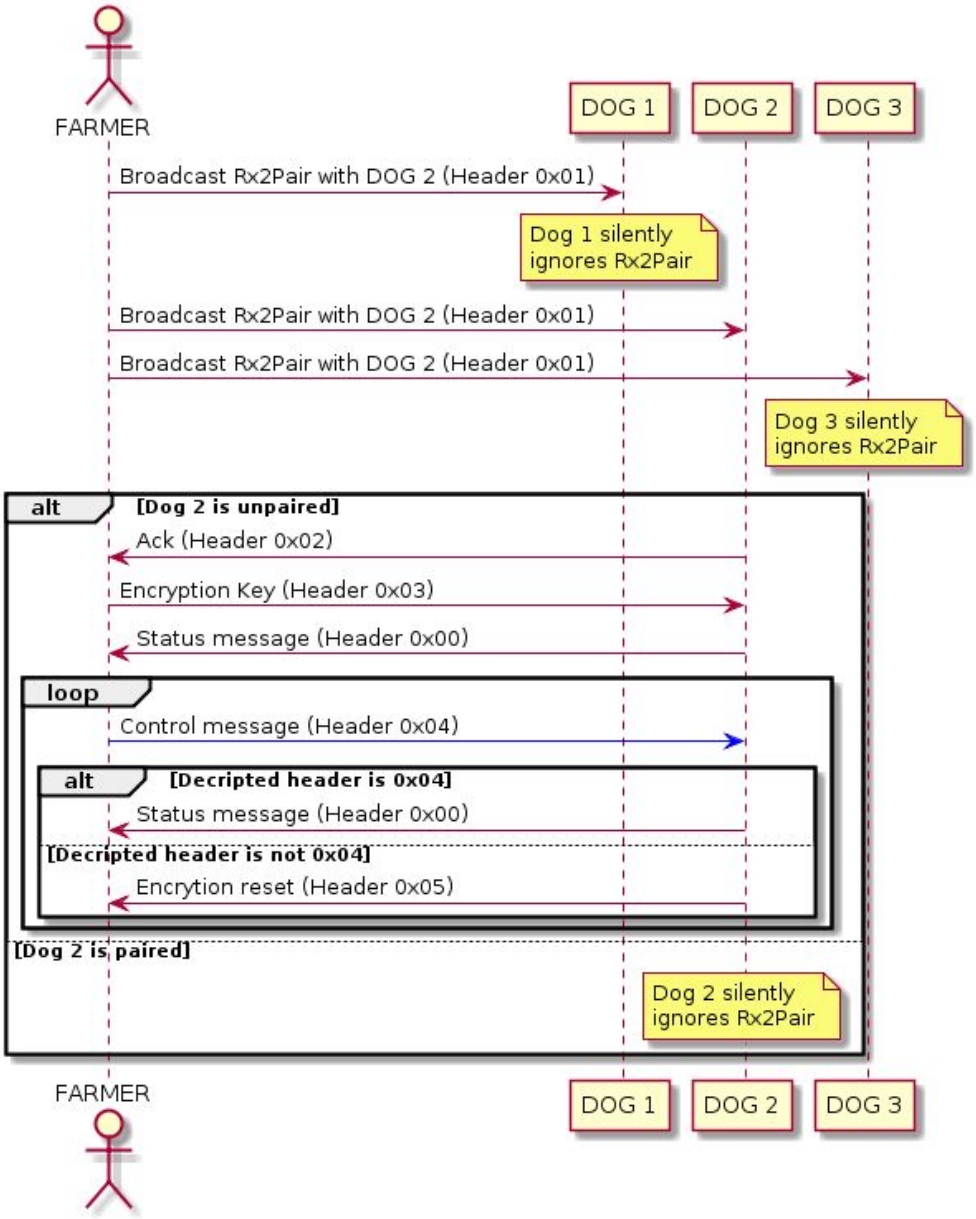

Figure 5: Sample Communications Sequence

NOTE:
1. Red arrow indicates unencrypted message and blue arrow indicates encrypted message.
2. For every message transmitted, if there is no response within 1s, the dog and farmer return to unpaired state.

Additional sequence diagrams for a non-nominal process (error-handling) are included in Section 7: Error-Handling Sequencing

6. State Charts

a. FARMER

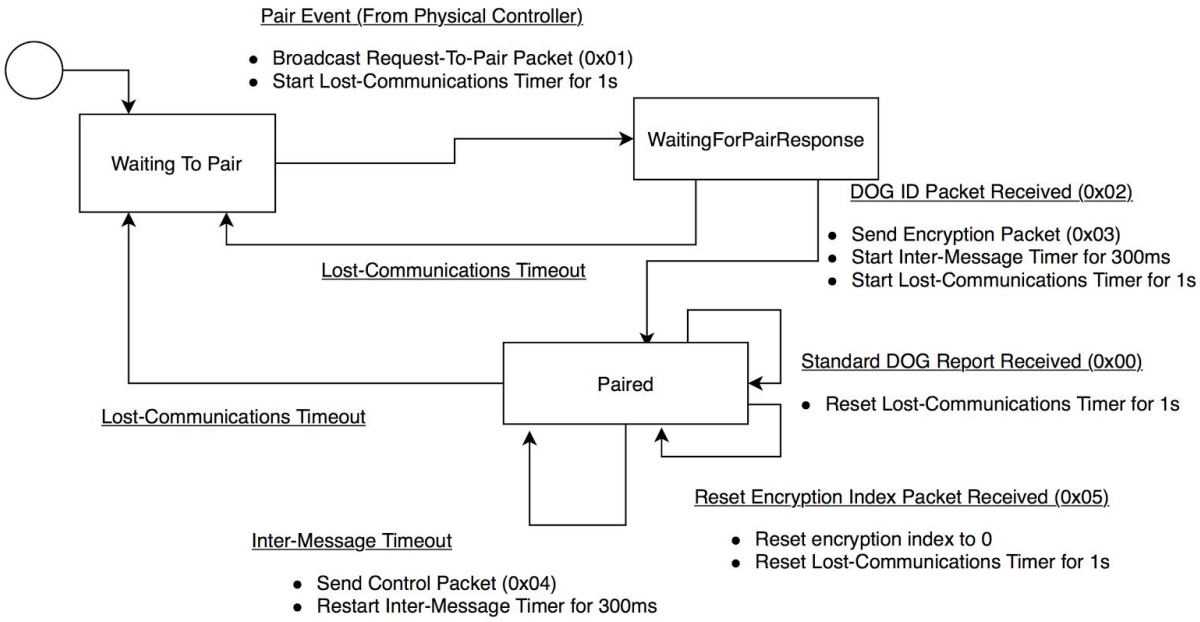

Figure 6: FARMER State Chart

b. DOG

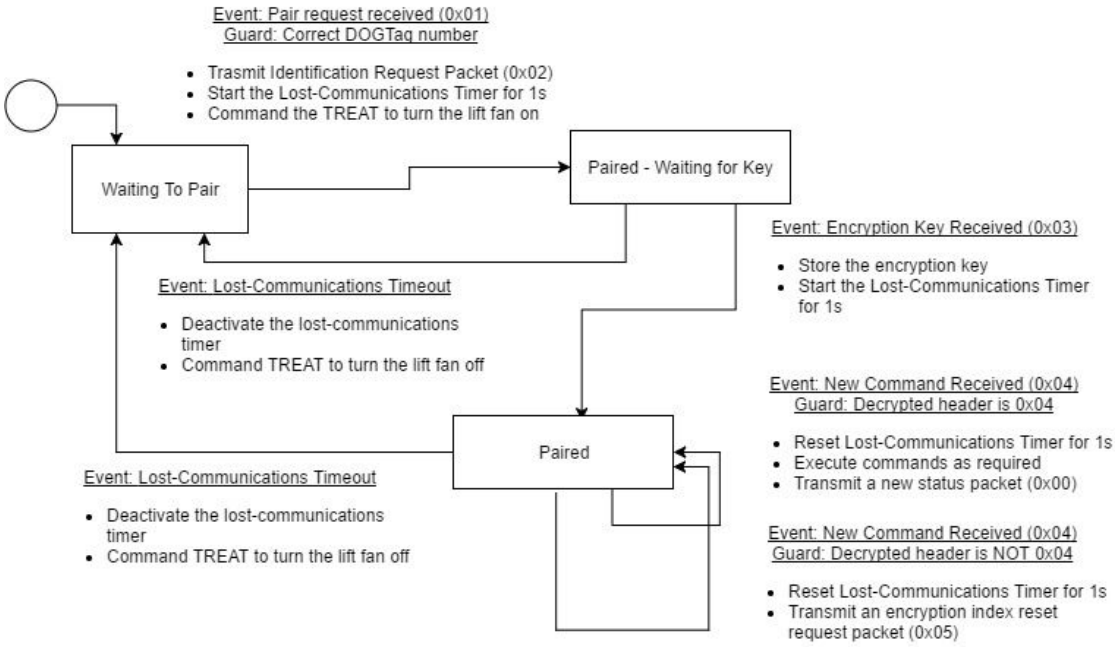

Figure 7: DOG State Chart

7. **Error-Handling Sequencing**
    a. **1 Second Timeout During Communication: Unpair**

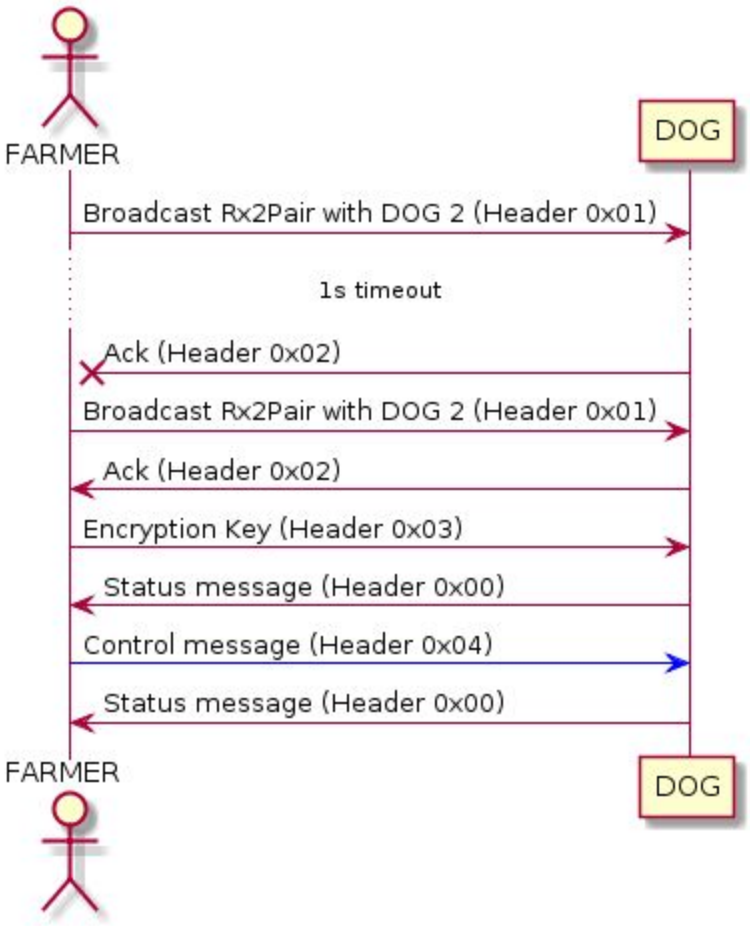

Figure 8: Communication Timeout Immediately Following Pair Request

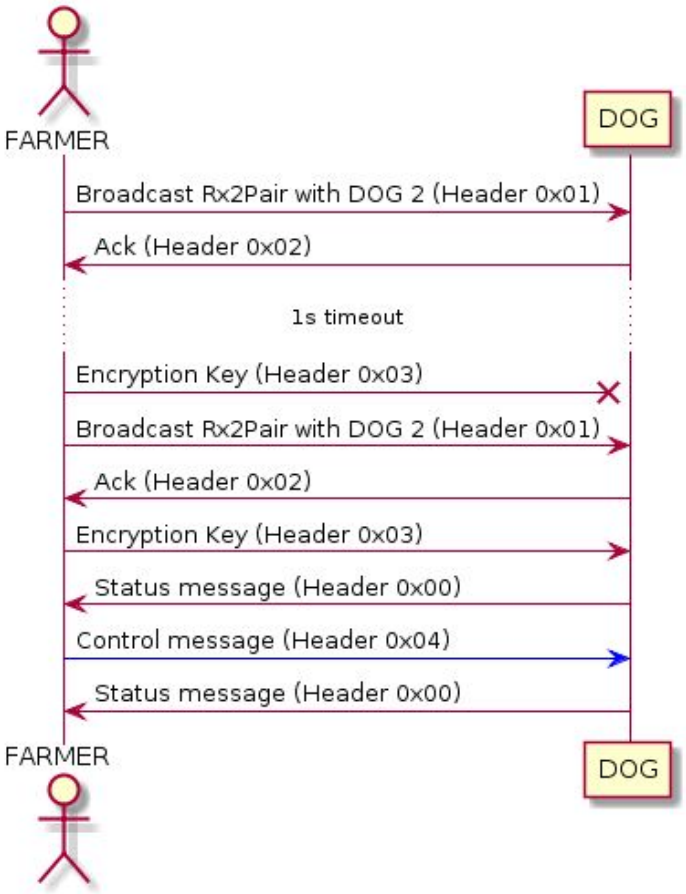

Figure 9: Communication Timeout Immediately Following ACK from DOG

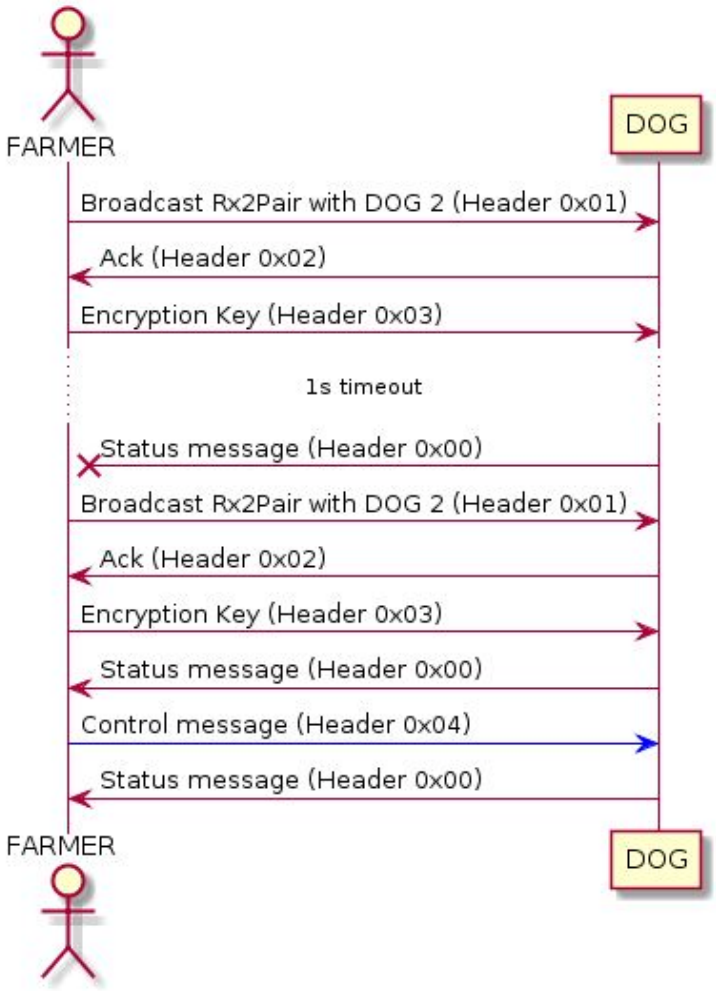

Figure 10: Communication Timeout Immediately Following Encryption Key

b. Header Decrypted not 0x04: Reset Encryption

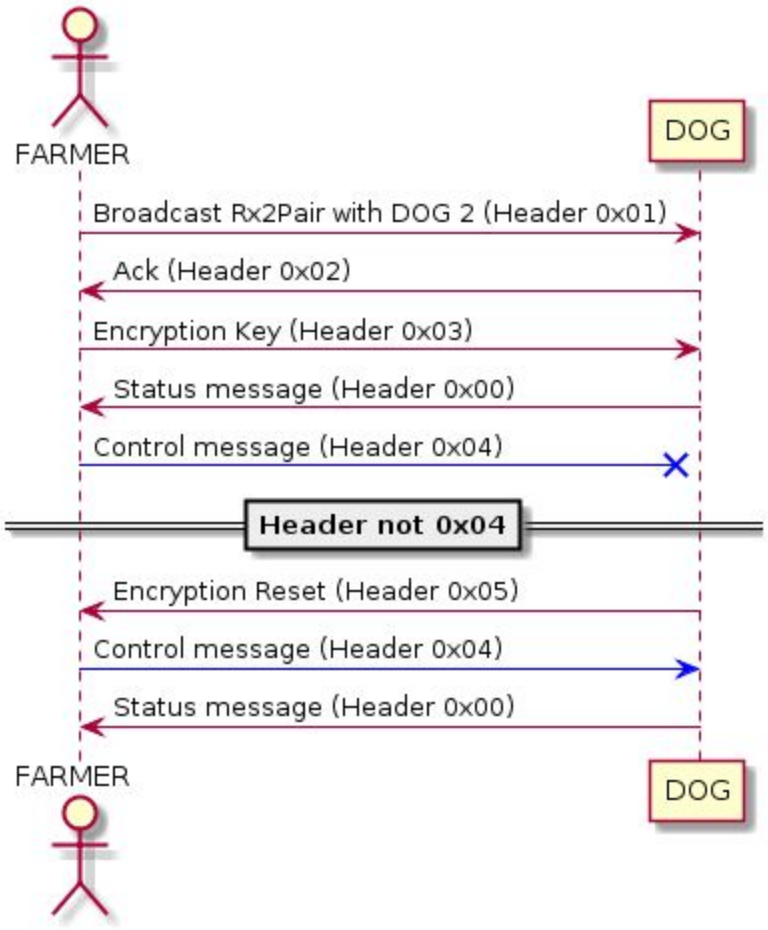

Figure 11: Encryption Reset Following Incorrect Decryption

8. Additional references

Error-handling is also discussed in the 2016 ME218 class protocol. For users with access to the "Testarossa" drive, communications scenarios are also discussed in:

U:_Samples\CommProtocolSpecArchive\Spring2016\CommunicationsScenarios.doc

The Spring2016 directory also contains the 2016 protocol in .docx form. The 2016 protocol was consulted as a primary reference for the construction of this document, and is recommended as a supplement to the 2017 protocol.

9. COVENANT Remarks

This protocol was produced by the Communications Oligarchy for Virtual Electronic Network for Asynchronous Network Things:

Table 10: COVENANT Representatives

Team Number	Official Representative
1	Ian de Vlaming
2	Alex Nathan Kahn
3	Jason Flahie
4	Isabel Gueble
5	Antoine Amon Junior
6	Seth Charles
7	Hang Yang
8	Brittany Hallawell
9	Iskender Kushan
10	Zach Chase
11	Roshena MacPherson