Crocodile Dentist

Embedded System Final Report

Z123332 CHEN, HE-MIN

End Term Project

2024/1/17

Objective

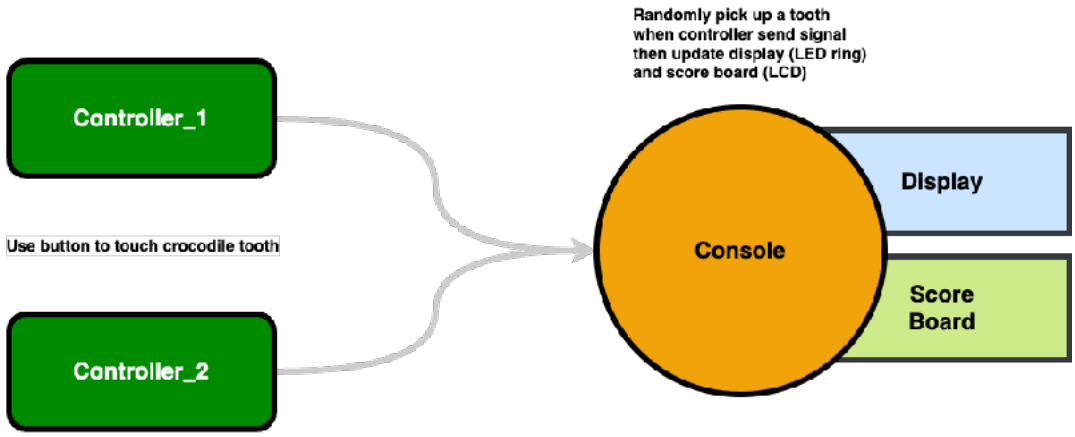
The first time I heard of this final project, I came up with an idea to make a small game. I've tried many design and faced some failure, and finally I made the simple but exciting game, Crocodile Dentist, a roulette game.

Instruction

How to play Crocodile Dentist?

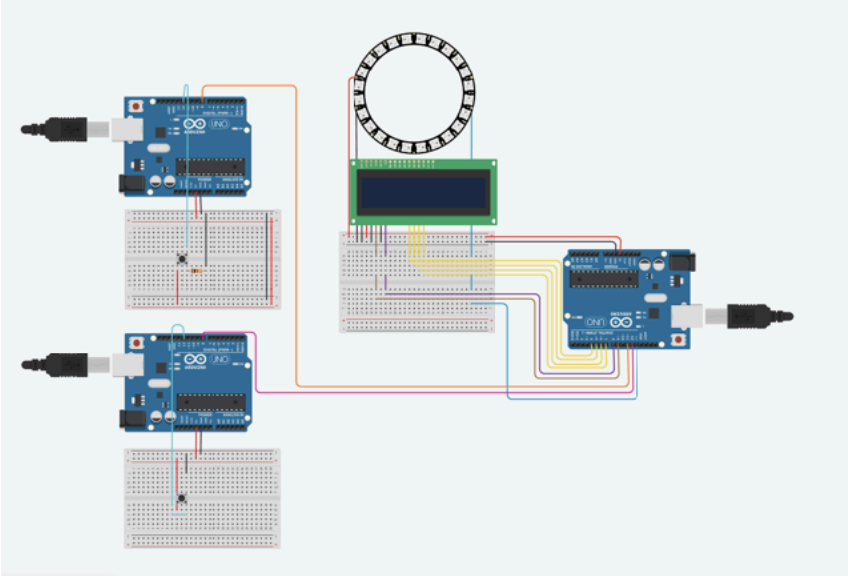
- Two players touch the crocodile tooth in turn.
- The one touch the tooth that will trigger crocodile to bite loses

By the two rules above, I design a hardware and you can see its abstraction below. Players will use each controller to act as touch crocodile's tooth by touch the button. Then the console will randomly choose a tooth that haven't been chosen. If the console pick up a tooth that is set to be the tooth will trigger the crocodile to bite in a player's turn, that player lose.



Hardware and Software Structure

Hardware



The whole system consist mainly by two parts, controllers (2) and console.

- Controller (Left)

  Each controller has a button (on port 12) and a cable on port 8 connect to the console. Players perform touch action by using the button in each turn. Every touch will inverse the state on the port 8 then trigger interruption on console.

- Console (Right)

  When console receives the interruption causing by the state change on the cable (on port 11, 12) connecting to each controller, the console will randomly pick up a LED on the Neopixel LED ring that haven't been light up and show message on the LCD display.

Software

- Controller

```
#include <stdio.h>
#include <stdlib.h>

/* Ports */
#define BUTTON  12
#define INSTRUCT    8

/* State Machine */
enum action_set {
  A_RELEASE,
  A_PRESS,
  A_MAX
};

struct action_t {
  enum action_set action;
};

enum state_set {
  S_IDLE,
  S_PRESSED,
  S_RELEASED,
  S_MAX
};

struct state_t {
  enum state_set state;
  void (*trigger)();
};

struct action_t get_current_action() {
  struct action_t current;

  switch(digitalRead(BUTTON)) {
    case HIGH:
      current.action = A_PRESS;
      break;
    default:
      current.action = A_RELEASE;
  }

  return current;
}
```

```
void stand_by() {}

void turn() {
  digitalWrite(INSTRUCT, !digitalRead(INSTRUCT));
}

struct state_t next_state[S_MAX][A_MAX] = {
  [S_IDLE] = {
    [A_RELEASE] = { S_IDLE, stand_by },
    [A_PRESS] = { S_PRESSED, stand_by }
  },
  [S_PRESSED] = {
    [A_RELEASE] = { S_RELEASED, stand_by },
    [A_PRESS] = { S_PRESSED, stand_by }
  },
  [S_RELEASED] = {
    [A_RELEASE] = { S_IDLE, turn },
    [A_PRESS] = { S_IDLE, turn }
  }
};

void setup() {
  Serial.begin(9600);

  pinMode(BUTTON, INPUT);
  pinMode(INSTRUCT, OUTPUT);

  digitalWrite(INSTRUCT, HIGH);
}

struct state_t s_current = { S_IDLE, NULL };

void loop() {
  digitalWrite(LED_BUILTIN, digitalRead(INSTRUCT));
  Serial.print(digitalRead(INSTRUCT));
  Serial.print('\n');
  struct action_t a_current = get_current_action();
  s_current = next_state[s_current.state][a_current.action];
  (*s_current.trigger)();
  delay(50);
}
```

- Problem

  The button's state machine can successfully light up built-in LED when the signal cable's state change but I'm not sure why the signal cannot be receive on the console side.

-

- Console

```
#include <stdio.h>
#include <stdlib.h>
#include <Adafruit_NeoPixel.h>
#include <LiquidCrystal.h>

/* LED(Display) */
#define LED 13
#define LED_SIZE    24
/* LCD(Score Board) */
#define RS  9
#define EN  8
#define DB4 4
#define DB5 5
#define DB6 6
#define DB7 7
/* Other ports */
#define CONTROLLER_1    11
#define CONTROLLER_2    12

int turn_count = 0;
int winner = 0;
int bomb_pos = 0;
int teeth[LED_SIZE] = {0};

//initialize LED and LCD
Adafruit_NeoPixel display = Adafruit_NeoPixel(LED_SIZE, LED, NEO_GRB + NEO_KHZ800);
LiquidCrystal score_board(RS, EN, DB4, DB5, DB6, DB7);
```

```
void update_led(int mode) {
  switch(mode) {
    case 0: // light up each LED on ring in random color
    for(int i = 0; i < LED_SIZE; i++) {
      display.setPixelColor(i, random(255), random(255), random(255));
      display.show();
      delay(500);
      display.clear();
    }
      break;
    case 1: // light up LED based on the state of teeth
      for(int i = 0; i < LED_SIZE; i++) {
        switch(teeth[i]) {
          case 1:   // player 1
            display.setPixelColor(i, 0, 255, 0);
            break;
          case 2:   // player 2
            display.setPixelColor(i, 0, 0, 255);
        }
      }
      display.show();
  }
}

// set the place will trigger crocodile to bite (lose)
void set_bomb() {
  bomb_pos = random(LED_SIZE);
  teeth[bomb_pos] = -1;
}
```

```
// what happen when touch
void touch() {
  int touch_pos;
  int player = turn_count%2 + 1;

  while(1) {
    touch_pos = random(LED_SIZE);   // pick a tooth, if s
    if(teeth[touch_pos] == 0) {     // a tooth not been touched
      teeth[touch_pos] = player;
      score_board.print("Player");
      score_board.print(player);
      score_board.print(" touch ");
      score_board.print(touch_pos);
      break;
    }
    if(teeth[touch_pos] == -1) {    // a tooth will trigger bite
      for(int i = 0; i < 3; i++) {
        display.setPixelColor(bomb_pos, 255, 0, 0);
      }
      winner = (player == 1)? 2: 1; // set winner
      break;
    }
  }
}

void check_winner() {
  if(winner != 0) {
    score_board.print("Game set, winner is player");
    score_board.print(winner);
    update_led(0);
    exit(0);
  }
}

void on_push() {
  touch();
}
```

```
void setup() {
  Serial.begin(9600);
  pinMode(CONTROLLER_1, INPUT_PULLUP);
  pinMode(CONTROLLER_2, INPUT_PULLUP);

  attachInterrupt(digitalPinToInterrupt(CONTROLLER_1), on_push, FALLING);
  attachInterrupt(digitalPinToInterrupt(CONTROLLER_2), on_push, FALLING);

  /* Initialize */
  set_bomb();   // set the tooth that will trigger crocodile to bite
  // initialize score board (LCD)
  score_board.begin(16, 2);
  score_board.clear();
  score_board.setCursor(0, 1);
  // initialize display (LED)
  display.begin();
  display.clear();
  update_led(0);
  for(int i = 0; i < 3; i++) {
    score_board.print("Start !!");
    delay(500);
    score_board.clear();
  }
}

void loop() {
  update_led(1);
  check_winner();
  turn_count += 1;
  score_board.clear();
  delay(500);
}
```