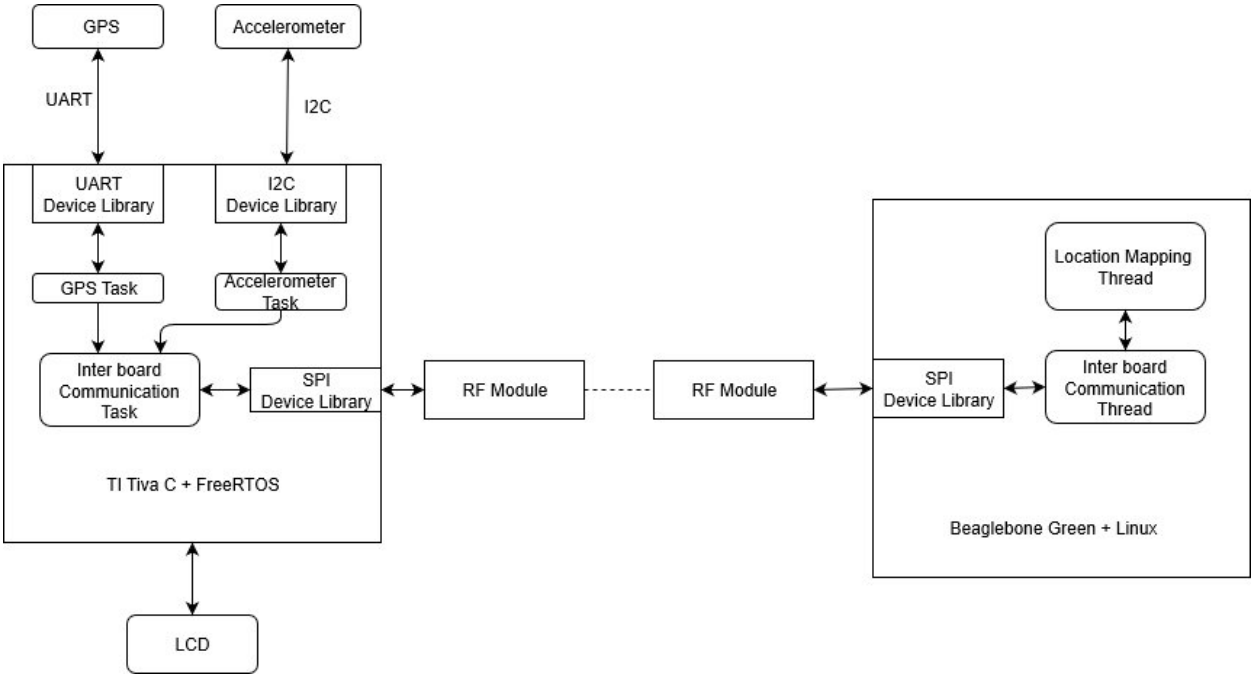
ECEN 5623 – Project 2

Positional Navigation and Accident Detection

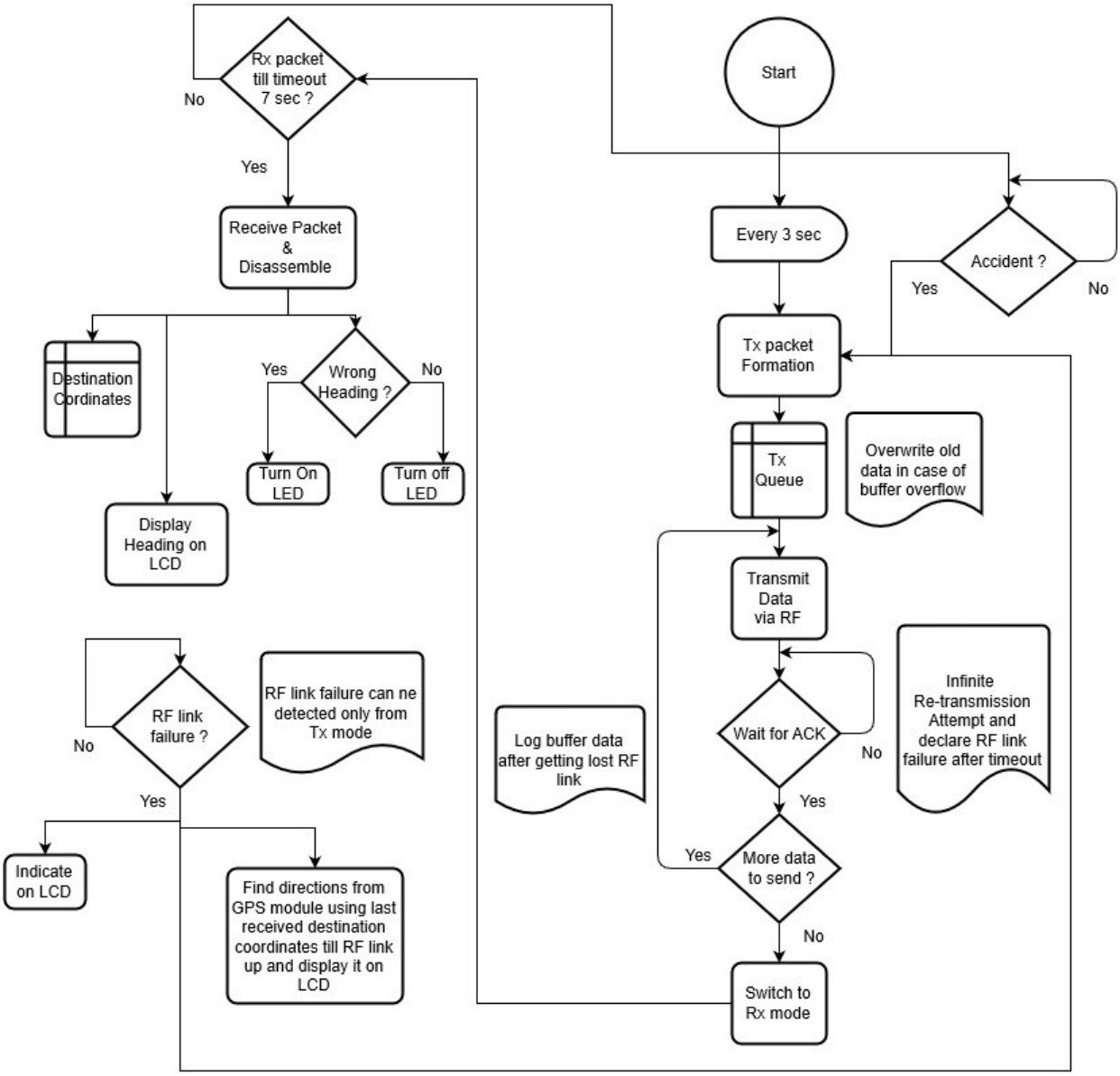
Vatsal Sheth & Sarthak Jain

4/29/19

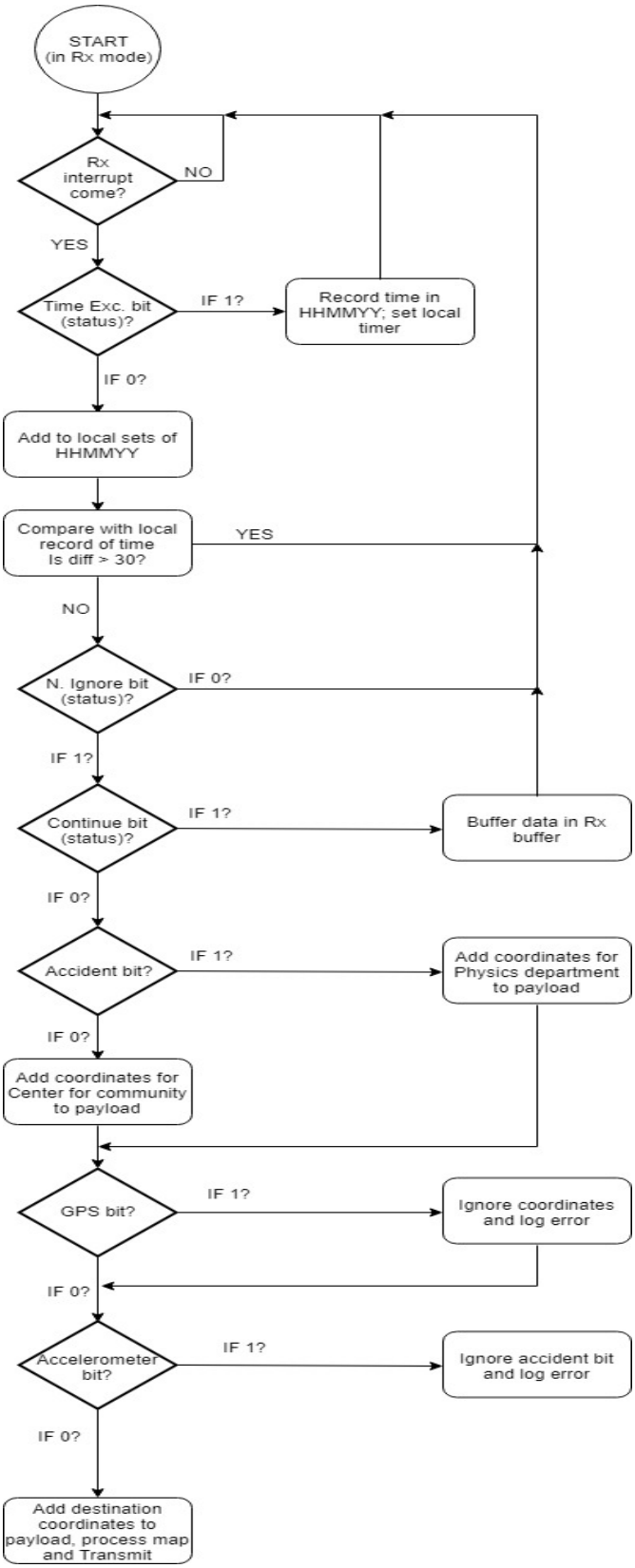
Block Diagram:



Remote Client Communication Flow and Control Diagram:



Control Node Rx Flow and Software Diagram:



Overview

We plan to implement a control model using two nodes, a control node and a remote node. The basic function of the application will be Accident Surveillance and Position Mapping. The remote node uses two sensors, an accelerometer and a GPS sensor. The GPS sensor keeps a track of the current location of the remote node and sends it periodically to the control node. The accelerometer keeps a track of any sudden changes in motion of the car, indicating an accident has occurred. If a sudden jerk has occurred, the remote node notifies the control node via an NRF24L01 transceiver. The control node now maps the position of the nearest hospital and sends it to the remote node over the transceiver. Specifics of the software structure are mentioned below: Software structure: 1. Control Node – The control node shall be the Beaglebone Green. We do not plan on keeping any sensors on the control node, it shall simply be our miniature version of a cloud computer. The control node keeps a mapped database of nearby locations. For the purpose of this application, this database shall be scaled down to a starting location, a target location and the shortest path between the two. In case the Beaglebone receives an accident alert from the remote node, it shall trace the shortest path between the current location of the remote node and the target location (nearest hospital) and send the same to the remote node.

Another functionality on the control node is that of position mapping. The remote node shall initially send a target location, and then periodically send its current location to the control node. The control node shall keep updating directions to the target location for the remote node. 2. Remote Node – The remote node shall be the TIVA TM4C1294XL. We plan on interfacing two sensors with the TIVA board, the GPS sensor and the accelerometer. The GPS sensor can be interfaced via UART, the NRF can be interfaced by SPI and the accelerometer can be interfaced by I2C. In normal operating conditions, the remote node shall send its location to the Beaglebone, and in return expect directions to the target location. When an accident occurs, the TIVA shall expect a location pointing out the nearest hospital. As part of our Tolerance behavior, if the connection between the control and remote nodes is broken, the remote node shall continue to store its location in a buffer, and reattempt connection with the control node. If connection does get re-established, the entire history of path traversed by the remote node will be sent to the control node, which shall display the same, either on GUI or via matrix elements' trace.

Control outputs we plan on using are a 16x2 LCD, which shall be connected on the remote node's side and display the commands of directions being sent to it by the control node. Another output we plan on using is a simple LED, or a buzzer. The LED can be made to blink every time the connection between remote and control is lost, signifying the remote will now begin storing data in the buffers.

Requirements:

1. The control loop mechanism to be implemented shall be of the control node receiving accident notification, mapping the location of the nearest hospital, and sending said location to the remote node.
2. Our plan for tolerance behavior, in which we plan to store the track of path travelled by remote node in a buffer in case of loss of connection. When connection comes back online, this trace of path will be sent to the control node, which shall display the same.

3. The system supports a level of degraded service, in case of failure of one or more services, the system will continue to operate. For eg, if the accelerometer stops working, the system can still navigate with the help of control node, albeit with the loss of an accident surveillance system.

Test Plan & Result:

- GPS Sensor GPSNEO6M:
  1. Co-ordinates Test: In this the current location co-ordinate are accessed over a period when the signal with satellites is stable and more than 3 satellite links is acquired and displayed using UART on terminal. This co-ordinate are then verified using Google Maps. Code passed this test with enough accuracy even though the values were fluctuating, mainly because of cheap antenna.
  2. Time Test: In this test data from GPS sensor is used to find the UTC time which is then subtracted with -6 time zone for Mountain time. Code passed this test with an accuracy of seconds.
  3. Direction Test: In this test source and destination co-ordinates are provided to API which then returns the compass headings in degree with respect to North. This is used in case of RF link failure as a fault tolerance and degraded behavior to find the directions. To test this two known co-coordinates from Google Maps are given and the direction is then verified. Code passed this test.
- 6 axis Accelerometer & Gyroscope MPU6050:
  1. I2C Test: This test is performed by reading the Who am I register of this sensor which return the Slave address of this sensor which is a known value. Then Config register is written and then read back to verify the write. So first test verifies read operations and second one verifies write. Overall these verifies the operation of I2C drivers.
  2. DMP engine test: Sensor has Digital processing capabilities with programable interrupts. Appropriate registers are written to set the motion detect threshold, duration, counters and interrupt. Then sudden motion emulating an accident scenario is created and interrupt is detected on Tiva to verify the correct programming of this sensor. Documentations of this sensor are real pain, as all version specs doesn't have all register mentioned. It is required to scroll between various versions of specs to search for registers.
- NRF24L01+ RF module:
  1. SPI Test: Registers are read for their reset values to verify read and then they are written and read back to verify write operations of SPI driver code.
  2. Interrupt Test: Transmit payload is written and CE is made high without any receiver nearby, this will raise max transmission attempt interrupt and this verifies the interrupt part of this module.
  3. Transmission Test: With receiver nearby, payload is written and transmit ACK interrupt is checked to verify successful transmission.
  4. Reception Test: With Transmitter nearby transmitting in loop, reception is verified from receive interrupt and then reding the data and verifying it.
- LCD Test:
  Various LCD API are tested by moving the cursor, printing string, integer, float values. Also strings are used to verify correct wrapping of text across lines as DDRAM in LCD controller doesn't have address in order.
- Inter-Task communication using Queue:
  Tests sending and receiving known data across two queue between different tasks is verified.
- Control Node on BeagleBone Green:
  A simple control loop has been implemented using the BeagleBone Green and the TIVA TM4C129XL. The TIVA takes inputs from the surroundings in the manner described above, and sends to the BeagleBone the following data:
  1. Latitude and Longitude data from GPS module
  2. Accelerometer data from sensor

- The boards are communicating using an NRF24L01+ radio frequency module, which is connected via SPI.
- The BeagleBone Green takes its decisions on where to guide the TIVA board using a control logic based on inputs sent by the TIVA, and a map database implemented entirely on the BeagleBone Green. The map. The map has two routes implemented on it, one to The Center for Community, and another to the Physics department, both from the current location of the TIVA board. Testing and Controls used for the mapping were:
    1. In case the TIVA strays from the correct path, the map self-corrects to bring the TIVA back to the original path, and then continues to the destination.
    2. Based on inputs from the GPS module and accelerometer, the BeagleBone Green either ignores data (in case of failure and logs the same), else uses the data to further improve location and accuracy.
    3. Requirements have been supported, as TIVA board maintains a buffer of data in case of link failure and sends the buffered data to BBG on connection re-establishment. Secondly, control loop is implemented in the form of positional mapping, and system supports a degraded level of service.