Tabletop Notifier

Group 6
Critical Design Review
Presentation

Motivation

- Tabletop display for wirelessly presenting and interacting with smartphone notifications
- Placed on a desk, countertop, or bedside for viewing messages and other notifications without need to operate phone

Basic Features

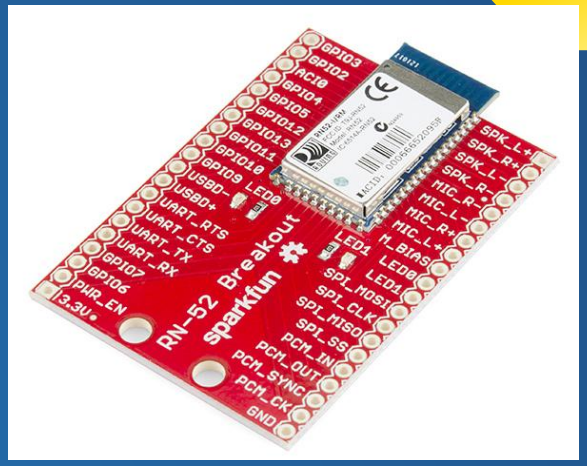
- Always-on clock to display the current time
- Receive text message, email, or other notifications wirelessly from paired smartphone using Bluetooth
- Display the text message, email, or other notification received from the paired smartphone
- Companion Android application/service for configuration and communication.

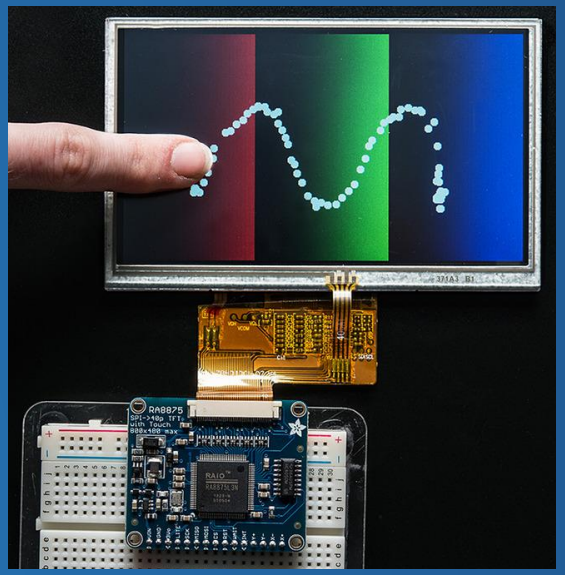
Stretch Features

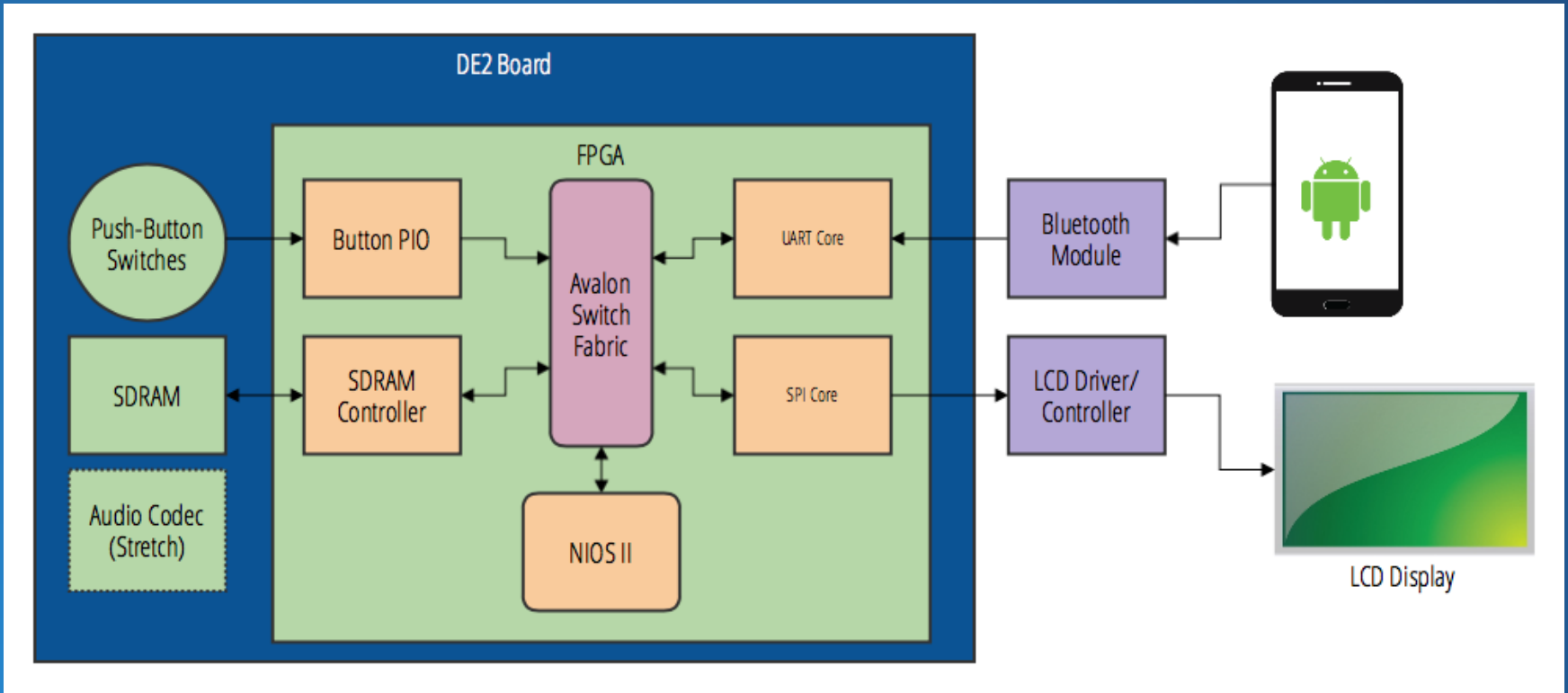
- Bluetooth audio output (music)
- Bluetooth audio input (speakerphone)
- Rich interaction with notifications
  - physical buttons, or
  - touch enabled display
- Alarms
- other “apps”?

Design | Hardware

- Bluetooth Module
  - UART
  - Audio interface
- LCD Controller
  - SPI
- Touchscreen Controller
  - SPI







Hardware Block Diagram

Design | Software

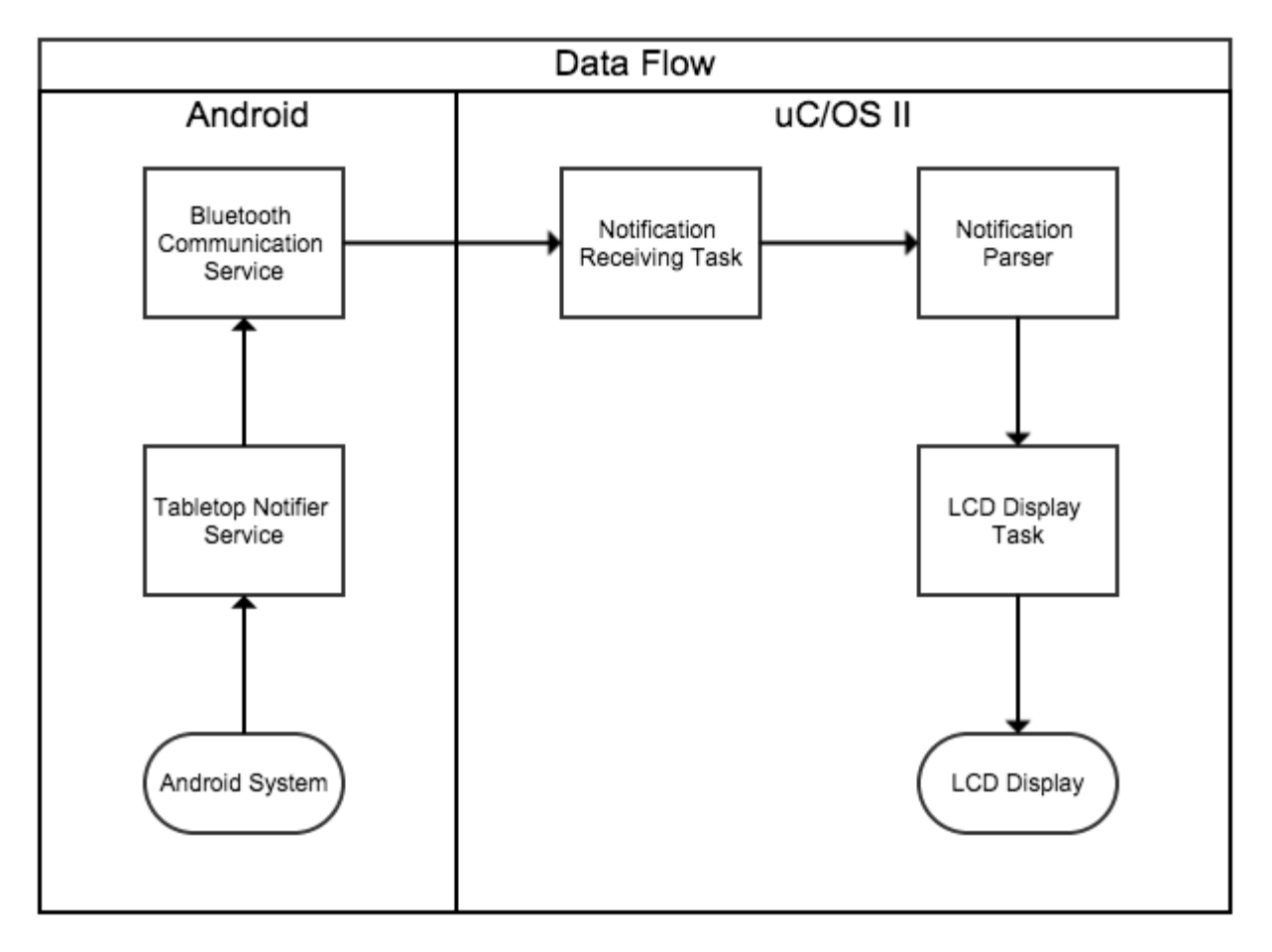
Android

- Connection and configuration interface
  - Activities, Fragments
- Bluetooth communication and notification capture
  - Services
  - Android Bluetooth API
  - JSON

uCOS II

- Tasks
  - receive notification data from Bluetooth
  - parse notifications
  - write to LCD
  - system apps (clock, etc.)
- Port of LCD drivers from Arduino

Dataflow Diagram



Code Sample (1)

```c
/* Data object to model */
struct notification {
    char title[SIZE];
    char text[SIZE];
};

/* Object specific parsing function */
static int json_notif_read(const char *buf, struct notification *notif) {

    /* Mapping of JSON attributes to C object's struct members */
    const struct json_attr_t json_attrs[] = {
        {"title", t_string, .addr.string = notif->title, .len = sizeof(notif->title)},
        {"text", t_string, .addr.string = notif->text, .len = sizeof(notif->text)},
        {NULL},
    };

    /* Parse the JSON object from buffer */
    return json_read_object(buf, json_attrs, NULL);
}
```

Code Sample (2)

```c
int main(int argc, char *argv[])
{
    /* Allocate space for object */
    struct notification *my_notif = malloc(sizeof(struct notification));

    /* Call object parsing function */
    int status = json_notif_read(argv[1], my_notif);

    if (status == 0) {
        printf("Title: %s\n", my_notif->title);
        printf("Text: %s\n", my_notif->text);
    } else {
        puts(json_error_string(status));
    }

    return status;
}
```

```
./microjson_demo '{"title":"my test title","text":"my test text"}'
Title: my test title
Text: my test text
```

App Notes

Possible Topics

- **Android**
  - accessing notifications from the operating system
  - Bluetooth API and Serial Port Profile
  - Android Test Framework
- **C**
  - port of LCD drivers
  - port of touchscreen drivers
  - parsing JSON on uCOS II with microjson library

Test Plan | Software

Android

- Unit tests written in Android Test Framework (JUnit)
- Communication over Bluetooth SPP tested by connecting to PC terminal

µCOS

- Written µCOS tasks will be first tested on the DE2 using the on-board 16X2 character display
- Until Bluetooth capabilities have been established, on-board push-buttons can be used to simulate incoming notifications
- So far...

Test Plan | Hardware

Altera De2

- v1.0 - Basics (16x2, LEDS, Buttons)
- v2.0 - Includes UART Altera core
- v3.0 - Includes SPI Altera cores (LCD, touchscreen)
- v4.0 - transfer to Non-Volatile memory

Backup Plans

- **Character LCD**
  - If we can’t get the drivers working for the LCD display controller
- **Push Button Interface**
  - If we can’t get the drivers working for the Resistive Touchscreen Controller