DESIGN REVIEW III

To: Prof. Kyle Winfree EE486C

From: FF1RR

Date: April 3rd, 2020

Re: DR3 for FF1RR

Dear Dr. Winfree,

Very respectfully, the document attached is our Design Review III, in this document we provided overview of project, we also show some details and the challenges that we are facing about our project.

Sincerely

FF1RR Group

FF1RR TEAM

DOCUMENT'S TITLE

Date: 04/03/2020

Project Sponsor:	IEEE and SICCS
Faculty Mentors	Truong Xuan Nghiem Trong Doan Nguyen
Team Members	Cheng Che Zhengjie Xuan Yawen Peng

CONTENTS

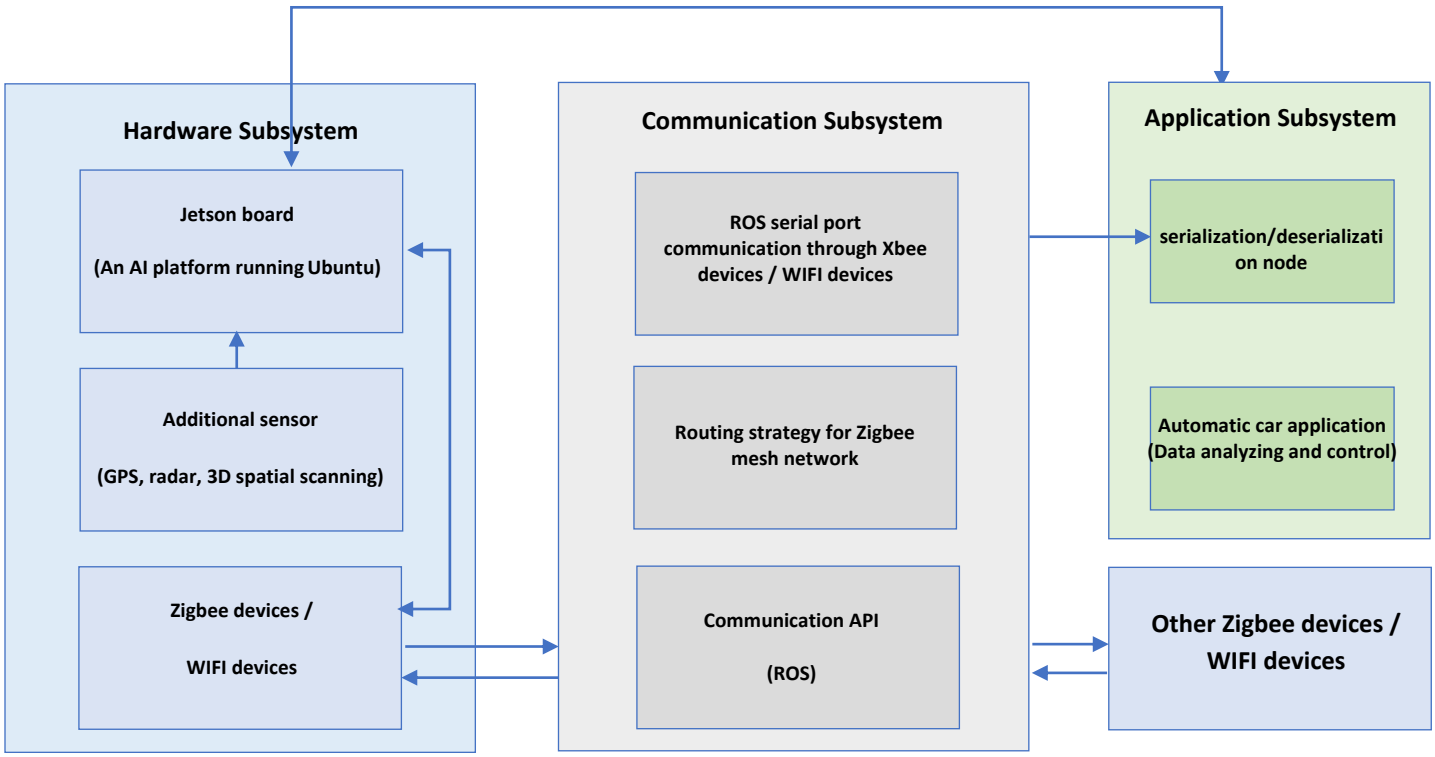
SECTION I: INTRODUCTION PROBLEM STATEMENT

The existed F1/10 autonomous car platform do not have V2V (vehicle to vehicle) and V2I (vehicle to infrastructure) communication capabilities. Each car is independent from other cars and from the infrastructure. To enable research in this area, this project aims to extend the current F1/10 platform with V2V and V2I communication. Our main task is to select one or several hardware options to extend the current F1/10 autonomous car platform with V2V and V2I communication capabilities.

Then we need to develop a standard communication API for the newly integrated hardware: The API must be integrated with the current F1/10 software stack, which is based on the Robot Operating System (ROS).

It must support standard communication functions such as broadcasting of information, handshake and connect, synchronization, transmission and receipt of vehicle status between nodes in a vehicle mesh network, etc.

SECTION II: PROJECT STRUCTURE & SCHEDULE



As shown in the figure above, our project consists of three subsystems: Hardware Subsystem,

Communication Subsystem, Application Subsystem.

Brief Introduction of Our Subsystem

Hardware Subsystem

Our work is to ***find appropriate device that follow the Zigbee standard*** and access to the Jetson board interface. After the implementation of the communication function, we need to add some extended functions, so that we need to add more device like GPS, Radar sensor on the Jetson board.

Communication Subsystem

Our aim is to build a complete communication API for Xbee, Since the car runs the Linux ROS software stack, ***we need to write an ROS based communication API directly for the car's platform***, the API must be integrated with the current F1/10 software stack, which is based on the Robot Operating System (ROS).

Plus, VANET (Vehicular Ad-hoc Network) still has some special problems to handle. For example, there must have a Coordinator in a Zigbee network, but the vehicles in VANET are not static. There are always vehicles joining or leaving the network, which requires us to ***design a strategy to replace the current Coordinator*** when it is about to leave the network.

Application Subsystem

Once the communication API is basically developed, we need to start thinking about real applications. First, we need to serialize and deserialize ROS message since the device can only transmit certain data type. Then the communication part for the existed F1/10 basically developed. We can build a small vehicular AD hoc network and test our communication part and refine it.

Communication Subsystem and Application Subsystem is the key point of our engineering task, after the selected hardware we need for the hardware development of the corresponding Communication API, it depends on our understanding of hardware and Communication protocol, and how we are familiar with the car platform (ROS).

The following table shows our team's detailed WBS:

WBS Spring 2020

	Task		***Deliverables***	***Responsibility***	***Alternative People***
1~2	Learning hardware communication protocol & API mode(802.11 & Zigbee) & ROS	Hardware installation	The device was installed on the Jetson board and was operational	Cheng Che	Yawen Peng
3~4		Evaluate future extensions and reserve space	A list of possible devices	Zhengjie Xuan	Cheng Che
5~14		Implement remote control of device parameters (AT Command based on Transparent mode)	Remotely change the identity of Zigbee devices (Coordinator/Router/End point) Remotely change the parameters of Zigbee devices, like PAN ID, Scan Channels and so on.	All	
15~19		Achieve the timing communication of three or more devices (Without ROS)	The three Zigbee devices can communicate normally	All	
20~35		Create sending nodes in the ROS system and subscribe to ROS messages to test the communication	Messages can be communicated via ROS nodes	All	
36~45		Consider the full communication API functionality and test the Xbee API mode instead of Transparent mode	API to achieve simple broadcast function	All	
46~55	Learning Xbee API mode	The API implements unicast	The API implements unicast	All	
56~65		Achieve a relatively complete API function	Through the API can complete a relatively complete communication	All	

66~75		Create a new message serialization node so that any ROS message can be serialized for easy transmission and acceptance	Completes the serialization node and successfully passes the Ackerman_msg test from the simulation program		
76~85		Use the complete car platform to achieve a variety of applications	The application runs successfully and cooperates well between cars	All	
86~95		Using sensor signal to realize a small-scale vehicular AD hoc network system so that the cars in it can drive automatically	Using the realized functions complete a small-scale autonomous driving network	All	

SECTION III: PROJECT DETAILS AND CHALLENGES

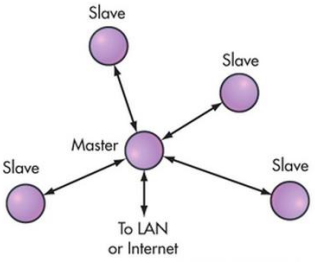
Hardware Subsystem

First, we have three protocol candidates for our hardware they are Zigbee, WIFI and Bluetooth. We finally chose Xbee S2C as our communication hardware.

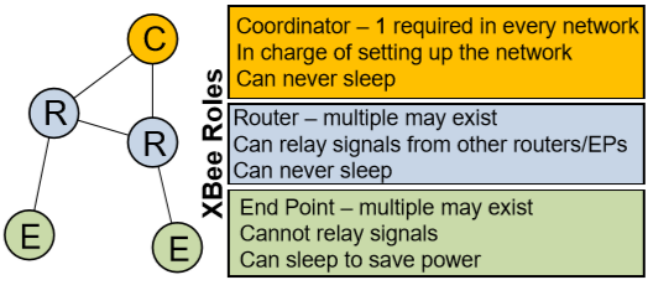
Reasons:

1. WIFI devices need to be in the same network segment to communicate, that mean they need a router to form a network segment. But we don’ t want any additional equipment.

2. Bluetooth is a star topology network, that means this network need a Master node so that devices in there can form a network. Plus, the Slave node can only communicate with the Master node but can’ t communicate with other slave nodes. So Bluetooth protocol is not suitable for our project either.



3. Zigbee protocol is a mesh topology network. It has a coordinator and many routers and end points. Communication between devices in the same network is very easy. It is also very easy for new devices to join the network. Although a coordinator is needed to form the Zigbee network, it is also very easy to replace a coordinator in the network.



Challenges:

1. Due to the cancellation of offline lab, we could not easily access the car platform, this may adversely affect our testing.

2. The car is currently remotely controlled. We do not know whether the communication generated by this system will conflict with our communication system.

3. Xbee has a short communication distance, but this can be easily addressed by upgrading the hardware.

Communication Subsystem

We need to implement two basic functions for our vehicle platform, unicast and broadcast. We hope to achieve these functions through the API mode of Xbee devices.

Start delimiter	Length		Frame data								Checksum
			Frame type	Data							
1	2	3	4	5	6	7	8	9	...	n	n+1
0x7E	MSB	LSB	API frame type	Frame-type-specific data							Single byte

Above is the frame of the Xbee API package, we can control the functionality of the frame by changing the frame data/type segment. For example:

This frame' s type is Transmit Request it allow us transmit data in the RF Data segment. And in the Frame data segment we can send data to specific addressed or we can broadcast to all devices under this network.

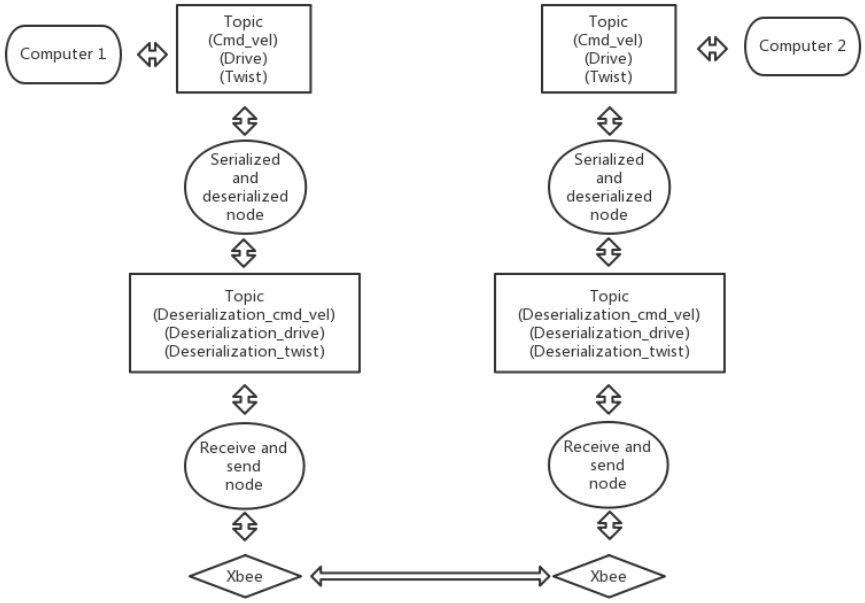
Frame fields		Offset	Example	Description
Start delimeter		0	0x7E	
Length		MSB 1	0x00	Number of bytes between the length and the checksum
		LSB 2	0x13	
Frame data	Frame type	3	0x10	0x10 - Indicates this is a *Transmit Request* frame
	Frame ID	4	0x01	Identifies the data frame for the host to correlate with a subsequent *Transmit Status (0x8B)* frame. Setting Frame ID to '0' will disable response frame.
	64-bit Destination address	MSB 5	0x00	Set to the 64-bit address of the destination XBee The following addresses are also supported: ▪ 0x0000000000000000 - Coordinator address ▪ 0x000000000000FFFF - Broadcast address ▪ 0xFFFFFFFFFFFFFFFF - Unknown address if the destination's 64-bit address is unknown
		6	0x13	
		7	0xA2	
		8	0x00	
		9	0x40	
		10	0xDA	
		11	0x9D	
		LSB 12	0x23	
	16-bit Destination address	MSB 13	0xA6	Set to the 16-bit address of the destination XBee, if known. The following addresses are also supported: ▪ 0x0000 - Coordinator address ▪ 0xFFFE - Unknwon address if the destination's 16-bit address is unknown, or if sending a broadcast
		LSB 14	0xB9	
	Broadcast Radius	15	0x00	Sets the maximum number of hops a broadcast transmission can occur. If set to '0', the broadcast radius will be set to the maximum hops value.
	Options	16	0x00	Bitfield of supported transmission options Supported values include the following: ▪ 0x01 - Disable retries ▪ 0x20 - Enable APS encryption (if EE = 1) ▪ 0x40 - Use the extended transmission timeout for this destination All other bits must be set to 0. Enabling APS encryption decreases the maximum number of RF payload bytes by 4 (below the value reported by NP). Setting the extended timeout bit causes the stack to set the extended transmission timeout for the destination address.
	RF Data	MSB 14	0x48	Up to 255 bytes of data that is sent to the destination XBee
		15	0x65	
		...	0x6C	
		17	0x6C	
		LSB 18	0x6F	
Checksum		22	0x6E	Hash sum of frame data bytes

Challenges:

The API pattern is more complex. We need to package and unpackage the data as required. Plus, the data transmitted by API mode is limited to 255 bytes. Therefore, we still need to use transparent mode when sending data exceeding the limit length. This requires that our communication API include a full dual implementation of API mode and transparent mode.

Application Subsystem

1. We need to serialize and deserialize ROS message since the device can only transmit certain data type. We need to think about the future of using this data not just in a programming language or OS format, but across platforms.



2. We need to build a small vehicular AD hoc network and test our communication part and refine it.

Challenges:

1. Now, we have two candidates for serialization Json and Python pickle library.

    Json: This library can convert Python/C++ datatype to Json datatype, so it has very good compatibility. But most of the ROS message has own structure so we can’ t convert them with Json directly, for example, in python we need to convert obj into dict so that we can convert the dict type with Json.

The table below shows the datatype that Json can code and decode.

Python	JSON
dict	Object
list, tuple	array
str	string
int, float, int- & float-derived Enums	numbers
True	true
False	false
None	null

Pickle: This library is a python library. And pickle can convert python datatype very easily. It can convert any Python object into a Python datatype byte.

Although pickle is very convenient in Python, its expansibility is poor. Since pickle is a Python library and it convert Python object into Python datatype. It will be difficult if we need to use data in another programming language.

So, we need to test and communicate with our client to handle it.

2. Due to the cancellation of offline lab, we could not easily access the car platform, this may adversely affect our testing.

SECTION IV: CONCLUSION

We have done most job of our hardware subsystem and communication subsystem. We are working on the serialization part. Due to the cancellation of the offline lab, we need to make an appointment with the client to use the car platform, this may slow down our progress. But we will try our best to overcome the difficulties and complete our project.