Software Formalization

Year: 2018 Semester: Spring Team: 16 Project: Track-on-track

Creation Date: 2/21/2018 Last Modified: March 3, 2015

Member 1: Nick Geirland Email: ngeirlan@purdue.edu

Member 2: Nathan McNally Email: nmcnall@purdue.edu

Member 3: Yunsheng Li Email: li1436@purdue.edu

Member 4: Aaron Kaiser Email: kaiser20@purdue.edu

Assignment Evaluation:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Item** | **Score (0-5)** | **Weight** | **Points** | **Notes** |
| **Assignment-Specific Items** | | | | |
| **Third Party Software** |  | x2 |  |  |
| **Description of Components** |  | X3 |  |  |
| **Testing Plan** |  | x3 |  |  |
| **Software Component Diagram** |  | x4 |  |  |
| **Writing-Specific Items** | | | | |
| **Spelling and Grammar** |  | x2 |  |  |
| **Formatting and Citations** |  | x1 |  |  |
| **Figures and Graphs** |  | x2 |  |  |
| **Technical Writing Style** |  | x3 |  |  |
| **Total Score** |  | | |  |

5: Excellent 4: Good 3: Acceptable 2: Poor 1: Very Poor 0: Not attempted

General Comments:

1.0 Utilization of Third Party Software

|  |  |  |  |
| --- | --- | --- | --- |
| Name | License | Description | Use |
| MSP430FR6989 Library | BSD | Standard register and bit definitions for the Texas Instruments MSP430 microcontroller. This file supports assembler and C development for MSP430FR6989 devices. | The MSP430FR6989 Library will be used to program the MSP430 microcontroller. The library will be used to facilitate communication between the microcontroller and the peripherals. |
| Android Development Library | Apache 2.0 License | “The primary purposes of Android are to create an open software platform available for carriers, OEMs, and developers to make their innovative ideas a reality and to introduce a successful, real-world product that improves the mobile experience for users.” [3] | The Android Development Library is the standard library used to program Android devices. This library will be the basis for app portion of the Track-on-track project. |
| Google Maps Android API Utility Library | Google Maps API’s Terms of Service | “Looking for advanced features to add to your maps? The Google Maps Android API Utility Library is an open-source library of classes that are useful for a range of applications.” [1] | The Google Maps Android API Utility Library will be the main tool used to add Google Maps support to the app. Google Maps will be used to plot the location of the Track-on-track device on request. |

2.0 Description of Software Components

The major components of the software will include: a smartphone app, functions to communicate with the cell modem, functions to communicate with the bluetooth module, and functions to communicate with the LCD and other miscellaneous activities.

The smartphone app will be used as the primary way for the user to communicate with the Track-on-track device. First of all, the app will have to prompt the user for all required permission. Once permissions are given, communication can begin. A text which is sent from the device can take several different forms. All request information will be sent in 1 text, which will consist of a series of delimited commands, with the position of the command defining what information or setting is being referred to. Each command will either be set to a value which tells the device to ignore that specific command, or a value which tells it not to ignore that command. The text will be sent by the press of a single button, and the contents of the text will be defined by the user in a secondary menu using a series of checkboxes. SMS messages received by the app will be in a similar format, except they will contain the requested information. Once the information is received, if GPS coordinate data was received, the app will parse it and use a Google Maps plugin to display the current location to the user. Other information such as battery life will be displayed, and changes to the state of the device will be answered by simple values which will refer to whether or not the device was able to change its state successfully or not. Bluetooth will have its own screen, and it will have a graphic to display whether the strength of the bluetooth signal is growing stronger or weaker. In addition, bluetooth will also have some options to send commands to the device.

Inside of the firmware, all code will be developed by us using the library developed by TI that defines register locations for our specific micro. The portions of the firmware that deal with communicating to the cell modem mainly focus over being a middle man between the android app and the cell modem. When the cell modem receives an SMS, it sends a message over UART to the micro which will wake it up from low power mode. From there, the micro should process the request, gather whatever information it will need to return from the cell modem, and send that back to the android app. Most of the time the information it needs to gather will involve either the location information of the device or of diagnostic information like battery level. The android app should also be capable of controlling settings of the device such as controlling the Bluetooth module, LCD, changing other settings in the device. If the request requires information to be returned then the software will format the data and send it back through SMS. This code will have to keep track of the commands that the GSM modem uses which is the Hayes or AT Command set.

The Bluetooth portions of the firmware are very similar to the portions that communicate with the cell modem. The code will be using most of the same functions and even many of the same configurations as both the GSM modem and the Bluetooth module both use UART with a 115200 baud rate. The major difference between the two sections of code is how to decode the communication between each module. The RN4020 module uses its own separate set of commands that are tailored more towards the functions of Bluetooth.

The LCD portion of the firmware is the simplest out of all of them. The LCD uses SPI to communicate with the microcontroller but, unlike the other two modules, isn’t complex enough to have something that can be considered its own command set. The code is also broken up into initialization and loop code. The majority of the code will be focused on keeping a menu that the user can interact with while the LCD is on. This menu will mainly be focused on allowing the user to change various settings around the board and will take input through the pushbuttons on the device. The input pushbuttons involve setting input interrupts and servicing them whenever the onboard switch is no longer in the lock position. The device also has a function to handle playing a loud noise through the speaker on board so it can be found. This function mainly involves setting the PWM of the device on the pin that’s connected to the speaker until we get a request to stop. The device also uses authentication from the user for the SMS portion of the device to prevent anyone from accessing the users location just by knowing their phone number. The device needs to be able to store the authentication passcode and offer the ability to request to change it through SMS or bluetooth.

3.0 Testing Plan

Each software component will need to be tested individual to confirm proper functioning before testing how multiple components function together. The tests will be conducted as follows:

Smartphone App

1. Attempt to send SMS message from app to known phone number with a variety of options set. The output to the other phone will be checked against the theoretical output. This test will be repeated several times in order to ensure all options function as expected.
2. Attempt to send SMS message from phone to app with a variety of options set. This test will ensure that the app can receive and properly display coordinates via the Google Maps plugin.
3. Connect to another bluetooth device, and then move the phone away from and closer to the source of the other bluetooth device. This will test whether or not the app can properly display whether the strength of the bluetooth signal is getting stronger and weaker.
4. Connect to another bluetooth device and attempt to send and receive messages. This test will prove that the app is capable of both sending and receiving information through bluetooth.

GSM Modem

1. Send SMS messages with different requests and test to make sure that the device is capable of responding back.
2. Request for location and check to make sure that it can send the location back through SMS.
3. Request to change settings such as turning Bluetooth, LCD on, or changing the authentication.
4. Request without correct authentication unlike the above three steps.
5. Test all of the above in multiple combinations to make sure they don’t interfere with each other.

Bluetooth

1. Check if the device can connect with an android phone.
2. Attempt to request the device to alert by playing noise through the speaker.
3. Request to change settings such as turning Bluetooth off, turning on the LCD, or changing the authentication.
4. Test the above in multiple combinations to make sure they don’t interfere with each other.

LCD/Miscellaneous

1. Check the lock switch to make sure it can’t be turned on when it’s not meant to be.
2. Make sure the menu can be displayed and changed correctly when the buttons are pressed and unlocked.
3. Test the alert noise and make sure it can make a significant enough noise.
4. Attempt to change the authentication and make sure it can still work with the updated authentication.

4.0 Sources Cited:

[1] “Google Maps Android API Utility Library,” *Google Developers*, 07-Jun-2017. [Online].

Available: <https://developers.google.com/maps/documentation/android-api/utility/>.

[Accessed: 21-Feb-2018].

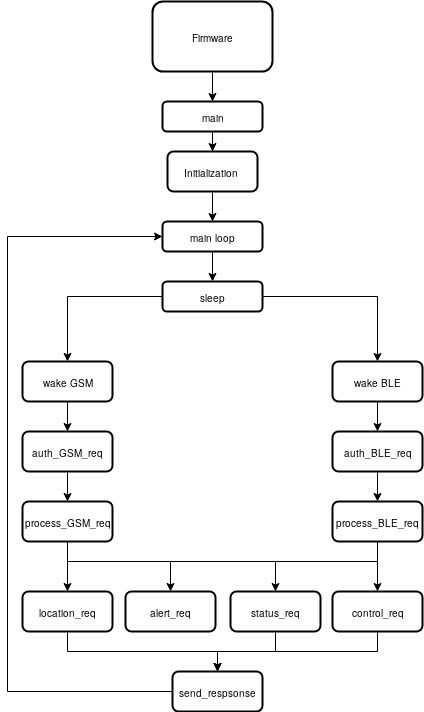
[2] “MSP430FR6989 LaunchPad™ Development Kit,” *Texas Instruments*, Jul-2015. [Online].

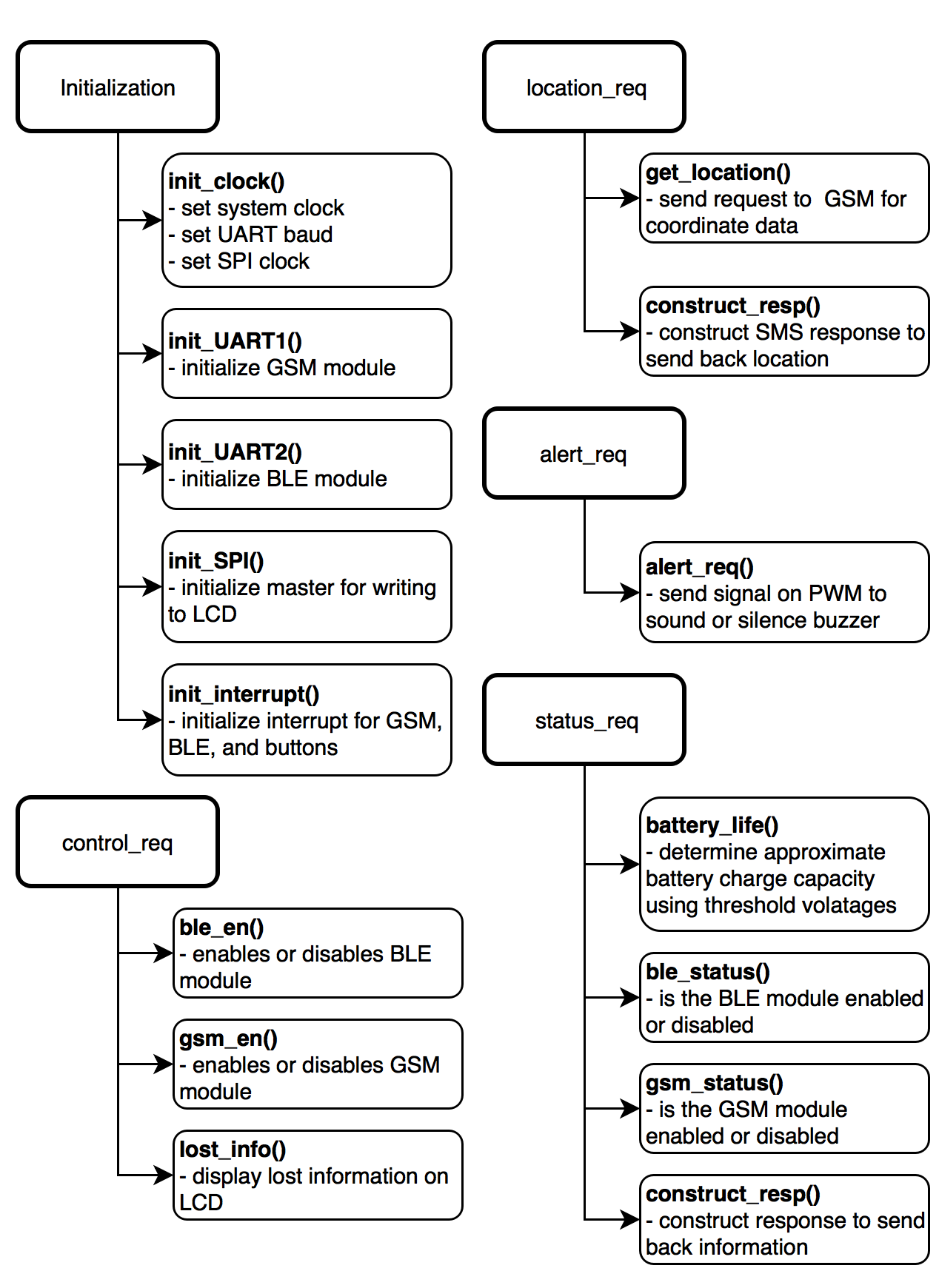
Available: <http://www.ti.com/lit/ug/slau627a/slau627a.pdf>. [Accessed: 21-Feb-2018].

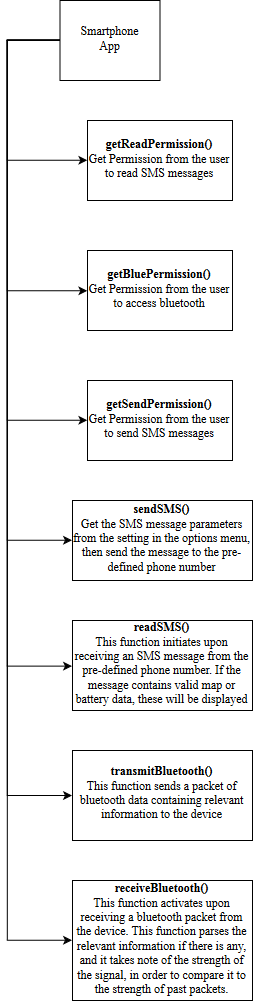
[3] “The Android Source Code,” *Android Source*, 14-Nov-2017. [Online]. Available:

<https://source.android.com/setup/>. [Accessed: 21-Feb-2018].

Appendix 1: Software Component Diagram



**

**