0.1 Module Requirements: Communications

The operational scenarios considered place certain requirements on the something system, and on the modules that comprise it.

0.1.1 Functional Requirements

The communication module has certain requirements:

- Both robot and commander are required to be powered up correctly
- The connection in between Minimal Board and XBee unit should be verified before using the communication unit
- The initialization of the software is done by system itself, but wait for welcome message on LCD to disappear is essential
- The XBee unit should be verified before using the communication unit, i.e. the antenna, XBee board, regulated board.

0.1.1.1 Inputs

Since the there are a little information is required to transfer in between robot and commander, such as global variables Max Speed, Max Yaw rate or in which mode the robot should operate in, the communication function takes as less input as possible to maintain the speed of transfer and ease of receive.

- Function call: A function will be called whenever a communication is requested, and for each global variable there is a related communication function, for instance Max speed, whenever maximum is changed through commander, function sendMaxSpeed will be called to complete this transmit.
- Serial Receive interrupt: Serial receive interrupt is hired here as a signal receiver. Whenever a signal has beed transmitted out on one part of the system (either robot or commander), a receive interrupt will trigger to sort this information and store it to correct place on the other part of the system.

0.1.1.2 Process

Here is an example of how communication works, from commander to robot by changing maximum speed

1. The maximum speed is changed by joystick on the commander and shown on LCD, in the software, a function will be called to send this data change to the robot
2. This function will firstly send a number *MaxSpeed*(equal to 252), which indicates this is a start of a transmission of maximum speed data for receiver on the robot to know and get ready, then it will send the real data of new maximum speed, after this a number *MaxSpeedEnd*(equal to 251) will be sent, which indicates the transmission of maximum speed data is over for receiver to get ready for next transmission.
3. On the robot side, the receiver will firstly receive a number *MaxSpeed*(equal to 252), which means it should get ready to receive data for maximum speed(receive pointer to address of global variable maximum speed), then data for maximum speed will be received and stored into global variable maximum speed, then a a number *MaxSpeedEnd*(equal to 251) will be received to indicated this is the end of this communication.

0.1.1.3 Outputs

The output of communication unit is simple, it will only change the value of corresponding global variable, and allow other function to use those global variables.

0.1.1.4 Timing

Baud rate is required to be considered. Since the two XBee units are by default to be paired up, the only baud rate that needs to determine is from minimal board to XBee unit. The baud rate is 9600, so for minimal board with oscillator frequency of 10 kHz, the SPBRG need to be set to 64 to maintain the validation of information transmitted.

0.1.1.5 Failure Modes

The communication will only fail if the functional requirements have not been met, i.e.

- Either robot or commander is not powered up correctly
- The connection in between Minimal Board and XBee unit is not valid
- The initialization of the software has not been done
- There are hardware damage within XBee unit

0.1.2 Non-Function (Quality of Service) Requirements

0.1.2.1 Performance

The communication module performed not too bad, but there are still some issue with it.

- Firstly, it is able to send and receive correct data and store it into correct place, which means the 'start number','end number' concept works.
- In addition, the transmitting and receiving can be done well and quickly, no lag in manual mode when controlling the robot through the commander.
- However, there are some specific problem still exist. The most serious one is the robot can not switch mode using commander, where we are unable to fix this problem since the same method is used for this mode switch as the motor control, but it is not working properly as other data transmission.
- The other problem is that when controlling robot in manual mode, the robot might stop responding to joystick, but it can still be turned off by commander. The reason for this might be there are some software issue that stuck the robot in a *if* or *while* loop.

0.1.2.2 Interfaces

User will not interface any communication software.

0.1.2.3 Design Constraints

There is a major constrain within the design is that the 'start number'/'end number' concept is using number from 255 to 230, so that we need to limit the transmit data to be not greater than 230, otherwise the system will definitely get confuse with it.

0.2 Conceptual Design: Communications

0.2.1 Rationale Design

The rationale design for the communication unit is fairly simple, it is using the idea of start bit and stop bit, where here we call it 'start number' and 'stop number'. It is using a integer number from 230-255 decimal to construct a frame to allow receiver identify this data. The reason for using this method is that receiver could respond quickly to those number input, and not a lot data are required to be transmitted, so this simple framing method should be a most efficient way to transmit and receive data.

0.2.2 Handshake function design

The handshake function means a certain sequence that checking the availability of the communication connection, so in this project, we came up with an idea using timer 0 overflow interrupt to generate this sequence. Every time this interrupt triggers (timer 0 from hex FFFF to 0000), both commander and robot will send out a sequence of data then let the other part bounce the data back to check the availability of the communication and global variable values, then if the connection does not work for a certain time, robot will automatically power off and commander will show connection down. Where in practical, there is a problem with timer 0 overflow interrupt that this interrupt flag never clear, no matter how i clear it, either in software or even in debug mode, it will back set on immediately. We have no idea why would this happen and David said he did not know why is this happen as well, then we give up on this handshake sequence.

0.2.2.1 Responses to identifiable error conditions

For the communication unit, if there are error or random data has been transmitted out, there will be a buffer to contain all these random data so that it will not affect the system. This buffer is a circular buffer, and it will overwrite himself so it could be enough for error data.

0.2.2.2 Responses to identifiable failure conditions

If the handshake function could work, the system would automatically response to the communication failure, unfortunately this could not been done yet.

0.2.3 Assumptions Made

There are several assumption made for this design:

- The XBee module is preset and ready to use
- After the data is transmitted from one XBee board, the other one will automatically receive it
- The program within XBee module will not affect the software we have designed

0.2.4 Constraints on Communication Performance

State any constraints that may prevent the design from satisfying its requirements.