Tabletop Notifier

Tabletop device for displaying smartphone notifications

Alexander Cheung

Michael Federau

Krista Oribello

4/10/2015

An always-on tabletop display and smartphone notification interface. View the time and notifications from a paired Android smartphone at a glance. Play music wirelessly through the device.

Abstract

The Tabletop Notifier project is a tabletop display for surfacing notifications received wirelessly from a user's Android device. The Tabletop Notifier functions as a glanceable notification relay, clock, and wireless audio receiver. The device can be placed on a desk, counter or bedside table to provide quick and convenient access to messages and other notifications. The always-on display enables viewing of notifications as they arrive without requiring a user to operate their phone. Using the 3.5mm audio minijack, users can wirelessly play music through the connected speakers. An Android based smartphone is required to pair with the display using Bluetooth in order to receive its notifications.

The device is implemented on the Altera DE2 development board. An LCD display assembly and the development board pushbuttons serve as the primary output and input mechanisms for the device, with future provisions to incorporate the integrated resistive touchscreen on the display assembly. Interfacing with the LCD display was achieved using an LCD driver board manufactured from the LCD display provider for the Arduino platform. The software drivers for the LCD driver board were ported from the Arduino platform to the Altera DE2. Communication with Android devices was achieved using an off-board Bluetooth Module and the Bluetooth Serial Port Profile and Advanced Audio Distribution Profile. These Bluetooth profiles are integrated into the module as well as the Android Bluetooth Stack, enabling Bluetooth audio functionality with minimal configuration. The Bluetooth Hands Free Protocol is also available, allowing for wireless speakerphone capabilities in future implementations.

Contents

1 Functional Requirements

The Tabletop Notifier provides a range of functionalities, each a wireless extension of a connected Android device. Android devices running version 5.0 and higher with compatible Bluetooth radios are supported.

1.1 Real-time Notifications

As notifications arrive on the connected Android device, they are displayed in real-time on the Tabletop Notifier. The name of the application sending the notification, as well as the title and body of the notification, are displayed immediately upon arrival. These notifications are then stored for later viewing. Any notification from any Android application is supported.

1.2 View Missed Notifications

Any notifications that appear on the device can be recalled with the push of a button. Notifications and their contents will be displayed individually for the user to cycle through and view in the order they arrived.

1.3 Always-on Clock and Notification Count

The 800x480 resolution, always-on LCD display shows the current time in addition to the number of unread notifications. As time progresses, the displayed time updates. As notifications are received from the connected Android device, the number of unread notifications is updated and displayed below the time.

1.4 Audio Playback

Android devices can connect to and use the device as an external Bluetooth Audio device. This enables users to wireless play music through the speakers connected to the Tabletop Notifier via its 3.5mm audio minijack. Audio playback operates independently from serial data transmission so notifications will still be received while audio is being streamed.

2 Design and Description of Operation

The Bluetooth module receives data wirelessly from an Android phone via the Bluetooth Serial Port Profile. This information includes notification data from Android status bar notifications captured by the companion application running on the paired Android device. The Bluetooth module simulates a wired serial interface between the Android device and the Altera DE2, relaying the received information to the DE2 via pins on the general purpose input/output (GPIO) header connected to a universal asynchronous receiver/transmitter (UART) core on the FPGA. Software running on the DE2's uCOS II operating system receives and parses the notification data and displays it to the user via the LCD display.

Communication between the DE2 and the LCD display is accomplished using a dedicated LCD driver board, which interfaces with the DE2 using Serial Peripheral Interface (SPI). An SPI core on the FPGA is mapped to the GPIO header pins, which are connected physically to the LCD driver board. The LCD display assembly is connected and controlled entirely by the LCD driver board.

User interaction is implemented using a push button interface, requiring a button parallel input/output (PIO) component on the FPGA.

2.1 Data Flow Diagram

The diagram below illustrates the data flow between the various software components of the Tabletop Notifier.

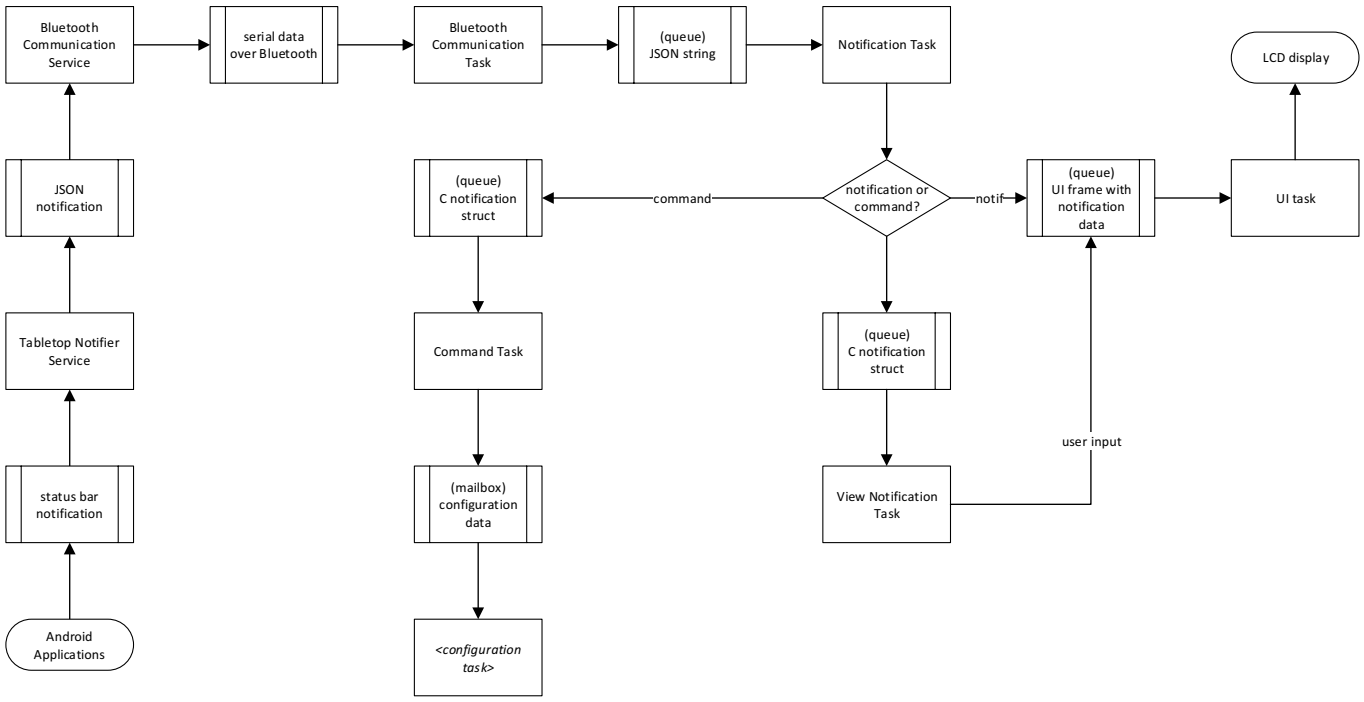


Figure 1: Data Flow Diagram

2.2 Hardware Block Diagram

The block diagram below illustrates the hardware components of the Tabletop Notifier. A requisite Android device is not shown.

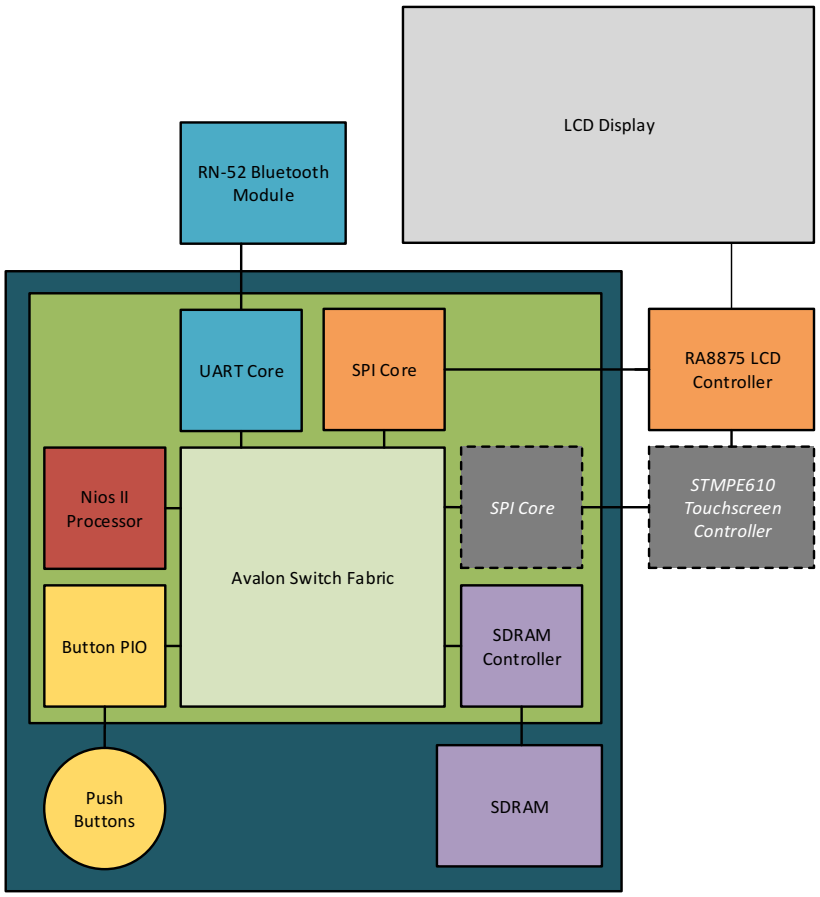


Figure 2: Hardware Block Diagram

3 Bill of Materials

Item	Part Name	Description	Supplier	Price
Altera Development Board	DE2	Development Platform: Product Page and Datasheet	U of A	$517.72 CAD
Bluetooth Module	RN-52	RN-52 Audio Bluetooth Breakout: Product Page and Datasheet	SparkFun	$39.95 USD
5.0" Touchscreen Display	KD50G21-40NT-A1	800x480 resolution TFT display with resistive touchscreen: Product Page and Datasheet	Adafruit	$39.95 USD
TFT LCD Driver Board	RA8875	Breakout board and driver chip for touchscreen display: Product Page and Datasheet	Adafruit	$34.95 USD
Speakers and 3.5mm audio jack		3.5mm audio output device	U of A	-

Total Cost: CAD $517.72 + USD $114.85. 5V 2A power supply required, but not included in total.

4 Reusable Design Units

4.1 Altera NIOS II "fast core" Processor

The Altera NIOS II "fast core" embedded processor will be used as the primary processor unit for adequate performance and responsiveness while being simple to implement. By using the NIOS II processor, the readily available Altera components for the DE2 can be used.

4.1.1 Specifications

The Nios II processor is a general-purpose RISC processor core. It supports the use of full 32 bit instruction set, data path and address space. It contains 32 general-purpose registers as well as 32 possible interrupt sources and possible external interrupts. The NIOS 2 has two closely related Software development tools: NIOS 2 SBT and NIOS 2 SBT for eclipse. Both tools flows are based on the GNU C/C++ compiler. The Nios II SBT for Eclipse provides a familiar and established environment for software development. Using the Nios II SBT for Eclipse, you can immediately begin developing and simulating Nios II software applications.

4.2 Altera SPI core

The SPI core used for interfacing between the DE2 and the RA8875 LCD controller board.

4.2.1 Specifications

The SPI core provided by Altera Qsys Software allows the DE2 board to communicate with a variety of off-board memory and control devices. This is accomplished by providing an Avalon Memory-Mapped interface on the back end and a SPI protocol on the front end. The core is configurable and will be implemented with a master protocol and data transfer width of 1 to 32 bits, this data width is to be determined after more data-rate testing can be done. The control of the core is done through the access to 5 registers and the Interrupts that can be generated by the core.

4.3 Altera UART core

The UART core used for interfacing between the DE2 and the RN-52 Bluetooth module.

4.3.1 Specifications

The Altera UART core is provided by the Altera Qsys Software and will be used to transfer data from the DE2 to the Bluetooth module. The Bluetooth module natively supports the UART communication protocol and uses it both when configuring the module and when sending data via the bluetooth connection. The core provides RS232 timing and allows for adjustments in baud rate, parity, stop and data bits. The core provides an Avalon Memory-Mapped slave interface that allows the NIOS 2 processor to communicate with the core simply by writing and reading the control and data registers. The user interface consists of six 16-bit registers, and an IRQ output that could request an interrupt when new data is received.

4.4 Adafruit RA8875 drivers

Manufacturer drivers for the Adafruit RA8875 LCD driver board was provided for the Arduino platform. These drivers, along with the Adafruit Graphics Library was ported to the Altera NIOS II/uCOS II environment to control the boards from the application software.

4.4.1 Specifications

- Source Size: 61.3 kB
- Compiled Size: 49 kB

4.5 microjson JSON parsing library

On the device side, notifications are received from the Android Companion Application in serialized JavaScript Object Notation (JSON) format. The microjson parser is a low-footprint JSON parser implemented in C for constrained embedded environments.

4.5.1 Specifications

- Source Size: 49 kB
- Compiled Size: 25 kB

5 Datasheet

5.1 Internal Connections

5.1.1 RN52 Bluetooth Module Connections

- 3.3V power source (current requirement varies on Bluetooth profiles, typically ~30mA)
    - connected to GPIO header 0 pin 28 (3.3V source)
- PWR_EN, tied via wire wrapping to the 3.3V source
- GND
    - connected to GPIO header 0 pin 29 (Ground)
- GPIO9 (when driven low, tells module to interpret UART input as configuration commands)
    - connected to GPIO header 0 pin 30
- UART_TX (115200 baud)
    - connected to GPIO header 0 pin 32
- UART_RX (115200 baud)
    - connected to GPIO header 0 pin 34
- SPK_L+ (-40 to -60 dBV)
    - connected to audio jack left
- SPK_R+ (-40 to -60 dBV)

    - connected to audio jack right
- SPK_L- and SPK_R-
    - tied together and connected to audio jack ground
    - also tied to ground through 10ohm resistors to reduce static noise

5.1.2 RA8875 Adafruit LCD Controller Connections

- VIN power source (current requirement varies depending on modes and screen brightness, typically ~20-50mA)
    - connected to GPIO header 1 pin 28 (3.3V source)
- GND
    - connected to GPIO header 1 pin 29 (Ground)
- SCK (SPI Clock line, Maximum Clock rate of 1Mhz, clock polarity: 0, clock timing: 0)
    - connected to GPIO header 1 pin 30
- MISO (SPI MISO line)
    - connected to GPIO header 1 pin 34
- MOSI (SPI MOSI line)
    - connected to GPIO header 1 pin 32
- CS (SPI Slave Select line)
    - connected to GPIO header 1 pin 31
- RST (Driven Low then High to reset device at startup)
    - connected to GPIO header 1 pin 36

5.2 Onboard Connections

- 2 pushbuttons (Key 0 reset, Key 1 Notification Interaction)

5.3 External Connections

- 3.5 mm audio out
    - connection to any 3.5mm audio device
- DE2 Power supply
    - input 9V, 1.3A standard power connection
- Bluetooth 3.0 class 2
    - 2.4~2.48Ghz, Sensitivity -85 dBm at 0.1 % BER, TX power 4dBm
    - Operation Range (~10 meters)
    - Max data rate of 3 Mbps

6 Background Reading

6.1 Android Bluetooth API Guide

The Android Bluetooth API Guide provides all of the necessary information for developers to utilize Bluetooth functionality on Android devices. The guide describes the relevant classes provided by the Android Application Framework and how to use them to leverage an Android device's Bluetooth network stack. See References (8): Android Open Source Project.

6.2 Bluetooth Standard

The Bluetooth standard is well documented and defined. Both the Core specifications and the individual profile specifications are outlined in the many documents listed on the webpage. See References (14): Bluetooth Specifications.

These specifications were important when deciding which Bluetooth module would be used and which Bluetooth profiles we wished to support.

6.3 Arduino SPI Library

The Arduino SPI Library was integral to the porting of Adafruit LCD libraries into a form that could be used with the DE2. Arduino provides a breakdown of what the different SPI functions accomplish. With this information, we were able to discern which functions were necessary for the LCD driver board and display to communicate effectively with the DE2.

7 Software Design

7.1 uCOS II

The operating system controlling the Tabletop Notifier hardware is uCOS II. Seven main tasks will be written to accomplish the basic functional requirements. Each of the tasks and their relationships is shown in the process interaction diagram below:

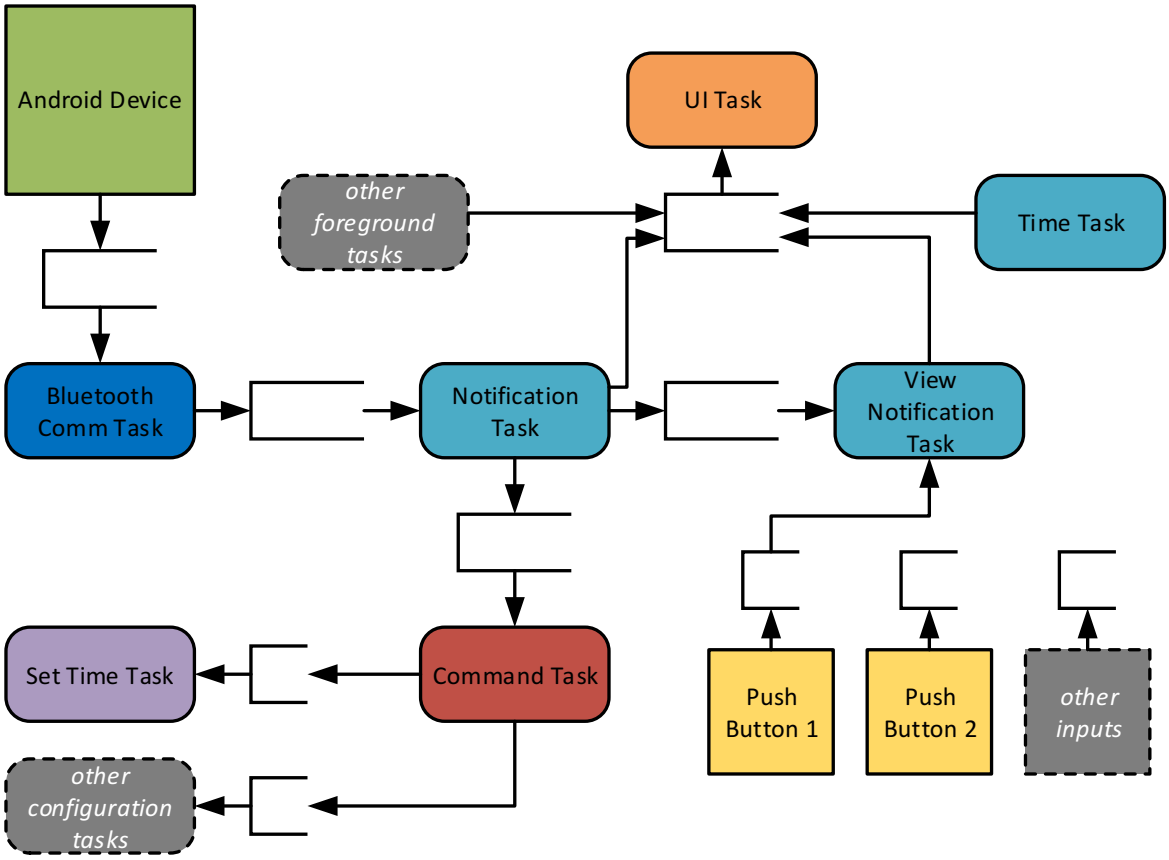


Figure 2: uCOS process interaction diagram

7.1.1 Bluetooth Communication Task

The Bluetooth communication task is responsible for reading serial data received by the Bluetooth Module. The task reads the input stream and posts each line of data to the JSON notification queue. The Android companion application is written such that each notification transmitted to the Tabletop Notifier will be followed by a carriage return and new line character. As such, each message in the JSON notification queue will be a complete notification in serialized JSON. When there is no data available to read, this task will block until data becomes available.

7.1.2 Notification Task

The notification task is responsible for converting Android notifications from serialized JSON format into a C data structure that can be more readily processed by the application. To accomplish the translation from JSON to C structure, the microjson library will be used to parse the serialized JSON notifications posted to the JSON notification queue by the Bluetooth communication task. The notification task uses the library to parse the JSON into predefined attributes of a notification C structure, which is then posted to the notification queue. The messages in this notification queue contain pointers to the translated notifications in shared memory.

7.1.3 UI Task

The UI task is responsible for performing updates to the user interface. User interface updates are contained in the form of structures that define both the data to display and the rendering procedure used to display the data. These updates are produced by other system tasks and posted to the UI frame queue. The UI task pends on this queue for updates and renders them upon arrival. The UI task has no knowledge of the data or the rendering procedure. This is by design, and the intention is to write output device-specific renderer functions to support multiple output mechanisms. With this logical separation, each task is able to dictate how its data should be displayed to the user without being directly responsible for its rendering. Additionally, this design enables support for multiple displays, as renderer functions can be written for any number of output devices. Renderers for each output device can tailor the appearance of the same data into a form appropriate for the device without any need to modify the tasks sending the data.

7.1.3.1 UI Task Contention

In order to prevent UI frames created by different tasks from interfering with each other, a mutex will be used guard the UI frame queue from simultaneous updates. The renderer functions invoked by the UI task will lock the mutex, preventing other applications from interrupting the rendering process by queueing up frames in the middle of rendering another set of frames from a different task. Tasks are required to lock the mutex before posting frames to the UI queue, and unlock the mutex once those frames are ready to be rendered.

Some renderers, such as those for rendering data from the time task, will not hold the lock for long periods of time, while others may hold the lock for indefinite time periods while waiting for user input. This behavior is specific to the data of each task, and the tasks will be designed to keep the risk of contention minimal unless user input is required.

7.1.4 Time Task

The time task is responsible for updating the current time and unread notification count, posting these updates to the UI frame queue. The task is configured to update these fields every second. The unread notification count is obtained by querying for the number of messages in the notification queue.

7.1.5 View Notification Task

The View Notification Task listens for inputs from push button #1 by pending on the push button's mailbox. When the user presses the push button, an interrupt is triggered and a message is inserted into the button's mailbox and consumed by the pending task. When the View Notification Task consumes a message from this mailbox, it sequentially posts each notification from the queue of unread notifications to the UI task to be rendered to the user in the order of arrival.

7.1.6 Command Task

The Command Task receives notifications from the Notification Task when the sending application on Android is the companion application itself. This type of notification contains configuration and settings data for background tasks (tasks which do not post frames to the UI task). The Command Task will decide based on the notification data what action needs to be taken and by which background task. The data for the action is then posted to the mailbox for the appropriate background task.

7.1.7 Set Time Task

The Set Time Task is a background task that is responsible for modifying the system time based on configuration commands from the companion Android application.

7.2 Android

The Android companion application runs on the Android Application Framework, utilizing the framework's provided Bluetooth, JSON, and Notification APIs within the defined Activity and Services App Components. The Android services (background processes) and their interactions are illustrated in the process interaction diagram below:

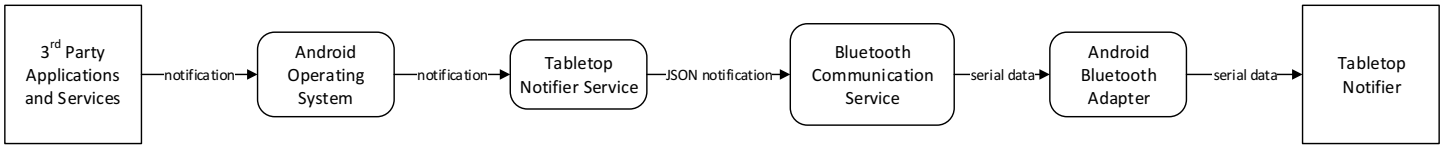


Figure 3: Android process interaction diagram

The application is designed primarily as a set of interacting Services. The Tabletop Notifier Service is responsible for listening to the Android operating system for status bar notifications generated by system or third-party applications. Once received, the service will extract the relevant data segments from the notification, format them in serializable JavaScript Object Notation (JSON), and pass them to the Bluetooth Communication Service for transmission to the Tabletop Notifier.

7.3 LCD and Touchscreen Driver

The LCD controller board - RA8875 - was used to connect the DE2 to the TFT LCD display. Adafruit provided a graphics library for the driver board to handle basic interfacing, drawing, writing text, and reading functionalities. These files, written for use with Arduino microcontrollers, have been ported for use with Nios II UCOS on the DE2.

The two Adafruit libraries provided were for Adafruit_RA8875 and Adafruit_GFX classes. As classes were unable to work and there was a great deal of function overlap between both libraries, Adafruit_LCD files were written to handle the LCD driver. These files include the functions for constructing the LCD struct and initializing it to work with the DE2. Graphics and text functions provided by Adafruit were changed minimally as functions all called the same low-level functions to read and write bytes of data to the RA8875. Low-level functions were mostly rewritten for SPI control with the DE2.

The libraries provide graphics functions for drawing shapes and filling the screen. Text functionality is also included although the RA8875 was limited to a single font and few sizes. Although the RA8875 was capable of improved graphics and text, our project display was written more simply because of time and hardware constraints.

8 Test Plan

8.1 Software

8.1.1 Android

Testing of the Android application was performed manually by connecting to a Bluetooth compatible laptop computer and observing the messages sent across the connection by the background services. Messages sent across the Bluetooth serial connection were verified using the screen terminal emulator connected to the Bluetooth device. Using this method enabled testing of the majority of the application's functionality, including receipt of notifications from the Android system by the application, establishment of Bluetooth SPP connections by the application's connection service, and transmission of serial data across the connection.

8.1.2 uCOS II

Written µCOS tasks were initially tested on the DE2 using console output and/or the 16x2 character LCD display until the RA8875 LCD display driver was ported to the platform. Notifications were simulated using an additional button to trigger notification generation in absence of a connected device. Once the Android app was completed and the Bluetooth connection could be established, the app was used to send test SMS messages and emails. This tested whether the handling and count of incoming notifications were properly handled by each uCOS task. Once the porting of the RA8875 driver board libraries was completed, testing began using the primary LCD display.

8.1.3 RA8875 LCD

To test the LCD display, we first needed to port the RA8875 driver board libraries to Altera NIOS II/uCOS II environment. Some hardware and software adjustments needed to be made in order to make the LCD display properly run through different commands of our LCD test code. The testing was able to point out what adjustments needed to be made on the low-level functions of the driver (SPI pin communication). The method in which the original Arduino code transferred data (through SPI) needed to be reworked into a way that was usable with the DE2. With the LCD powered up and working, we tested the LCD's capabilities by using the higher-level graphics and text functions.

8.2 Hardware

8.2.1 RN-52 Bluetooth Module

The RN-52 Bluetooth module was tested in several steps. Firstly, a test Altera UART core was created and connected to the default RS232 port on the DE2. This was then connected via serial cable to the computer to ensure that the UART core functioned as anticipated. This test core was then altered to connect its transmit and receive lines to 2 of the many GPIO pins. At this point a small sample program was written to send out a repeating batch of data on the transmit pin, which was then monitored using the Tektronix TDS 1002 oscilloscope. After the pins were determined to be operating correctly the Bluetooth module was finally connected. At this point the Bluetooth module was tested by sending several commands outlined in the RN-52 Command Reference User’s Guide, this causes the RN-52 to send a response back to the DE2 which we can then read.

8.2.2 LCD and Touch Screen SPI Cores

The LCD controller board uses the SPI serial communication protocol as the primary means of communication with other devices. Altera SPI cores are added to the FPGA and their respective outputs are mapped to several pins on the GPIO Headers. To begin testing, small test programs were written that output simple repeating, easily discernable bit patterns that were then verified using the oscilloscope. Once pin outputs were verified to be operating correctly (especially in terms of maximum voltages and currents), the pins were then connected to the controller board.

9 Results of Experiments and Characterization

While integrating the various components to operate together, it was required that both the DE2 and the component were operating as outlined in the datasheets and specifications. It would be catastrophic to have mismatched voltages or currents being sent through channels not designed for them. To minimize the risk to the components and DE2, all inputs and outputs were tested through the use of the digital oscilloscope. The steps involved in this required a known output to be sent to the DE2's GPIO pins and monitored. Once connected and sent to the component, the component was monitored to ensure that the correct output sequence was created. Then the output of the component could be connected back into the DE2. This process required more time than perhaps necessary, but ensured that no accidental incorrect connections occurred. This was necessary because the time constraints of the project would not easily allow for replacement parts to be obtained or integrated.

While deciding on the amount and types of hardware required to implement the Tabletop Notifier to provide the desired functionality. A decision on the type of connection to various components was required. This is of particular interest on the side of the RN-52 Bluetooth module. The module allows for many different ways of connecting it to the DE2 (UART, SPI, USB and PCM for audio). However, only one of these connection formats (UART) can be used to configure the device. Because the module is only configured at startup and while being configured cannot be reliably used to communicate with any Bluetooth device, we decided to combine the data streams and sending of configuration commands onto the one UART connection. This required us to use a GPIO pin to tell the Bluetooth module when to interpret the incoming data as command or data to be sent. However it reduced the number of connection cores required on the FPGA and overall reduced the total logic elements required.

In our IO Demonstration, it was pointed out that the font of text used on the LCD appeared small and primitive. This was due to limitations in using a certain function to write text to the screen (textWrite() given by Adafruit_RA8875 library). The text on-screen was improved by using a function taken from Adafruit_GFX libraries. This method reads data from glcdfont.c (provided by Adafruit) which draws out characters one pixel at a time. Using this function to write to the display gave bigger font that was easier to see on the large display. However, this function was never used in our final project because the rendering of each frame to the screen took noticeably longer. We agreed that the speed of the rendering on the screen was more important to the overall function of the Tabletop Notifier rather than the font of the text.

10 Safety

10.1 Soldering Risks

Minor soldering, when required, will be conducted using adequate eye protection and proper soldering techniques.

10.2 Electrical Risks

Maximum Voltage: 9V, Maximum Current: 1.3 A (the power source for the DE2)

Sensitive circuits will be handled with the appropriate ESD safety precautions, including grounding straps, ESD mats, and care when handling of circuits.

11 Environmental Impact

Three of the components of the Tabletop Notifier are RoHS compliant: RN52 Bluetooth audio module, and the RA8875 TFT LCD driver board. The DE2 board and LCD touchscreen are not RoHS compliant. If the LCD screen should crack, safety measures as outlined in the LCD datasheet would need to be performed and proper disposal techniques should be used.

12 Sustainability

The active and sleep modes of the components can be seen below.

Active:

Written µCOS tasks will be initially tested on the DE2 using console output and/or the 16x2 character LCD display until the RA8875 LCD display driver is ported to the platform. Notifications can be simulated using an additional button to trigger notification generation in absence of a connected device. This will test whether the handling and count of

incoming notifications are properly handled by each uCOS task. Once the porting of the RA8875 driver board libraries are completed, testing can begin using the primary LCD display. Calculated active power in between 0.3732 - 0.5112 W

Sleep:

Component	Typical Power
RA8875 LCD Driver	1.056 mW
RN52 Bluetooth Module	< 1.65 mW

As the Tabletop Notifier is intended for all-day use. The power consumption has been calculated with active power.

City of Edmonton - Residential cost of energy = $ 0.05431/kwH

Yearly Power Consumption = 3.26923 - 4.47811 kWh/year

13 References

[1] Adafruit. (2015, Jan. 19). RA8875 Driver Board for 40-pin TFT Touch Displays - 800x480 Max. Available: http://www.adafruit.com/products/1590

[2] Adafruit. (2015, Jan. 19). Resistive Touch Screen Controller - STMPE610. Available: http://www.adafruit.com/products/1571

[3] Adafruit. (2015, Jan. 19). 5.0" 40-pin TFT Display - 800x480 with Touchscreen. Available: http://www.adafruit.com/products/1596

[4] Roving Networks. (2015, Jan. 19). RN52 Bluetooth Audio Module DataSheet. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/rn-52-ds-1.1r.pdf

[5] Roving Networks. (2015, Jan. 19). RN52 Bluetooth Audio Module Command Reference User's Guide. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/50002154A.pdf

[6] Altera. (2015, Jan. 19). Altera DE2 Development and Education Board User Manual. Available FTP: ftp://ftp.altera.com/up/pub/Webdocs/DE2_UserManual.pdf

[7] Altera. (2015, Jan. 29). Embedded Peripherals IP User Guide. Available: http://www.altera.com/literature/ug/ug_embedded_ip.pdf

[8] Android Open Source Project. (2015, Jan. 29). Bluetooth API Guide. Available: http://developer.android.com/guide/topics/connectivity/bluetooth.html

[9] Altera. (2015, Feb. 2). Embedded IP Peripherals Guide. Available: https://eclass.srv.ualberta.ca/pluginfile.php/1773677/mod_resource/content/2/ug_embedded_ip.pdf

[10] Altera. (2015, Feb. 2). 16x2 Character Display. Available: https://eclass.srv.ualberta.ca/pluginfile.php/1773679/mod_resource/content/2/Character_LCD.pdf

[11] Altera. (2015, Feb. 2). Avalon Tri-State Conduit Components Guide. Available: https://eclass.srv.ualberta.ca/pluginfile.php/1773681/mod_resource/content/1/ug_avalon_tc.pdf

[11] Altera. (2015, Feb. 2). NIOS 2 Documentation. Available: https://eclass.srv.ualberta.ca/mod/resource/view.php?id=1227736

[12] Adafruit. (2015, Feb. 2). Adafruit RA8875 Driver. Github: https://github.com/adafruit/Adafruit_RA8875

[13] Adafruit. (2015, Feb. 2). Adafruit GFX Library. Github: https://github.com/adafruit/Adafruit-GFX-Library

[14] Bluetooth. (2015, April 10). Bluetooth Specifications. : https://www.bluetooth.org/en-us/specification/adopted-specifications

14 Appendix A – Quick Start Manual

14.1 Android Companion Application

This section explains how to install the Tabletop Notifier Android companion application.

14.1.1 Prerequisites

You will need an Android device running Android 5.0 or above that is compatible with Bluetooth 3.0.

14.1.2 Procedure

1. On the Android device, download the Android Application Package (tabletop-notifier.apk) from https://bitbucket.org/ece492w15/tabletop-notifier-android-companion/downloads/tabletop-notifier.apk

2. Open the .apk on the Android device to install the application.

14.2 Altera DE2 Preparation

14.2.1 Required Materials

- Altera DE2 Development Board
- DE2 Power Cable
- USB Connection Cable (for programming only)
- Tabletop Notifier Prototype board
- 2x 40-pin Ribbon Cables
- 3.5mm audio in speakers
- TabletopNotifier.pof
- TabletopNotifier.sof
- The required Software files

14.2.2 Connections

To connect the Tabletop Notifier to the DE2, use the 2 Ribbon Cables to connect the DE2 GPIO header 0 to the header on the prototype board labeled 0, and the DE2 header 1 to the header labeled 1. Then connect the DE2's power, USB and the Prototype board 3.5 mm headphone jack to the speakers.

14.2.3 Setup Procedure

Begin by programming the DE2 with the provided "TableTopNotifier.pof" file, then after powering down the board and switching the operation switch to "run" mode, program the board again. This time using the "TableTopNotifier.sof" file. Now open the Eclipse tools for Nios2 and create a new project and BSP from template, choose the "hello MicroC/OSII" template. Name the project "TableTopNotifier". Delete the generated "hello_ucosii.c" file and copy the supplied Software folders and files into the project. Clean and Build Project. Then using the same steps outlined in the third tutorial flash the code onto the DE2.

14.2.4 Initial State

Once the above steps are done, every time the board powers on, it will do so into its initial state. In this state the time will be set to 00:00 and the RN52 Bluetooth module will have two alternating blinking lights to signify that the Bluetooth module is unpaired and awaiting connection.

14.2.5 Connecting to the Bluetooth Module

The RN52 Bluetooth module operates in two ways for this project.

The first is the audio out mode. To activate this feature of the Tabletop Notifier: activate the Bluetooth connection on your phone and through your phones operating system pair to the device "RN52-7E17". This will allow the Tabletop Notifier to function as a wireless Bluetooth speaker for any application on your phone.

The second operation requires you to have the previously discussed Android application installed on your phone. Through this application the second Bluetooth connection to the "RN52-7E17" is made. This connection is purely data and is used to send notifications and configurations to the Tabletop Notifier.

14.2.6 Sending Notifications and General Operation

The provided Android application allows several pre-defined messages to be sent to the Tabletop Notifier.

15 Appendix B – Future Work

15.1 Interface Improvements

Two major areas for future work on the user interface are incorporating the touchscreen for user input and displaying a rich graphical user interface.

The LCD display assembly includes an integrated resistive touchscreen. The signals from the touchscreen are sent along with the display data to the RA8875 LCD driver board. These signals are passed back out through 4 pins X+/- and Y+/-. These signals can be connected to the STMPE610 touchscreen controller board. Similar to the RA8875, the STMPE610 drivers will need to be ported to the Altera DE2 platform.

The display drivers for the LCD controller board are only suitable for drawing text and simple shapes and lines. A lightweight GUI library would ideally be used instead to render windows, buttons, and other canonical user interface elements.

15.2 Hands-free

Hands-free speakerphone functionality is a logical extension for the Tabletop Notifier. With the addition of a microphone and echo cancellation software/hardware, the Bluetooth Hands-Free Profile (HFP) can be used to enable this feature. The RN-52 Bluetooth Module and any modern Android Bluetooth-enabled device includes native support for the HFP, requiring no additional configuration or application changes to enable.

15.3 Notification Interaction

In its current state, the Tabletop Notifier is a unidirectional relay of information. Modern Android notifications are complex data structures that often have many options for interactivity. Some of these interactions are simple, such as dismissing the notification on the device itself, while others are more complicated and specific to the application that sent the notification.

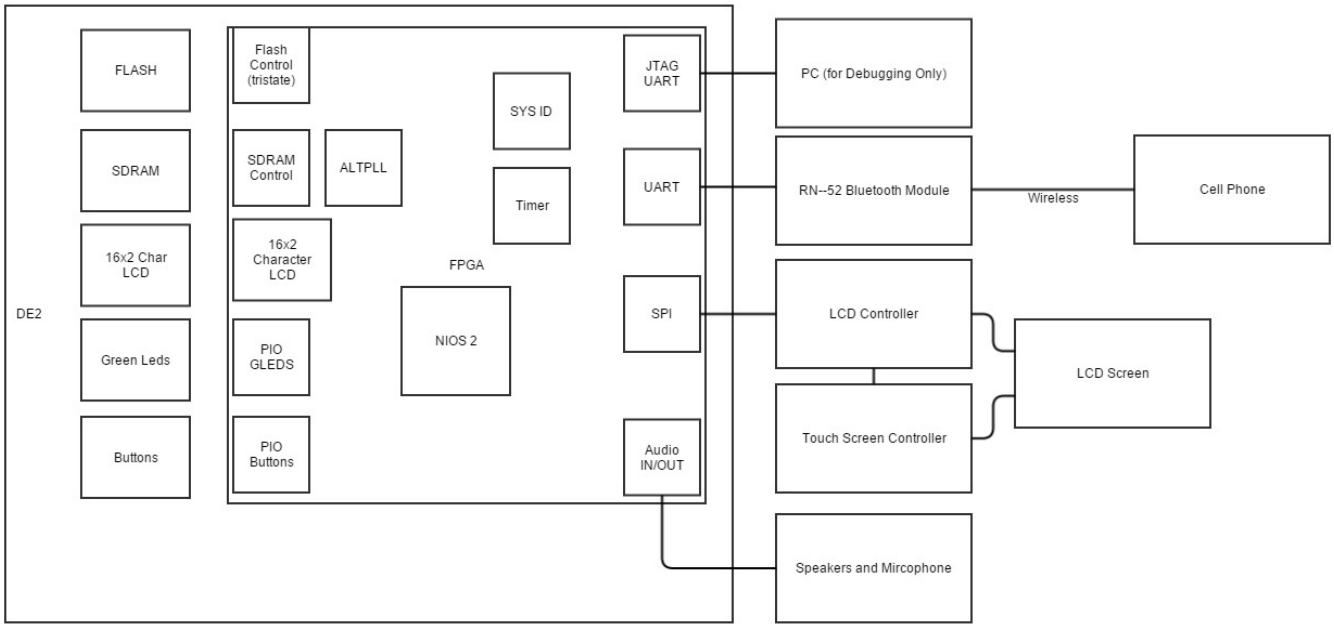
For example: notifications from Google's Gmail application have hooks that allow for archiving emails directly from the notification, notifications from music applications often have hooks for controlling playback.

One method of invoking these types of notification actions is by using the Android Wear Notifications API. In theory, using the Android Wear API is not restricted to Android Wear devices due to the open nature of the Android ecosystem. The Pebble Smartwatch is a great example of a third-party hardware platform that leverages the Android Wear API.

15.4 Application Platform

Another area for future work with enormous potential is turning the Tabletop Notifier, or a mature descendant of it, into a platform rather than a single device. This would require designing a framework for third party developers to access exposed functionality of the Tabletop Notifier in order to design their own applications for the device. Again, the Pebble Smartwatch is a good example. The Pebble operating system is based on FreeRTOS and utilizes a form of middleware to manage relocatable applications.

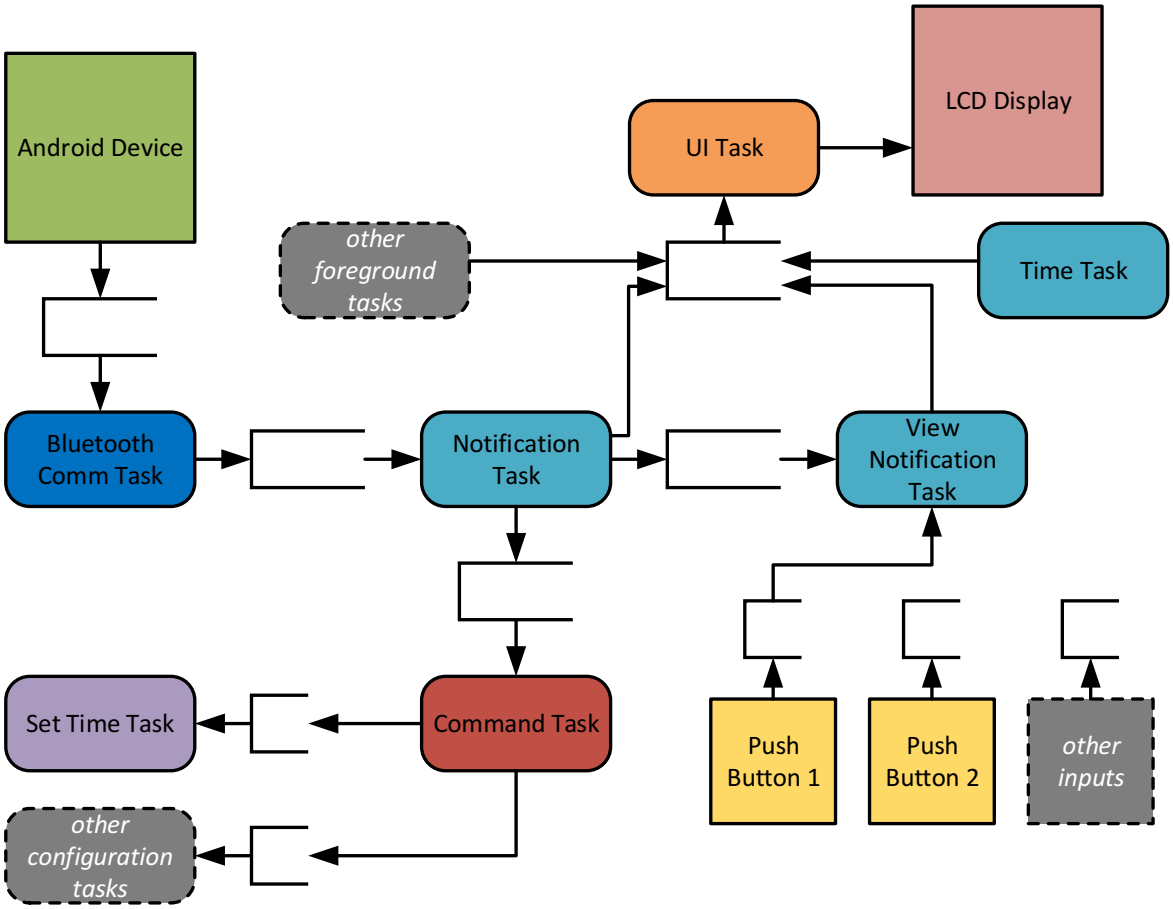
16 Appendix C – Hardware Documentation



17 Appendix D – Source Code

17.1 uCOS

17.1.1 Process Interaction Diagram



17.1.2 Source Code Index

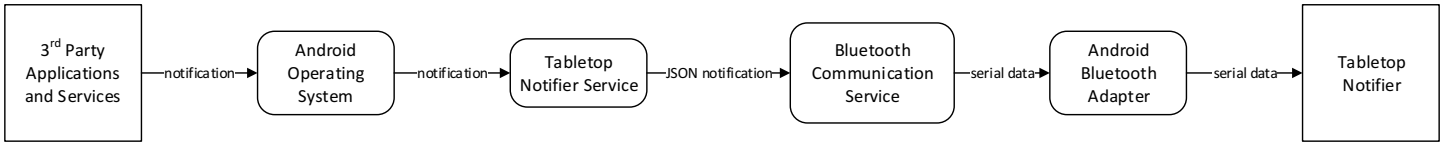
File	Description	Status
mjson/json.c	microjson library file	T
mjson/mjson.h	microjson library file	T
mjson/mjson.c	microjson library file	T
tasks/action_task.h	action task header	T
tasks/action_task.c	action task implementation	T
tasks/bt_comm_task.h	bluetooth communication task header	T
tasks/bt_comm_task.c	bluetooth communication task implementation	T
tasks/notification_task.h	notification task header	T
tasks/notification_task.c	notification task implementation	T
tasks/time_task.h	time task header	T
tasks/time_task.c	time task implementation	T
tasks/ui_task.h	UI task header	T
tasks/ui_task.c	UI task implementation	T
utils/error_handlers.h	error handler header	T
utils/error_handlers.c	error handler implementation	T
main.c	main program implementation	T
android/notifications.h	notifications header	T

android/notifications.c	notifications implementation	T
renderers/renderers.h	renderers header	T
renderers/renderers.c	renderers implementation	T
char_display_renderer/char_display_renderers.h	character display renderer header	T
char_display_renderer/char_display_renderers.c	character display renderer implementation	T
lcd_renderer/lcd_renderers.h	lcd display renderer header	T
lcd_renderer/lcd_renderers.c	lcd display renderer implementation	T
log_renderer/log_renderers.h	log renderer header	T
log_renderer/log_renderers.c	log renderer implementation	T
lcd/Adafruit_LCD.h	adafruit lcd library file	T
lcd/Adafruit_LCD.c	adafruit lcd library file	T
lcd/glcdfont.c	lcd font-drawing file	T
buttons/buttons.h	push-button interrupt header	T
buttons/buttons.c	push-button interrupt implementation	T

The source code is available on Bitbucket here: https://bitbucket.org/ece492w15/tabletop-notifier-ucos-application/src/

17.2 Android

17.2.1 Process Interaction Diagram



17.2.2 Source Code Index

File	Description	Status
Package ca.alexandercheung.tabletopnotifier		
MainActivity.java	Main Android Activity for connection and configuration interface	T
DeviceListActivity.java	Bluetooth device selection	T
TabletopNotifierFragment.java	Fragment containing Bluetooth controls	T
TabletopNotifierService.java	Background Android Service responsible for listening to system notifications and passing them to the `BluetoothCommService` for transmission	T
BluetoothCommService.java	Android Service responsible for setting up and managing Bluetooth connections in addition to performing data transmissions when connected	T
Constants.java	Interface containing message types sent between the application's Services	T

The source code is available on Bitbucket here: https://bitbucket.org/ece492w15/tabletop-notifier-android-companion/src/

The compiled .apk can be found in the Downloads section of the same repository here: https://bitbucket.org/ece492w15/tabletop-notifier-android-companion/downloads/tabletop-notifier.apk

17.3 Quartus Project

The Quartus archive of the project, including the source code for the top level VHDL design, is available in the Downloads section of the uCOS software repository here: https://bitbucket.org/ece492w15/tabletop-notifier-ucos-application/downloads/TableTopNotifier.qar