Spring 16

Android Controlled LED Table

Eugene Balaguer, Greg DePaul, Jay Jackman

Executive Summary: The goal of this project is to design an implement an LED grid on top of your everyday coffee table. This grid is reconfigurable through a Bluetooth connected android device. The application is extendable to a multitude of different uses, for example a drawing canvas, a displayable game, and the convenience of an electronic clock.

Table of Contents

Table of Figures

Table of Tables

Project Overview

The goal of this project is to retrofit a table with an LED grid, extending the functionality of your typical wooden table into a technical marvel. The grid of LED's will be reconfigurable using an android smartphone, providing a miniature canvas, games, conveniences and more to what would otherwise be a flat surface. We make use of a Raspberry Pi as well as its ability to interface with Bluetooth in order to provide the user with the ease and simplicity of having the android app on hand to do with the table as they will. This application will specifically take a grid drawn image from an android device and then reformat and display said image onto the LED grid. Proximity sensors would allow users to interact with the table, which can then be transmitted back to the user's smart phone.

Requirements and Testing Strategy

For the ease of reading, we have created prose of the requirements and testing strategies for each requirement. The requirements are in black and the testing strategies in red:

1) User can choose different modes of use via a menu:
    a) All game modes will be confirmed to appear on the menu
    b) User can choose Ball Balance mode
        i) Ball should move based on the tilt of the phone via the accelerometer inputs
            (1) User will place phone on known inclines to confirm that the ball rolls as planned
    c) User can choose Drawing mode
        i) User should be able to drag their finger across the screen and also tap the screen to illuminate grid tiles
            (1) User will touch and swipe their finger on the phone grid to confirm correct drawing behavior
    d) User can choose Clock mode
        i) Grid should display a functioning digital clock
            (1) The displayed time will be monitored alongside a known working clock to confirm that they match
    e) User can choose FFT Music mode
        i) Grid should display a FFT music display using the phone microphone
            (1) Grid display will be compared to a Matlab output of the same wav file to confirm accuracy
    f) User can choose picture mode
        i) Grid should display a pixilated version of an image by dividing the picture into segments and taking the average color for each segment
            (1) Grid output will be compared to the result of a program that is known to utilize this method of pixilation
    g) User can choose checkers mode
        i) User should be able to play a game of checkers on the grid
            (1) User will play a game of checkers to confirm that the rules are working properly
2) Android Application should run regardless of availability of LED grid
    a) The above tests will be run without connection to an LED grid
3) User can interact with a 16x16 grid displayed on the Android Device
    a) User will interact with the displayed grid
4) Android Device can deliver the grid information to the LED Grid
    a) LED Grid will be monitored simultaneously to the phone grid to confirm that they are the same
5) LED Grid displays the correct information
    a) LED Grid will be monitored simultaneously to the phone grid to confirm that they are the same

Application Analysis

The User Interface will alternate between two main screens: A menu to select game mode, and then the main screen, that will always show the grid.
Depending on the game mode selected, the UI will be different, but the grid will always be displayed after selecting a game mode. A default prototype screen is provided for the Drawing Mode screen in Figure 1.

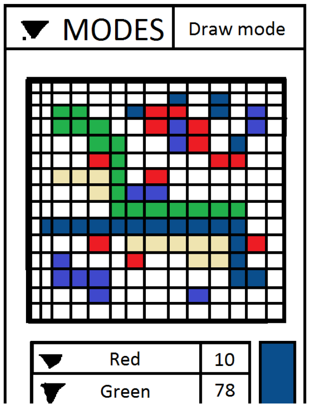


Figure 1 – Interface for the drawing activity on an android screen.

Domain Analysis and Integration

The interface between the Raspberry Pi and the Android App: The Raspberry Pi will be configured with an SSH server by setting up a Static IP on the host router for local connections on the network. We will not be configuring the server to be on a static global IP. Once this is achieved the Raspberry Pi can be connected to through http sockets. Raspberry Pi natively uses python (although java functionality is available) to setup a socket on the configured router. On the android application side of the connection, the android app will connect to the socket at the static IP. Once the connection has been established, commands to change the display of the LED grid can be sent.

For general use case, the server will only allow one socket connection to be made. In the case of Checkers, two socket connections may be made in parallel. For the touch table functionality, the aforementioned socket connection will be used such that the python table sends touch events back to the application.

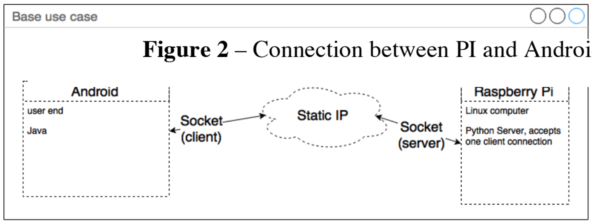


Figure 2 – Connection between PI and Android

Raspberry Pi to LED Communication:
The LED grid is controlled through the Serial Peripheral Interface (SPI) through general purpose input output (GPIO) pins on the Raspberry Pi. Physically, individual LED color is determined by and RGB value in a shift register. Shift registers are chained, so to control the LED grid, the SPI sends new values to the shift registers.

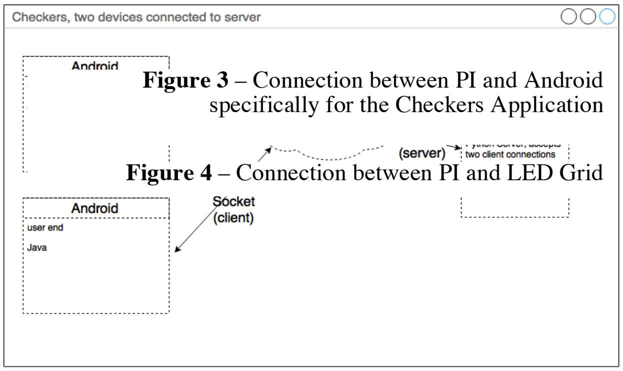
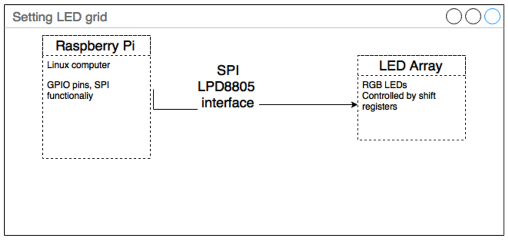


Figure 3 – Connection between PI and Android specifically for the Checkers Application

Figure 4 – Connection between PI and LED Grid



Important Algorithms

Fast Fourier Transform (FFT): The Fast Fourier Transform is based on the mathematical decomposition of the Discrete Fourier Transform (DFT). DFT translates a digital signal from the time domain into the frequency domain. This allows us to see which frequency bands have the most power. In reference to human hearing, this shows which pitches are being played the loudest, and if visualized creates an interesting display. The FFT improves on the DFT by summing smaller DFT of windowed frequencies together, or binning.

For our purpose the FFT will be performed on a window of 20 Hz to 22 kHz, the range of human hearing. The FFT requires the sampling frequency to be two times the max frequency. So for this function, only devices that have at least 44 kHz sampling freqency will be able to perform the FFT.

Class Design

It's clear we have three major components to this project. The first component is the Android application itself, through which the user interacts to manipulate the grid on the table. The second is the Raspberry PI of course, which needs to be able to reconfigure the LED strip to what the user has commanded. The third component, though not as apparent, is the communication between the phone and the raspberry pi. This is done by using the Command Design Pattern.

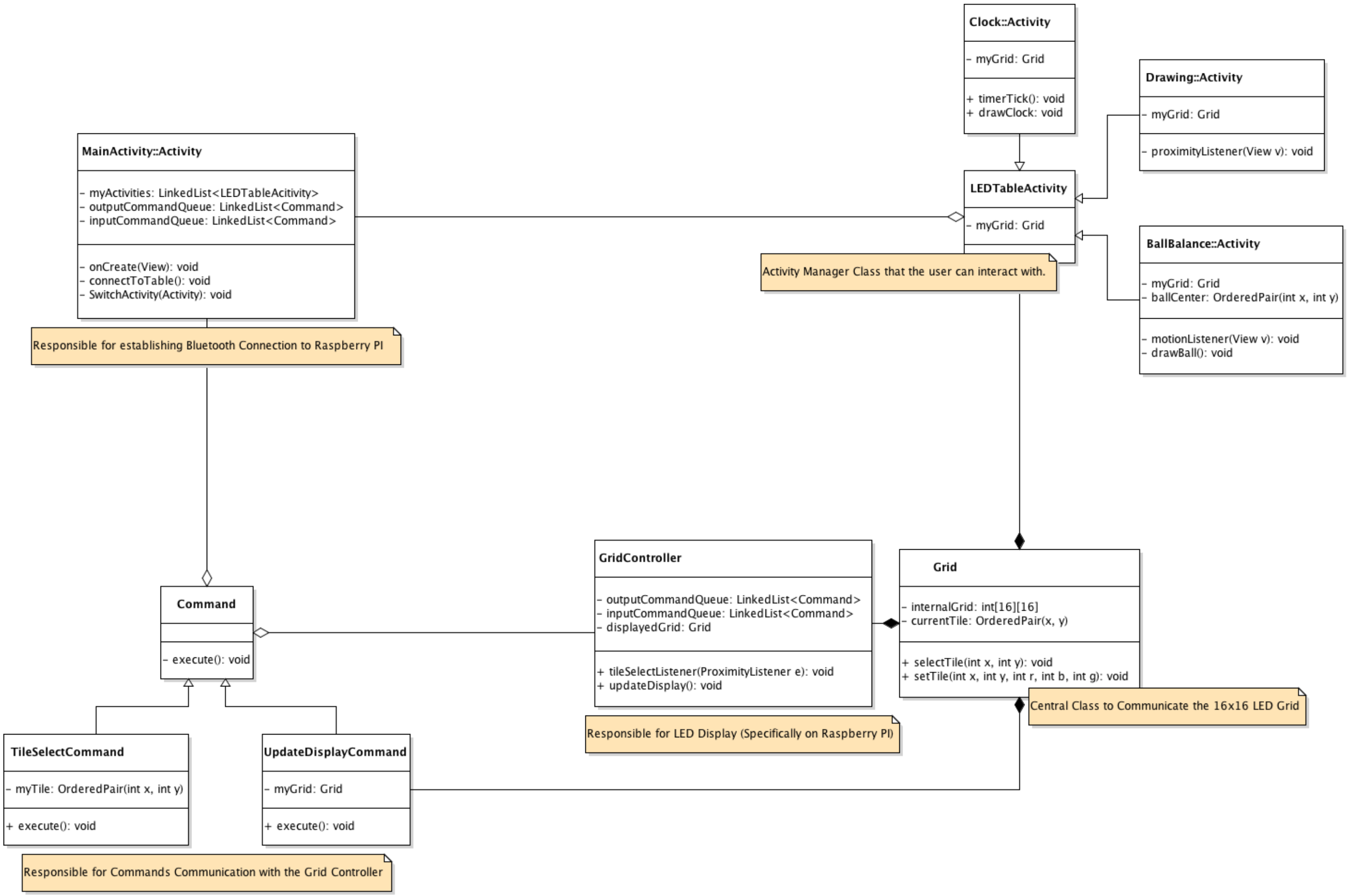

Clock::Activity
- myGrid: Grid
+ timerTick(): void
+ drawClock: void
Drawing::Activity
- myGrid: Grid
- proximityListener(View v): void
MainActivity::Activity
- myActivities: LinkedList<LEDTableAcitivity>
- outputCommandQueue: LinkedList<Command>
- inputCommandQueue: LinkedList<Command>
- onCreate(View): void
- connectToTable(): void
- SwitchActivity(Activity): void
LEDTableActivity
- myGrid: Grid
Activity Manager Class that the user can interact with.
BallBalance::Activity
- myGrid: Grid
- ballCenter: OrderedPair(int x, int y)
- motionListener(View v): void
- drawBall(): void
Responsible for establishing Bluetooth Connection to Raspberry PI
GridController
- outputCommandQueue: LinkedList<Command>
- inputCommandQueue: LinkedList<Command>
- displayedGrid: Grid
+ tileSelectListener(ProximityListener e): void
+ updateDisplay(): void
Grid
- internalGrid: int[16][16]
- currentTile: OrderedPair(x, y)
+ selectTile(int x, int y): void
+ setTile(int x, int y, int r, int b, int g): void
Command
- execute(): void
Central Class to Communicate the 16x16 LED Grid
Responsible for LED Display (Specifically on Raspberry PI)
TileSelectCommand
- myTile: OrderedPair(int x, int y)
+ execute(): void
UpdateDisplayCommand
- myGrid: Grid
+ execute(): void
Responsible for Commands Communication with the Grid Controller

Figure 5 – UML Diagram for Android Application

Task Allocation and Breakdown

We decided to focus on everyone's individual strengths for this project. Eugene will be responsible for developing the Android Application interface. Greg will be responsible for the communication between the application and the Raspberry PI. Lastly, Jay will be given the task of displaying the grid as well as pulling the proximity data. In terms of UML break down:

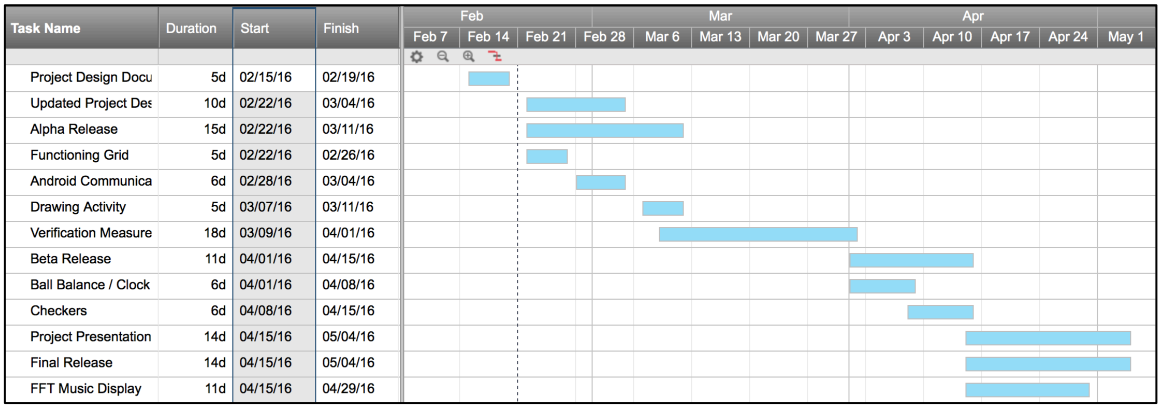
- Greg: Command (and the classes that inherit from Command) and Grid
- Eugene: LEDTableActitity (and the classes that inherit from LEDTableAcitity)
- Jay: GridController

Though the Grid class falls under Greg, everyone has the opportunity to make changes in order to facilitate implementing their own responsibilities.

Everyone is responsible for helping construct the LED Table Grid, made from selected addressable LED lines connected to a Raspberry PI.

Timeline for Completion

We've constructed a Gantt chart to display the major completion times for this project. During each release, our major goals are subjugated below to show the priority of the period.



Task Name	Duration	Start	Finish
Project Design Docu	5d	02/15/16	02/19/16
Updated Project Des	10d	02/22/16	03/04/16
Alpha Release	15d	02/22/16	03/11/16
Functioning Grid	5d	02/22/16	02/26/16
Android Communica	6d	02/28/16	03/04/16
Drawing Activity	5d	03/07/16	03/11/16
Verification Measure	18d	03/09/16	04/01/16
Beta Release	11d	04/01/16	04/15/16
Ball Balance / Clock	6d	04/01/16	04/08/16
Checkers	6d	04/08/16	04/15/16
Project Presentation	14d	04/15/16	05/04/16
Final Release	14d	04/15/16	05/04/16
FFT Music Display	11d	04/15/16	04/29/16

Table 1 – Gantt Timeline