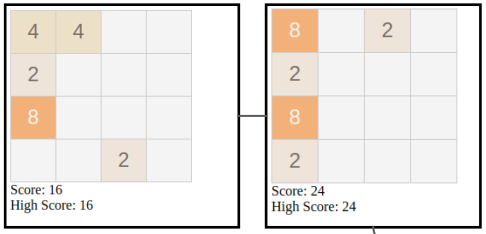
2048 on Pi

Rules of 2048

- Played on a 4x4 grid of tiles, each with their own value
- The player can slide the tiles in a cardinal direction
- A new tile will appear after each move (a 2 or a 4)
- If two tiles with the same value collide, they merge
- The primary objective is to reach the 2048 tile
- The game ends when there are no valid moves left

Included Features

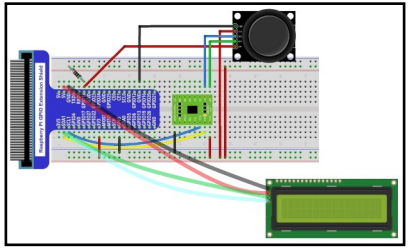
- WASD Keyboard controls to move tiles + space for other input
- Joystick controls to move tiles + the z-axis for other input
- External LCD display for the score and high score
- Runs via a locally hosted server
- Uses multiple threads to handle Joystick input in parallel
- Browser GUI with CSS interface



Graphical User Interface

Slide tiles with the Joystick

Display score and high score



ADS7830 ADC
- SDA → SDA1
- SCL → SCL1

JOYSTICK
- GND → GND
- +5V → +5V (VCC)
- VRX → A1 (ADS7830 ADC)
- VRY → A0 (ADS7830 ADC)
- SW → GPIO18

LCD
- GND → GND
- +5V → +5V
- SDA → SDA1
- SCL → SCL1

Code Snippets

The X and Y axes of the joystick are read from channel 3 of AD1 and AD0 respectively

```
xAxisVal = handle_joystick_input.joystickADC.analogRead(ADC_CHANNEL_X)
yAxisVal = handle_joystick_input.joystickADC.analogRead(ADC_CHANNEL_Y)
```

The game is played with a 4x4 grid of ints, with critical game data stored in a global container

```
grid = [[0 for _ in range(GRID_WIDTH)] for _ in range(GRID_WIDTH)]
GAME_STATE = {'grid':new_game(), 'score':0, 'high_score':0}
```

Keyboard input is handled via an Event Listener, whereas the Joystick polls for updates on the client and server

```
document.addEventListener('keydown', function(event) {
if(['w', 'a', 's', 'd'].includes(event.key.toLowerCase())) { …
```

```
function pollJoystickInput() {
  let xhr = new XMLHttpRequest();
  xhr.open('GET', '/joystick_move');
  xhr.onreadystatechange = function() { …
```