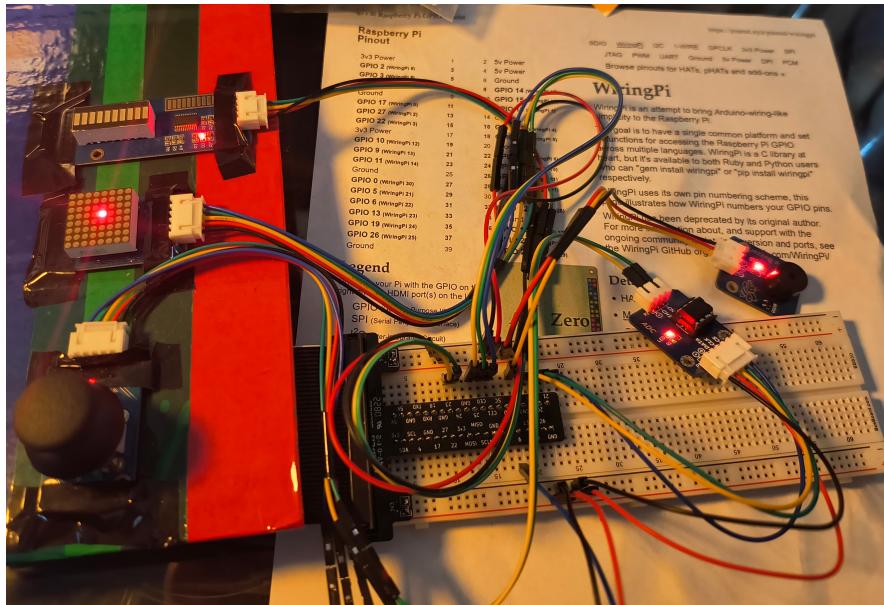


# RPI Simon-Inspired Pattern Game

A Raspberry Pi controlled Simon-inspired pattern game. Match the directions in the correct sequence to continue to the next level.

Created by [Miles Murgaw](#) as a final project for CNIT 105 @ Purdue Polytechnic.



## Features:

- 8x8 LED Matrix for displaying directions, and other effects. Main display.
- LED Bar for displaying current level.
- Passive buzzer for audio effects and tunes.
- XY + Z Button Joystick for matching patterns. Primary input.

## Project Organization:

This project is organized into a handful of files.

- `main.c` contains the main game logic and control flow.
- `led_matrix.c` contains the led matrix rendering logic.
- `led_bar.c` contains the led bar rendering logic.
- `buzzer.c` contains the tunes and audio effect logic.
- `joystick.c` contains the logic to read joystick input, including a bit-banged protocol implementation for the ADC ADC0832.

# Startup Flow

The program runs in only two threads: the main game thread, and a secondary thread for rendering the LED matrix.

Below is the described step-by-step program flow.

1. Upon startup, peripherals are initialized, including the matrix LED which spawns the render thread.
2. The program will start the core loop which handles starting the game when the user is ready.
3. If the joystick button is pressed, the game will start.

## LED Matrix Render Thread

The LED Matrix can only enable individual LEDs at a time, and in order to display pictures, we must quickly loop through and turn on and off the individual leds that each frame requires, creating the illusion of a picture.

During initialization, the LED matrix spawns a new thread who's sole job is to continuously render the currently selected frame. This allows the main game thread to continue executing without being blocked or limited by the LED matrix render thread.

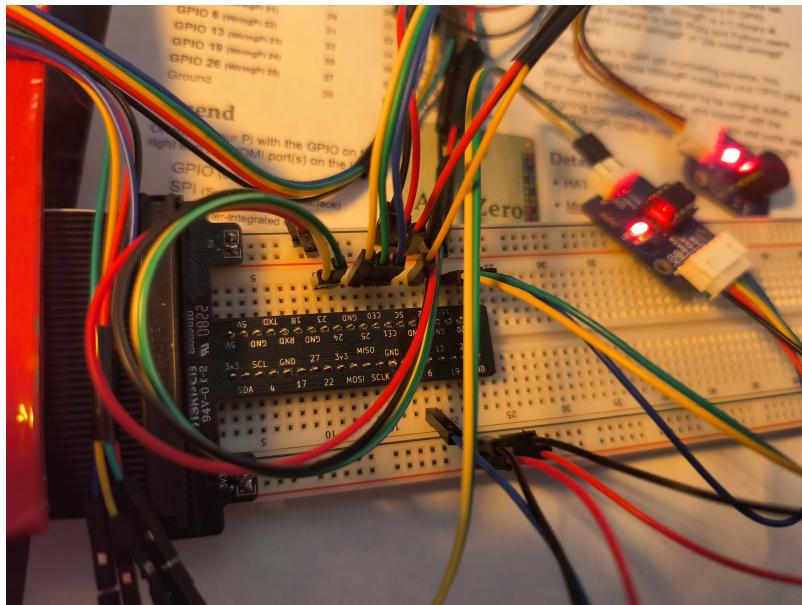
## Game Flow

The game is organized into a single infinite while loop.

1. A new pattern will be chosen at random and added to the list of patterns to match. The pattern will be briefly displayed to the user.
2. A new loops runs until the number of inputs matches the total number of patterns to match. If the user incorrectly matches a pattern, the loop exits and indicates a failed state.
3. If the previous loop indicated a fail state, the game will indicate that the user lost, and will exit the core game loop.
4. If the user succeeded in matching all patterns, the level will increase and the game loop will repeat.

## Pin Descriptions

Each file contains it's required pin definitions used by the wiringPi library. Each device has it's own PWR and GND, all connected to 5V, other than the joystick and ADC which uses 3.3V.



## Joystick

The joystick uses an ADC for the analog inputs.

<b>Breakout Pin</b>	<b>RPI GPIO</b>	<b>ADC</b>
Y	N/A	CH1
X	N/A	CH0
Z	GPIO 06	N/A

## ADC

The ADC is used to read analog joystick signals.

<b>Breakout Pin</b>	<b>RPI GPIO</b>
DATA	GPIO 16
CLK	GPIO 20
CS	GPIO 21

## LED Matrix

<b>Breakout Pin</b>	<b>RPI GPIO</b>
ST_CP	GPIO 25
SH_CP	GPIO 08
DS	GPIO 07

## LED Bar

<b>Breakout Pin</b>	<b>RPI GPIO</b>
DCKI	GPIO 23
DI	GPIO 24

## Passive Buzzer

<b>Breakout Pin</b>	<b>RPI GPIO</b>
S	GPIO 12

## Compiling

This project is meant to be compiled on a Raspberry Pi. This has only been tested on an RPI 4B.

Debug:

```
gcc src/main.c src/led_matrix.c src/led_bar.c src/buzzer.c src/joystick.c
```

Release:

```
gcc src/main.c src/led_matrix.c src/led_bar.c src/buzzer.c src/joystick.c
```