Automated Scoring System for Ticket to Ride

Team 43 -- Matthew McCracken and Andrew Douglas
ECE 445 Project Proposal -- Fall 2017
TA: Kexin Hui
1 Introduction

1.1 Objective

Ticket to Ride is a fun game to play with friends, but when it comes to counting your score, it can put a strain on your relationship. The scoring process includes tallying your scores obtained from laying down train cars on certain paths and trusting that your friends did it correctly. It also includes the longest path bonus, which can take a while to count and determine whose path is longer. Destination cards can be confusing to determine. Besides keeping track of the score, it is also tedious to place the individual train cars on the board to claim a route. This can take up a lot of time and slows the game down unnecessarily.

Our solution to this problem is to automate the scoring process of this game. Automation will greatly increase enjoyment, alleviating the most cumbersome and time consuming part of the whole experience. The automation will include indicating the current player’s turn, showing current scores, and automatically calculating end-game scores including bonuses from destination cards and longest path. Instead of manually placing train cars on the board, LEDs will automatically be illuminated when a player captures a route. The color of the LEDs will indicate which player controls the route.

1.2 Background

From personal experience, tallying score is the worst part of Ticket to Ride. This sentiment is shared by Jacob Bryan (current TA for this course), who first suggested we implement the solution to this problem. There is feasible demand for a product like this, as there are automated versions of popular board games out there already, such as electronic Monopoly and The Game of Life. There was also a project in 2013 that automated the scoring process of Settlers of Catan that performed well - showing that there was a demand for automated scoring processes for board games like Ticket to Ride.

1.3 High-Level Requirements

- Train spaces should light up when the corresponding path is claimed.
- The game must be able to automatically calculate and display the players’ score during the game, as well as automatically factor in longest path bonus and destination cards at the end of the game.
- The game should automatically end when a player’s number of train cars dips to 2 or below.
2 Design

2.1 Block Diagram

The user input, control, and LED modules will take care of lighting the LEDs when a path is taken. The MCU will have dedicated memory to be able to store player scores as well as calculate scores in real time. The MCU will also keep track of number of train cars for each player, allowing for an automatic end to the game when the number of train cars goes below 2 for any player. The MCU accomplishes this by receiving signals from the control module that indicate what action each player took on their turn. These signals are then decoded by the MCU and used to update the game state. The LCD module is connected to the MCU and is used to give feedback to players about their current point totals. This screen will indicate when the game has ended and also show all of the players’ point totals so that the winner is determined immediately.
2.2 Physical Design

Figure 2: The Physical Board

Above is an image of the Ticket to Ride board game. Buttons will be placed at each of the different cities (the orange dots on the board) and will be used to indicate that a player is taking control of a rail line between two cities. The buttons will also allow a player to enter in the two endpoints listed on a destination card drawn. In addition to the buttons on the cities, there will be a few more buttons located in the lower left corner of the board allowing the users to indicate the number of players, pass the turn along to the next player, and indicate that they are entering a destination card rather than claiming a rail. Instead of placing the plastic trains to claim a route, LEDs will light up along each route to indicate that it has been claimed. The color of the LEDs on the path will be different depending on which player took control of the route. An LCD screen will be placed in the lower right corner of the board (where the scoring details are now located) and will display the current point total for each player and indicate which player is currently taking their turn. There will be a one inch deep wooden housing constructed that the game board will rest in - all of the wiring and electrical components can be placed in the housing to achieve a polished look in the finished product.

2.3 Functional Overview

2.3.2 Control System Module

The control module will take a 36 bit data signal input from the user interface module, as well as output a power line to the LED module and a 12 bit data signal to the MCU module. The control module will include logic set control signals necessary for gameplay, as well as handle any button presses that occur in the user interface, such as number of players, end turn, and destination card input mode. When a set of buttons are pressed, the control module will determine if the move is valid. This can be done by maintaining one bit flip flops for each of the
rail lines that indicate whether or not they have been claimed. If the move is not valid, the module sends a ‘not valid’ signal (12 bits of 1) to the MCU. If the move is valid, the control module will encode the path that was claimed, send the command to the MCU block, then turn on the LEDs associated with the path.

The encoded signal that will be sent to the MCU will be generated by a compression function. This is necessary because it would be impractical to send all 36 bits to the MCU, but a unique representation of each pair of cities is still necessary. In the 36 bit input vector, only two positions will have 1s at a time. These 1s correspond to the two cities between which a player would like to claim a route. Given this limitation, the compression function can simply output the number of zeros before the least significant 1 followed by the number of zeros between the least significant 1 and the most significant 1. Since the input is 36 bits long, the largest number of sequential zeros that could occur is 34. Six bits are needed to represent the number 34 and thus the output of the compression must be 12 bits in order to correctly handle all possible inputs. An example of the inputs and outputs to this function is as follows:

<table>
<thead>
<tr>
<th>Input (36 bits)</th>
<th>Output (12 bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0...011</td>
<td>000000 000000</td>
</tr>
<tr>
<td>0...010</td>
<td>000000 000001</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>110...0</td>
<td>100010 000000</td>
</tr>
</tbody>
</table>

Table 1: Compression Function

2.4 Block Requirements

<table>
<thead>
<tr>
<th>Module</th>
<th>Requirement</th>
</tr>
</thead>
</table>
| Control System       | • Compress 36-bit data from the User-Input module into 12-bit data to be sent to the MCU.  
                        | • Input 5V and 2.5A so we can provide 5V with a tolerance of ± 0.5V and 25mA ± 5mA to the correct LEDs (based on user input) in less than 0.25 s. |

Table 2: List of Requirements by Module

Verifications

<table>
<thead>
<tr>
<th>Module</th>
<th>Verification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control System</td>
<td>• Use oscilloscope to compare input-output voltages from logic gates</td>
</tr>
</tbody>
</table>

4
and ensure the correct compression is output.
- Use multimeter to determine input and output voltage and current. Timing will be verified using a stopwatch to determine the delay between buttons pressed and LEDs turning on.

Table 3: List of Verifications by Module

Figure 3: Block diagram of compression I/O
Figure 4: Flowchart of compression algorithm
Figure 5: Circuit schematic of compression algorithm

<table>
<thead>
<tr>
<th>Part</th>
<th>Price / part</th>
<th>Quantity</th>
<th>Price</th>
<th>Running total</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 bit parallel-in serial-out shift register</td>
<td>$0.39</td>
<td>5</td>
<td>$1.95</td>
<td>$1.95</td>
</tr>
<tr>
<td>4 bit parallel-in parallel-out shift register</td>
<td>$2.49</td>
<td>3</td>
<td>$7.47</td>
<td>$9.42</td>
</tr>
<tr>
<td>10 bit counter</td>
<td>$0.51</td>
<td>1</td>
<td>$0.51</td>
<td>$9.93</td>
</tr>
</tbody>
</table>

Table 4: Pricing information for data-compression[2]
2.5 Risk Analysis

The control block is the block that poses the most risk to the completion of this project. The control unit needs to control LEDs, encode path data, and send it to the MCU for score calculation. It is the hardest module to implement solely because of the complexity of the hardware involved. There are a large number of routes that all have to produce a unique code that can be sent to the MCU. This unit will also be responsible for checking whether or not a move is valid. It will have to keep track of which routes are claimed and implement logic to check that the move a player has entered does not interfere with an existing route. Both of these components will require a significant amount of hardware design that can be time consuming and difficult to troubleshoot. The MCU module could also pose some issues as neither of us are very familiar with using an MCU, but we know C and have used an Arduino before so there should be enough background knowledge to carry us through that module.

3 Ethics and Safety

We will uphold the IEEE code of ethics[1] during the development of our project. Any intellectual property concerns about building upon an existing board game will be avoided as we do not intend to sell the final product, however we will do our due diligence as to not plagiarize other works while designing our project. Our project has very minimal safety concerns, and could even improve upon the safety of the original board game. We will be eliminating the need for small game pieces and thus will be able to remove a choking hazard for small children. All of our electronics will be safely housed underneath the board in an insulated environment. We will take care that there are no exposed electrical components that could harm users. We will also put in a fail-safe to protect our circuit should the batteries be put in the wrong way, so there will be no chance of fire or frying our circuit.

References