Code Documentation

for

Digital Play Counter

Version 1.0 approved

Prepared by: Oksana Tkach

Esha Parikh

Marshall Nambiar

Eric Zhong

Rainbow Co.

June 9, 2017

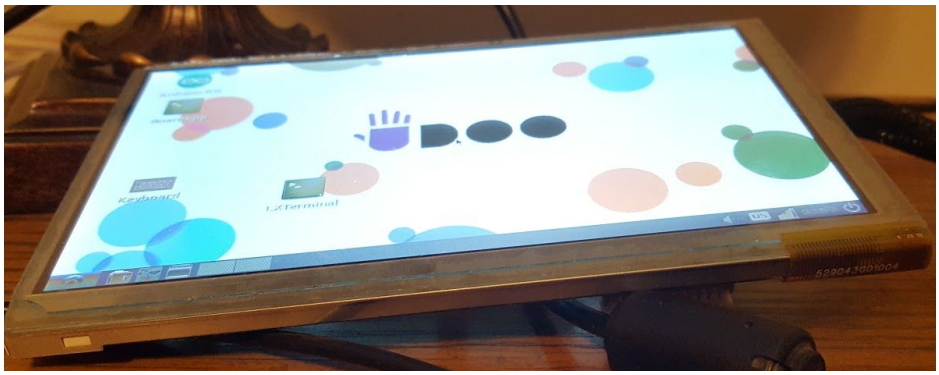
Table of Contents

Device Overview

The Digital Play Counter is a device that displays a number 0-99 on a 7 segment LED display, similar to the ones commonly seen in crosswalk signs and scoreboards. Although, it was originally designed to communicate a play number from a football coach on the sidelines to players on the field, the Digital Play Counter is applicable wherever you may need to display a number for any period of time. The Digital Play Counter is also advantageous to professionally made scoreboards because it is several times cheaper to create. A user can put the number they would like to display into the app on a touchscreen monitor, have that signal travel through to the UDOO, or brain of the device, then have that number get displayed on the 7 segment LED.

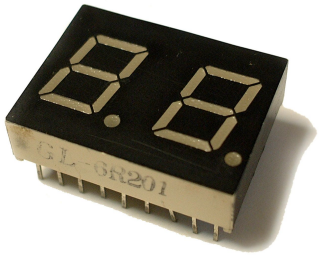
Hardware

LVDS 7” Touchscreen



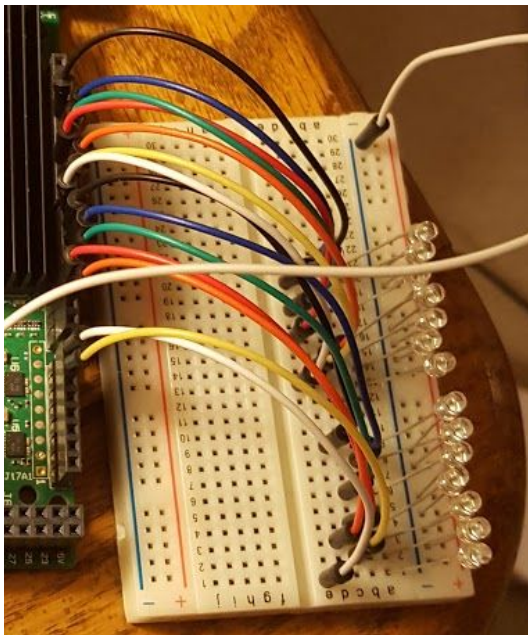
The LVDS touchscreen is custom made to connect to the UDOO Quad, the computer we will be using for this project. The LVDS serves as a monitor on which to view the app as well as a form of input. Our current touchscreen, however lost its touch capabilities and we now need to plug a mouse into one of the UDOO USB ports.

7 segment LED

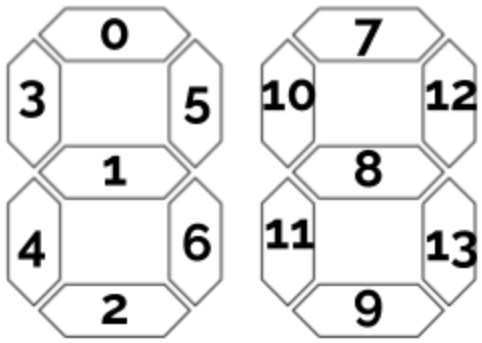


A 7 segment LED is a common output and display board. It is typically seen at sporting events, where it is used as the scoreboard.

Prototype 7 segment LED

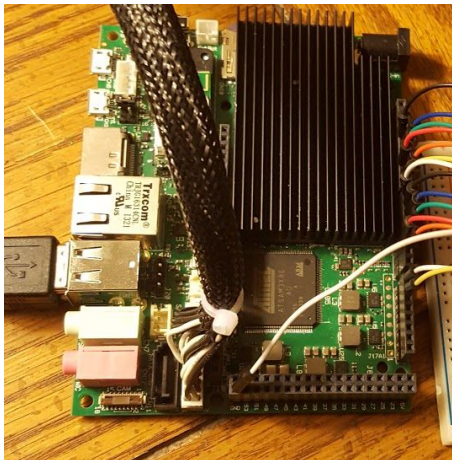


This prototype was created because we currently do not have a 7 segment LED display compatible with the UDOO's output voltage. Each cluster in the image above has 7 segments and acts like a seven segment LED. The LEDs can be numbered from 0-13 from the bottom up, and correspond with the following segments on a real 7 segment LED.



Please note that LEDs 0 and 1 lead to ports 14 and 15 on the UDOO respectively. This is because ports 0 and 1 were not working on this particular UDOO - they seem to have burned out. The port to write to was also a change made in the code, and any other UDOO should still be wired as it shown above to make sure that UDOO works correctly.

UDOO



This is the brain of the DIgital Play Counter. It runs the app that it outputs onto the lvds, decides which segments from the 7 segment LED need to be turned on/off and then turns them on/off.

Code

UDOObuntu

UDOObuntu is the operating system the UDOO runs on, such as Windows or OS X. Instead of being either of those operating systems, UDOObuntu is instead a spin off of another free, popular, and open source OS called Linux.

When the UDOObuntu is powered up, the UDOO should a desktop icon that says *BoardApp*. This isn't the program itself, just a shortcut to the program, but you can click on it to launch the program. If this doesn't work, you can launch the program manually by opening *lxterminal*, also found on the desktop, and typing in *java -jar BoardApp.jar*, then hitting return.

OperationPad

public OperationPad(InputGUI dad)

This creates a new panel, in the East of the InputGUI. It adds 3 JButtons to it; *Delete, Clear,* and *Enter.*

Delete removes the last digit from the number in the input field (the bottom text box/ beige rectangle in the graphical user interface). For example, if the input read "648" and you pressed delete, the input would now read "64". This button doesn't affect the number displayed on the number display board/with LEDs.

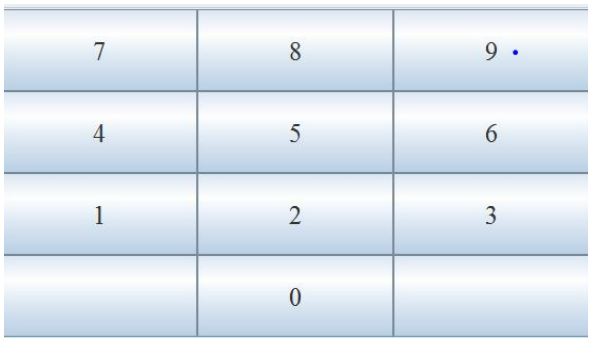
Clear clears/makes both the input text field and the output text field blank, as well as turns off all lights currently being displayed on the number display board.

Enter first clears the lights lit up on the number display to avoid gibberish after the next input turns on new lights. Then, it takes the data from the input text field, prints it to the output text field, and turns on the corresponding LEDs on the number display board to create the desired number being shown on the LED display.

NumberKeypad

public NumberKeypad(InputGUI dad)

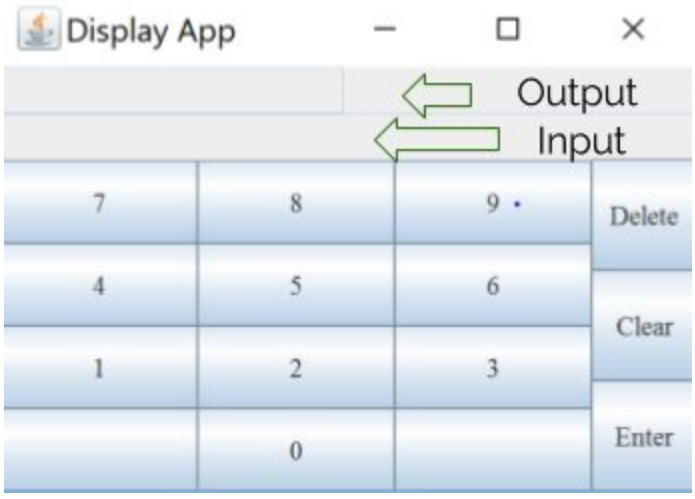
This creates a panel for the GUI as grid of digits 0-9 as shown below



Each of the digits corresponds to a group of LED lights that would light up the display number board. For example, clicking 11, would display 11 on the 7 segment LED. The blank buttons don't do anything, they are there for formatting purposes.

DisplayPanel

public DisplayPanel (InputGUI dad)



Creates a panel in the North section of the InputGUI that contains two text fields. The one found on top is the output text field, displaying the data that was entered before the *Enter* button was hit the last time. The bottom text field is the input text field, it displays the numbers you have selected from the digits but have not yet displayed to the digital number board.

public void deleteChar ()

Creates a string that is all but the last digit of what is currently in the input text field, then writes that string to the input text field. The effect becomes that of deleting the last number from the input text field.

public void clearOut ()

Sets both the input and output text fields to be an empty string.

public void setNumber ()

This transfers the number in the input text field to the output text field and clears the input text field for the next input. Note, this method does not control the digital display board, only the app.

public void addToInput (String ch)

This adds the String ch to the input text field. In this app, this would be a digit that was pressed getting added to the input text field.

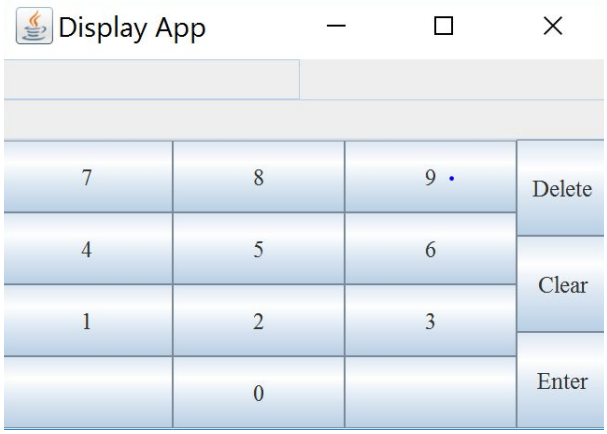
public String getOutput ()

This method returns the text that is in the output text field. This is used for control of how big of a number is inputted, as our digital display can only display number 0-99.

InputGUI file (Main file)

public final String beginPath = "/sys/class/gpio/"

public InputGUI (String title)



This constructor creates the graphical user interface and uses the previously aforementioned files to create a simple app that lets you click on numbers to choose a number.

public void actionPerformed (ActionEvent event)

This method determines which operational button has been pressed and acts accordingly. It reads the label of the pressed operational JButton.
Delete removes the last character from the input text field by calling deleteChar() from the DisplayPanel

Clear clears the text fields and turns off all segments by calling clearBoardAction() and clearOut() from the DisplayPanel

Enter first turns off all the current segments via clearBoardAction() then writes to the output field via DisplayPanel's setNumber()

public void setup()

This method goes through all 14 segments, gets their gpio file numbers from the method getGpioNum, and uses the printToFile method to write"out" to the file path "/sys/class/gpio/gpioXX/direction" where XX is the gpio number returned from getGpioNum. This switches the UDOO ports to output.

public int getGpioNum (int num)

This method returns the corresponding gpio (general pin input/output) for each port number. For example, port 0 corresponds to the file number gpio116.

public void printToFile (String filename, String contents)

Filename is not the full path of the file, just the “gpioXX/direction” or “gpioXX/value”. This method takes the beginPath variable, appends the filename variable to it, then prints the contents to the full file path.

public void clearBoardAction()

This method goes through all 14 segments and turns each of them off.

public void turnOnSegment (int seg)

This method takes the segment number as a parameter and writes 1 in the value file of the corresponding gpio directory.

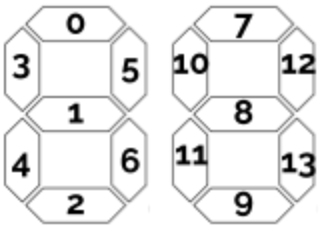
public void setNumber (String NumberStr)

Turns on the segments from the output text field.

Takes the output text field text as input, parses a 2 digit string from it. It then sends each digit over to getSegments to get an array of the segments it needs to turn on.. It goes through the given int array and turns on each of the indicated segments from the getSegments method.

public int[] getSegments (int num, boolean first)

Takes a single digit as input and a boolean that indicates whether it is the first or second digit. Returns the corresponding segment numbers that need to be turned to create that number based on diagram below.



public static void main(String[] args)

Initializes an InputGUI class.