Dino Game User Manual

Team Members:

Creighton Bennett

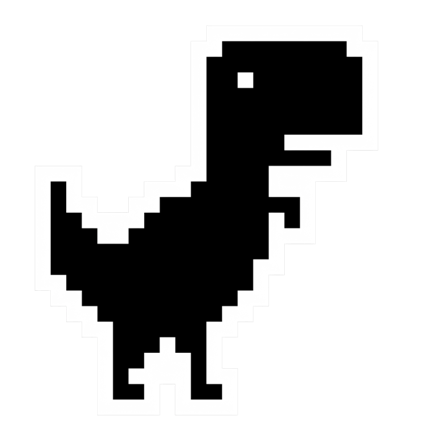
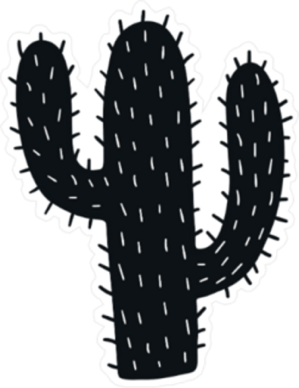
Kaden Jackson-Dietzer

Eshan Mathur

Rebecca Patino

Minghan Wang

Peer Mentor:  
Jakob Dieffenbach

# Introduction

This game is based off the classic google chrome dinosaur game, which most people can play when they do not have access to the internet. The user attempts to keep the game alive by jumping over cacti that appear from off the right of the screen. The user attempts to get the highest score possible, and the score increases in accordance with the amount of time the user spends alive.

There are only two buttons for the user to input anything

PB [0]: Action

* This starts the game
* Makes the dinosaur Jump

PB [1]: Reset

* This will stop the game and restart the score tracker

Our displays consist of one ili9341 display and two Seven-Segment Displays (SSD). We use the ili9341 display to display the images and the gameplay, and the Seven-Segment Displays to display the final score.

Our game has 4 modes:

IDLE

RUN

GAMEOVER0

GAMEOVER99

The user may switch between the IDLE mode and RUN mode with PB [1]. The mode GAMEOVER0 is after the user collides with a cactus and is not at the maximum allotted score. GAMEOVER99 is the mode where the user has reached the maximum score possible before collision with a cactus.

# *Gameplay*

When the game is enabled, the player will start in the IDLE mode. By pressing PB [0] (the jump button), the player will begin the RUN mode, where the cacti are generating, and the player must dodge them.

## Difficulty Scaling:

As the user advances in the game, the difficulty escalates progressively. This is accomplished by increasing the speed of the cacti's movement (shift towards the user), requiring the user to jump sooner and navigate obstacles with greater attention and precision.

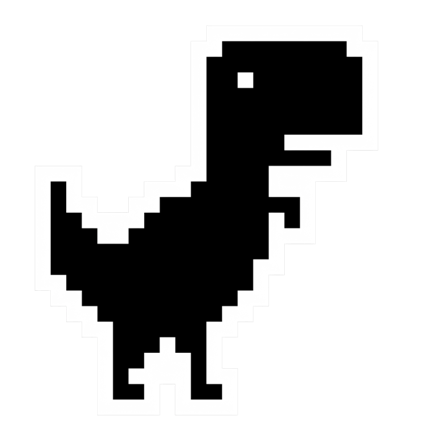
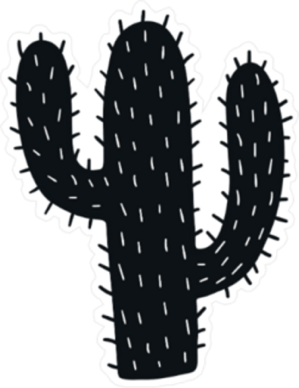
## RUN Mode:

RUN mode is entered when the user pushes PB [0] (jump button) while the game is in the IDLE state. Once the button is pushed, the screen begins to shift the randomly generated cacti towards the dino. The user will then have to time the jumps using PB [0] to avoid collision with the cacti. As mentioned earlier, as the game progresses, and the user accumulates a higher score, the difficulty rating changes accordingly to the Difficulty Scaling module. You can reset the game to the IDLE mode at any point in the run mode by pushing PB [1] (reset button). If the user collides with a cactus, the designated game over mode is displayed.

## Game Over Modes:

There are two games over modes that could appear to the user. If the user collides with a cactus before reaching the maximum possible score (99), the GAMEOVER0 mode will be placed into effect. In GAMEOVER0, the users’ score that was reached before collision is displayed, and the user then must push PB [1] (reset) to return to the IDLE state.

The other mode, GAMEOVER99, is when the user reaches a score of 99 (the maximum possible score for our game), the game is stopped (regardless of if there is a collision with a cacti) and set into the GAMEOVER99 state. In this state, a final score of 99 is displayed, and similarly to the GAMEOVER0 mode, the user must push PB [1] (reset) to exit that mode and enter the IDLE mode.

***Appendix 1 – Project Overview Flow Chart***

A diagram of a computer program

Description automatically generated

***Appendix 2 – Basic Wiring Diagram and GPIO Pin Output***

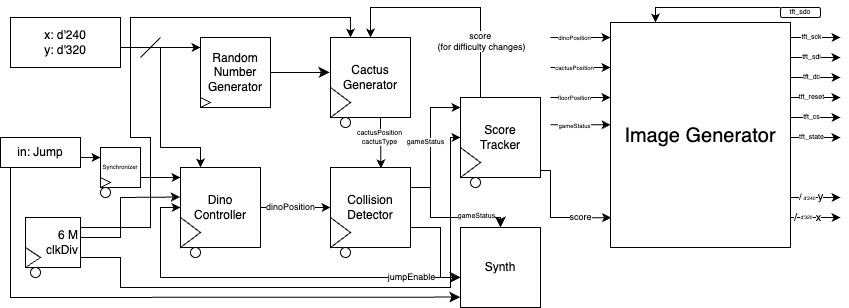
A circuit board with wires and a display

Description automatically generated

|  |  |
| --- | --- |
| **GPIO Pin:** | **Function:** |
| **GPIO[0:6]** | **Pins to the 7 segments of the display** |
| **GPIO[7]** | **Toggle power to the two SSDs** |
| **GPIO[15]**  **GPIO[18]** | **Button input 1**  **Button output 1** |
| **GPIO[19]**  **GPIO[22]** | **Button input 2**  **Button output 2** |

***­­­***

­­­­­­­­­

­­­­­

**Inputs-** There are two starting inputs that must be considered. The first is if the user pushes the jump button (which of course will first be pushed through a synchronizer before it gets pushed to any other module). The second is the starting location of the Dino, in coordinate form for the pixel formation (x, y).

**Dino Controller –** In this module, you should aim to implement the controls to make the dinosaur jump. It will take the input from the Jump input, the universal 6 M clock Divider, the jump input from the user, the enable jump from the collision detector, and the dino coordinates. In this module you should just program the dino jump dynamics, which is essentially just how the pixels will shift, and calculate where the final dino position is. There should only be one output, and that is the dino Positio

**Random Number Generator-** In this module, you will generate a random number based off multiple frequency dividers timed through logic to offset it. It is very important to note that when the user presses the button, you must reset the clocks so that there is no overlap and the signals of the clock running into the RNG are not continuing.

**Cactus Gen –** This module should be used to send the instructions to the image generator and essentially generate all for the cacti. This module takes inputs from the RNG, 6 M clock Divider, and the score Tracker. ChatGPT. In this module, you will utilize RNG inputs within a set of case statements to determine the pixel width between cactus 1 and cactus 2. Two counters will be employed to increase the position of the second cactus and record its values. This ensures that when the cactus reaches the maximum screen pixel width, it moves off-screen. The process involves generating two cacti, with the first spawning at a fixed location and the second at a random distance in pixels after it. By incrementing and monitoring only the second cactus, you can determine when to regenerate the cacti. As the score increases, adjustments will be made to increase spawn speed, aligning it with the new shift speed based on the 6 MHz clock. The RNG will also be used in a case statement to determine the types of cacti spawning in (tall, medium or small) There would be two outputs, cactus position and cactus type.

**Collision Detector-** In this module you will be writing the code to detect the collision between the cactus and dinosaur. This will be done by taking the inputs, cactus type, cactus position, and dino position. In this module, you will determine the lowest Y coordinate pixel location of the dinosaur's position and compare it with the Y pixel coordinates of the cactus position. If the dinosaur's Y coordinate is equal to or below the cactus height and their X coordinates align, a collision is detected. Similarly, a collision is identified if the cactus pixel position matches the dinosaur's Y coordinate at any point. It also determines if the user can jump. To prevent the user from “double jumping” or being able to jump while already mid jump, if the user is in jump, it sets an enable signal low. Once landed it returns to high. This module has two outputs, game status and jump enable.

**Score Tracker-** In the score tracker module, inputs are received from the game status and the 6 MHz clock divider. This module monitors collisions: if a collision occurs, the score stops increasing. However, if no collision is detected, the score increases by 1 point for every 10 seconds that the player avoids collisions. The elapsed seconds are determined using the clock divider.

**Image Generator –** In the image generator module, users utilize it to generate images on the ili9341 screen. This module receives inputs from the score, game status, dinosaur position, floor position, and cactus position. Within the image generator, code is written to render each pixel on the screen. It processes these inputs and invokes other functions to compute the placement and display of pixels accordingly.

**Synth –** The Synth is what we created to have in game sound when the dino is running, a bouncing sound when the dino jumps, and a ‘death’ sound that plays when there is a collision with a cactus. It is your basic synth module creation.