SIMON REIMAGINED

Embedded System Project Documentation

Created by: Adam Gal Version: 1.0

Date: 2025.12.26

Project Summary

A modern reinterpretation of the classic Simon memory game, featuring:

- **Custom PCB**
- **Arduino Nano firmware**
- **3D-printed mechanical design**
- **Full simulation & testing**
- **Complete assembly documentation**

Portfolio Use

Personal Embedded Engineering Portfolio Project

1. Project Overview

Simon Reimagined is a redesigned memory game based on the classic Simon toy, built around an Arduino Nano microcontroller. The goal is to create a fully functional embedded system that challenges users to memorize and repeat increasingly complex sequences of lights and sounds. The project includes a custom PCB, 3D-printed housing, and documented firmware.

---

2. Hardware Architecture

2.1 Key Components

- **Arduino Nano** – microcontroller
- **4 colored push buttons** – user input (red, blue, green, yellow)
- **4 LEDs** – visual feedback
- **Buzzer** – audio feedback
- **Toggle switch (ON/OFF)** – power control
- **Custom circular PCB** – central layout for all components

2.2 PCB Design

Designed by Adam Gal using EasyEDA. Version: V1.0. Included documents:

- **SCH_v1.pdf** – schematic diagram
- **Bottom_Plastic.pdf, Top_Plastic.pdf, Latch_Plastic.pdf, Button.pdf** – mechanical drawings
- **Assembly.pdf** – assembly guide

---

3. Firmware

3.1 Features

- **Game start**: triggered by a dedicated start button
- **Sequence generation**: random color and tone sequence
- **User input**: player repeats the sequence
- **Error handling**: incorrect input triggers buzzer and LED feedback, then resets

---

4. Mechanical Design

4.1 3D Structure

The housing consists of fourt main parts:

- **Bottom shell** – mounting points and power access
- **Top shell** – cutouts for buttons and LEDs
- **Latch element** – stabilizes button placement
- **Buttons** - Custom-designed colored plastic buttons aligned with top shell cutouts.

---

5. Simulation and Testing

Simulated in Tinkercad:

- **LED and buzzer behavior**
- **Button input and game logic**
- **Voltage monitoring and debugging**

---

6. Assembly

Based on Assembly.pdf:

- Mount Arduino Nano
- Connect buttons and LEDs
- Attach buzzer and toggle switch
- Assemble housing (screws and alignment)

---

7. Attachments

- Firmware code (Arduino IDE compatible)
- SCH and PCB designs (EasyEDA)
- Mechanical drawings (PDF format)
- 3D renders and photos of the final device
- Simulation environment (Tinkercad)

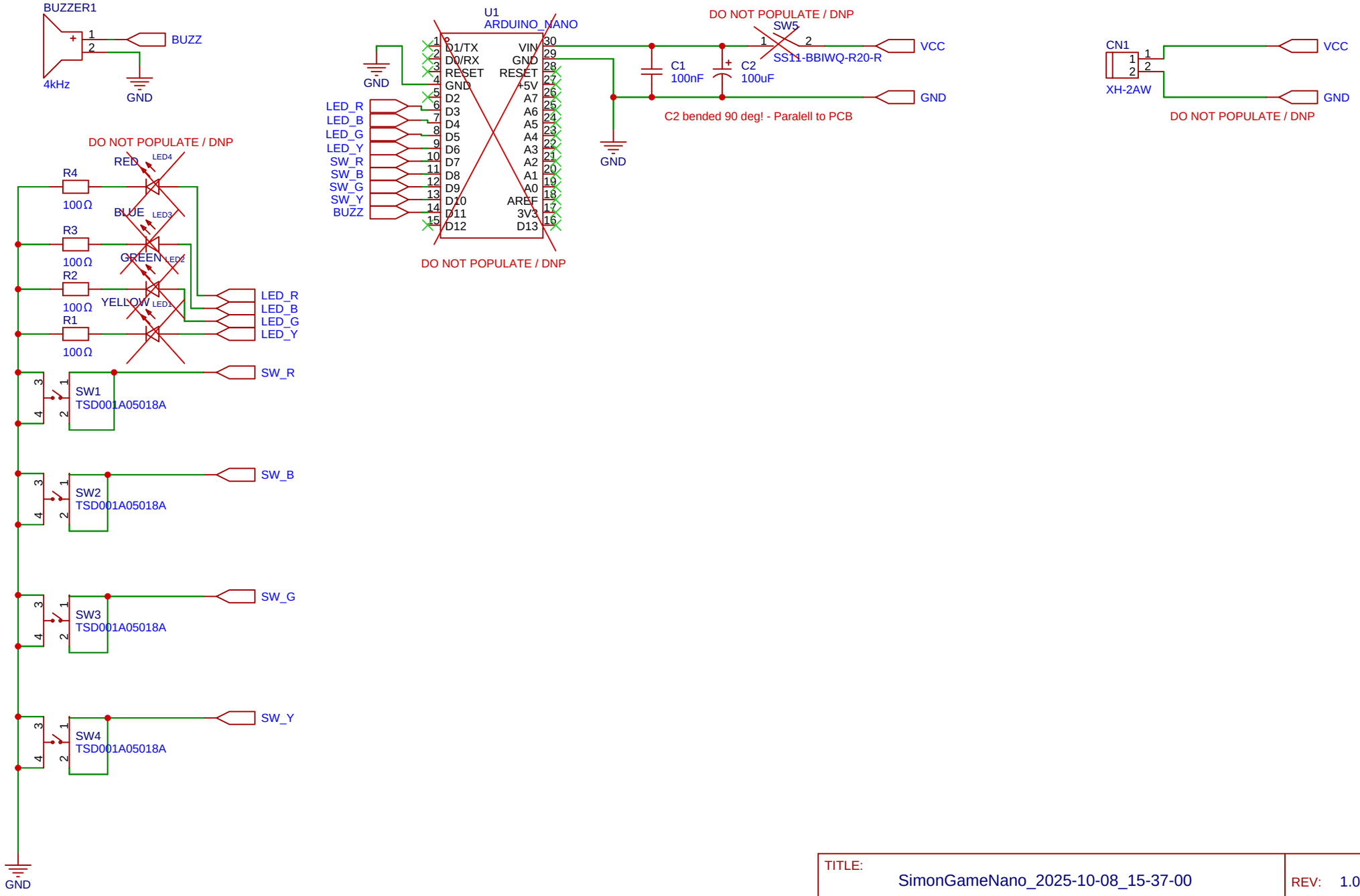
```
//DATE: 2025.10.08
//Author: Gal Adam
// ======================================
// SIMON GAME - Arduino Nano
// ======================================
// A memory game where players repeat an increasingly long sequence of colors.
// - 4 colored buttons with LEDs
// - Buzzer for audio feedback
// - Green button (index 2) is used to start the game

// ----- PIN CONFIGURATION -----
const int BUTTON_PINS[4] = {10, 9, 8,7};  // Button inputs (with internal
pullup)
const int LED_PINS[4] = {3, 4, 5, 6};      // LED outputs for each color
const int BUZZER_PIN = 11;                 // Buzzer output

// ----- AUDIO TONES -----
const int TONES[4] = {310, 209, 415, 252}; // Unique tone for each color

// ----- GAME VARIABLES -----
const int MAX_SEQUENCE = 100;              // Maximum sequence length
int sequence[MAX_SEQUENCE];                // Stores the color sequence
int sequenceLength = 0;                    // Current sequence length
int userInputIndex = 0;                    // Tracks user's position in sequence

// ----- GAME STATES -----
enum GameState {
  WELCOME,        // Show startup animation
  WAIT_FOR_START, // Wait for player to press start button
  PLAY_SEQUENCE,  // Display the sequence to memorize
  USER_INPUT,     // Wait for player to repeat the sequence
  GAME_OVER       // Show game over animation
};
GameState gameState = WELCOME;

// ----- DEBOUNCE TIMING -----
unsigned long lastButtonPress = 0;         // Last button press timestamp
const unsigned long debounceDelay = 200;   // Minimum time between button
presses (ms)

// ======================================
// SETUP
// ======================================
void setup() {
  // Configure button pins with internal pullup resistors
  for (int i = 0; i < 4; i++) {
    pinMode(BUTTON_PINS[i], INPUT_PULLUP);
    pinMode(LED_PINS[i], OUTPUT);
  }

  pinMode(BUZZER_PIN, OUTPUT);

}

// ======================================
// MAIN LOOP
// ======================================
void loop() {
  switch (gameState) {
    case WELCOME:
      welcomeSequence();
      gameState = WAIT_FOR_START;
      break;
```

```
    case WAIT_FOR_START:
      if (waitForStartButton()) {
        sequenceLength = 0;
        digitalWrite(LED_PINS[2], LOW);  // Turn off blinking LED
        delay(500);  // Short pause after start button press
        gameState = PLAY_SEQUENCE;
      }
      break;

    case PLAY_SEQUENCE:
      addToSequence();
      playSequence();
      userInputIndex = 0;
      lastButtonPress = millis();  // Reset debounce timer
      gameState = USER_INPUT;
      break;

    case USER_INPUT:
      {
        int input = readButton();
        if (input != -1) {
          playColor(input);

          // Check if input matches the sequence
          if (input != sequence[userInputIndex]) {
            gameState = GAME_OVER;
          } else {
            userInputIndex++;
            // Check if entire sequence was entered correctly
            if (userInputIndex >= sequenceLength) {
              delay(500);
              gameState = PLAY_SEQUENCE;
            }
          }
        }
      }
      break;

    case GAME_OVER:
      gameOver();
      gameState = WAIT_FOR_START;
      break;
  }
}

// ======================================
// ADD NEW COLOR TO SEQUENCE
// ======================================
// Adds a random color to the sequence.
// Ensures first color is never green (to avoid confusion with start button).
void addToSequence() {
  if (sequenceLength < MAX_SEQUENCE) {
    int newColor;
    do {
      newColor = random(0, 4);
    } while (sequenceLength == 0 && newColor == 2);  // First color cannot be
green

    sequence[sequenceLength++] = newColor;
  }
}

// ======================================
// PLAY ENTIRE SEQUENCE
```

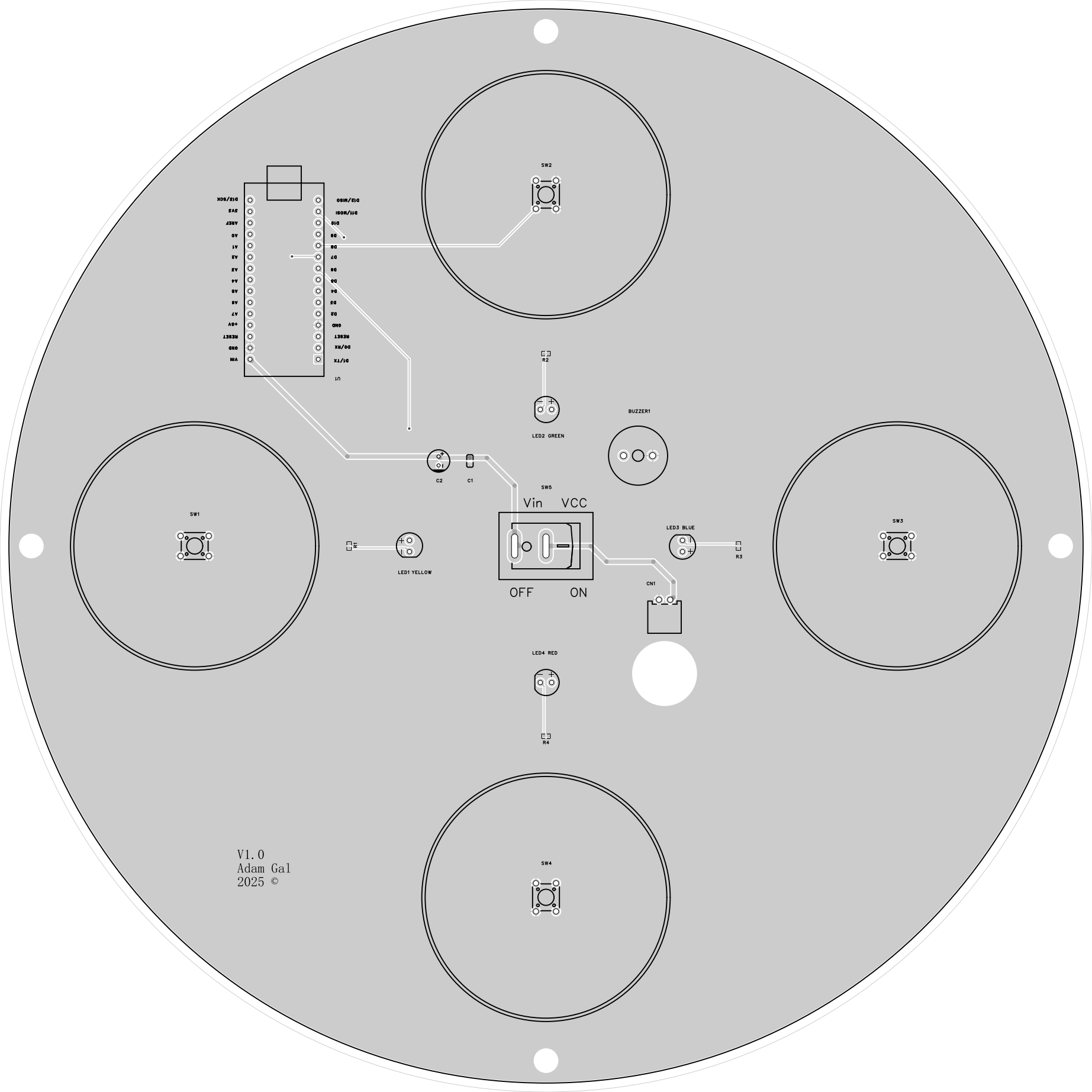
```
// ========================================
// Plays back the current sequence for the player to memorize.
void playSequence() {
  for (int i = 0; i < sequenceLength; i++) {
    playColor(sequence[i]);
    delay(300);  // Pause between colors
  }
}

// ========================================
// PLAY SINGLE COLOR
// ========================================
// Lights up LED and plays tone for a single color.
void playColor(int index) {
  int duration = 150;

  digitalWrite(LED_PINS[index], HIGH);
  tone(BUZZER_PIN, TONES[index], duration);
  delay(duration);
  digitalWrite(LED_PINS[index], LOW);
}

// ========================================
// READ BUTTON INPUT
// ========================================
// Non-blocking button read with debounce.
// Returns button index (0-3) or -1 if no button pressed.
int readButton() {
  // Debounce: ignore inputs that are too close together
  if (millis() - lastButtonPress < debounceDelay) {
    return -1;
  }

  // Check each button
  for (int i = 0; i < 4; i++) {
    if (digitalRead(BUTTON_PINS[i]) == LOW) {
      lastButtonPress = millis();

      // Wait for button release
      while (digitalRead(BUTTON_PINS[i]) == LOW) {
        delay(10);
      }

      return i;
    }
  }
  return -1;
}

// ========================================
// GAME OVER ANIMATION
// ========================================
// Flashes all LEDs and plays error sound.
void gameOver() {
  delay(300);  // Wait for button release

  // Flash all LEDs 3 times with low buzzer tone
  for (int i = 0; i < 3; i++) {
    tone(BUZZER_PIN, 100, 200);
    for (int j = 0; j < 4; j++) {
      digitalWrite(LED_PINS[j], HIGH);
    }
    delay(200);
    noTone(BUZZER_PIN);
```

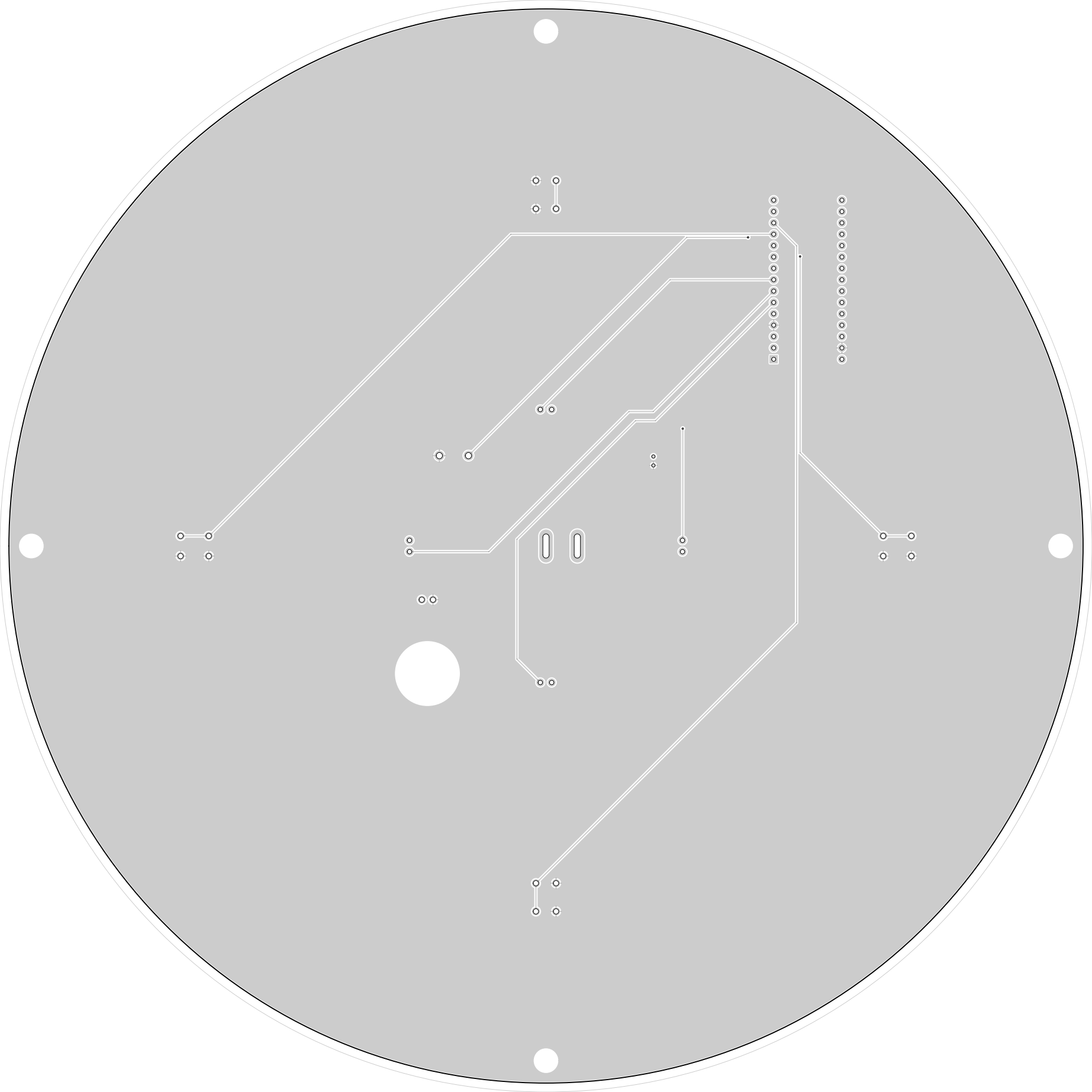
```
    for (int j = 0; j < 4; j++) {
      digitalWrite(LED_PINS[j], LOW);
    }
    delay(200);
  }

  delay(1000);  // Pause before allowing restart
}

// ========================================
// WELCOME ANIMATION
// ========================================
// Plays startup sequence: cascade effect followed by synchronized flashes.
void welcomeSequence() {
  int duration = 150;

  // Cascade: each color lights up in sequence (2 times)
  for (int repeat = 0; repeat < 2; repeat++) {
    for (int i = 0; i < 4; i++) {
      digitalWrite(LED_PINS[i], HIGH);
      tone(BUZZER_PIN, TONES[i], duration);
      delay(duration);
      digitalWrite(LED_PINS[i], LOW);
      noTone(BUZZER_PIN);
      delay(100);
    }
  }

  // Synchronized flash: all colors at once (4 times)
  for (int repeat = 0; repeat < 4; repeat++) {
    for (int i = 0; i < 4; i++) {
      digitalWrite(LED_PINS[i], HIGH);
    }
    tone(BUZZER_PIN, 400, duration);
    delay(duration);
    noTone(BUZZER_PIN);
    for (int i = 0; i < 4; i++) {
      digitalWrite(LED_PINS[i], LOW);
    }
    delay(150);
  }
}

// ========================================
// WAIT FOR START BUTTON
// ========================================
// Non-blocking function that blinks green LED while waiting for start button.
// Returns true when start button (green, index 2) is pressed.
bool waitForStartButton() {
  static unsigned long previousMillis = 0;
  const long interval = 300;  // Blink interval (ms)
  static bool ledState = false;

  unsigned long currentMillis = millis();

  // Toggle LED at regular intervals
  if (currentMillis - previousMillis >= interval) {
    previousMillis = currentMillis;
    ledState = !ledState;
    digitalWrite(LED_PINS[2], ledState ? HIGH : LOW);
  }

  // Check if start button is pressed
  if (digitalRead(BUTTON_PINS[2]) == LOW) {
```

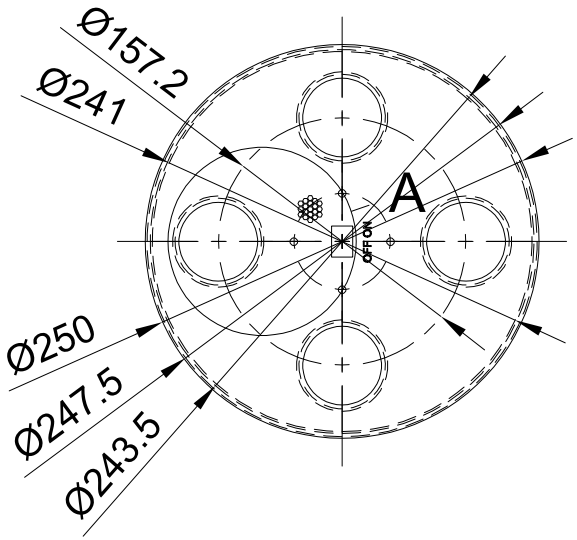
```
    while (digitalRead(BUTTON_PINS[2]) == LOW) {
      delay(10);  // Wait for button release
    }
    return true;
  }

  return false;
}
```



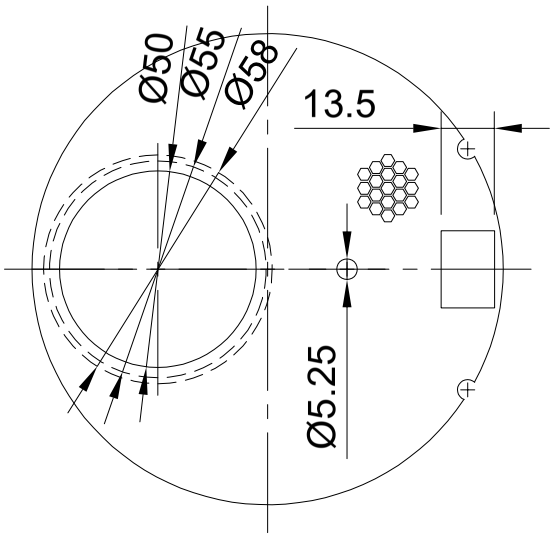
TITLE: SimonGameNano_2025-10-08_15-37-00		REV: 1.0
EasyEDA	Company: Adam Gal	Sheet: 1/1
	Date: 2025-10-05 Drawn By: Adam Gal	

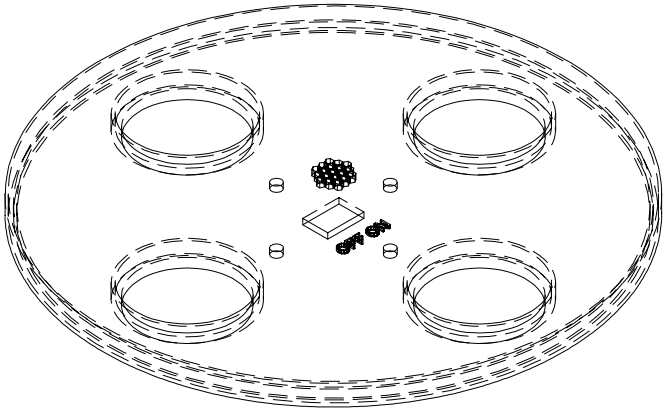

SW2
U1
R2
LED2 GREEN
BUZZER1
C2
C1
SW1
SW5
Vin VCC
SW3
LED3 BLUE
R3
R1
LED1 YELLOW
CN1
OFF ON
LED4 RED
R4
SW4
V1.0
Adam Gal
2025 ©

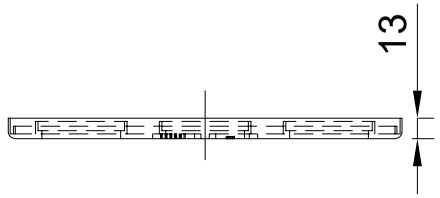


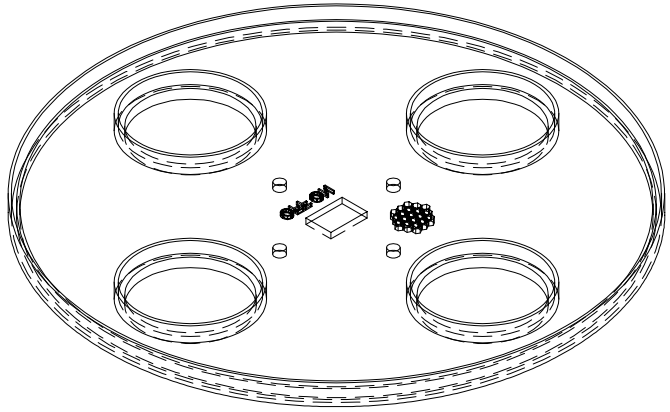


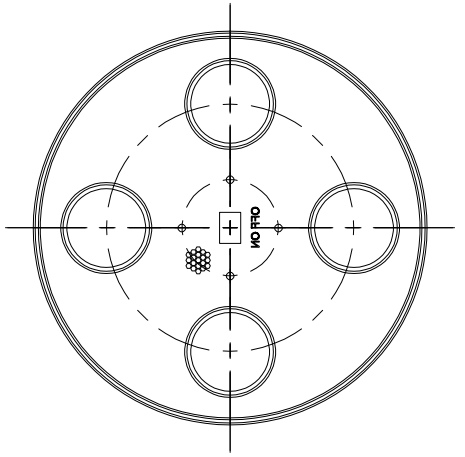
A (1:2)





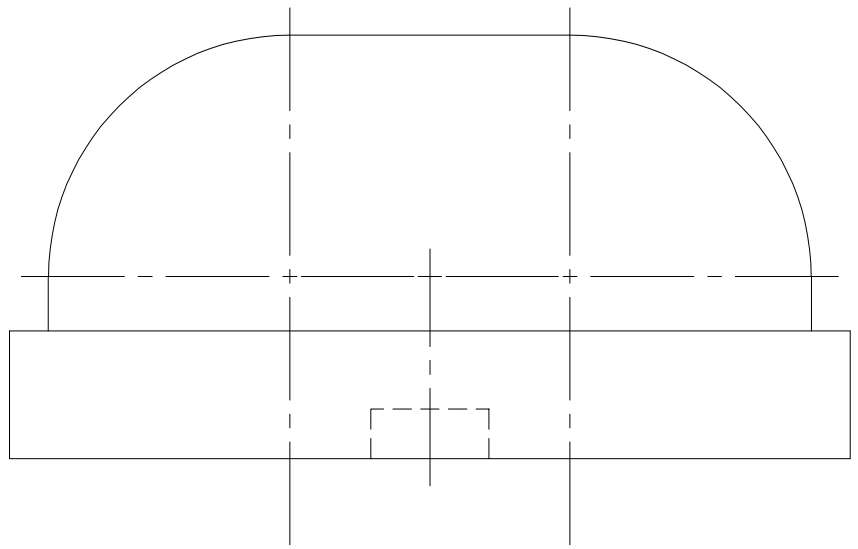


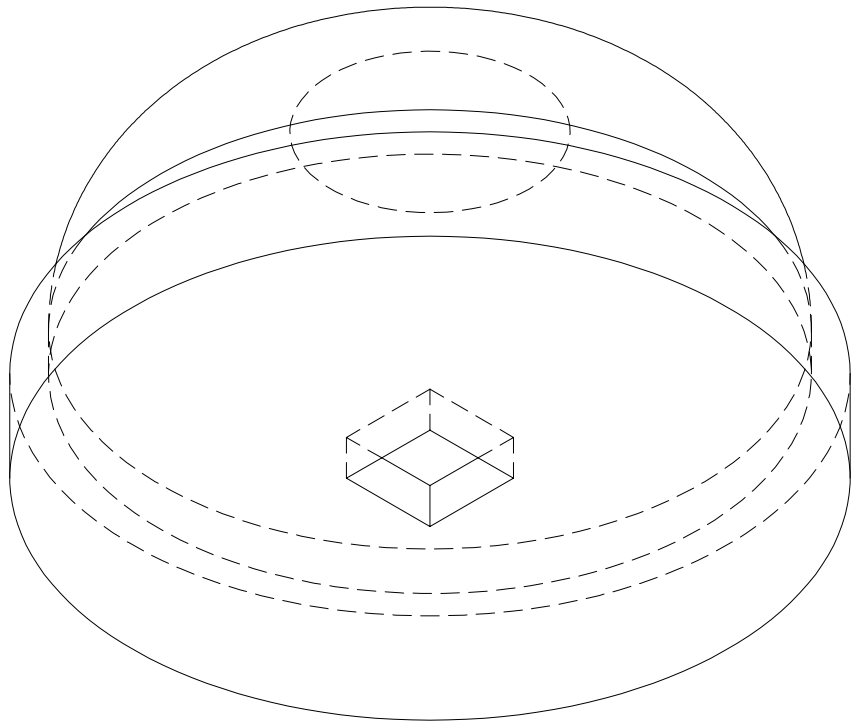


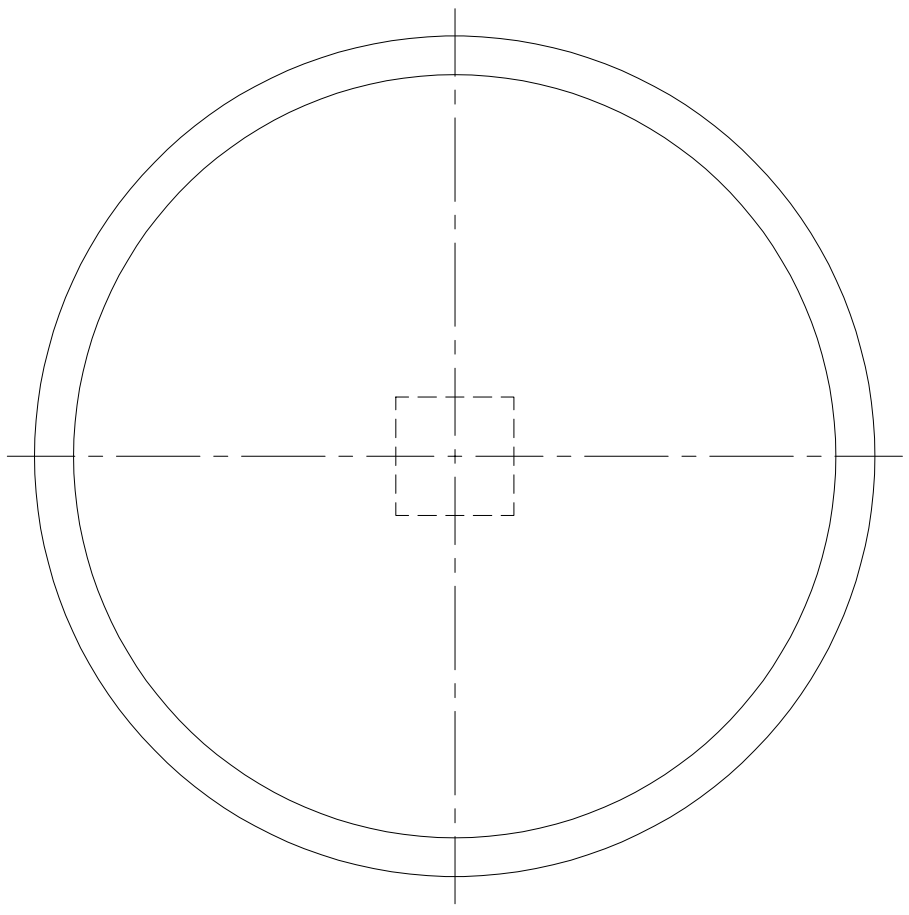


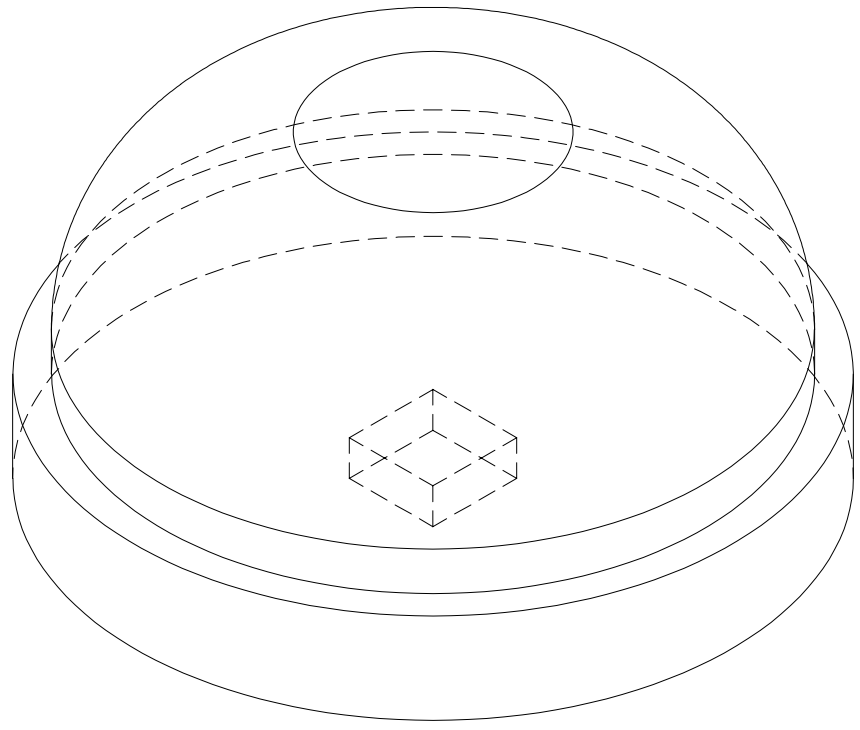
Dept.	Technical reference	Created by	Approved by
-		Adam Gal 27/11/2025	-
		Document type	Document status
			-
		Title	DWG No.
		Top_Plastic	20

Rev.	Date of issue	Sheet
1.0		1/1

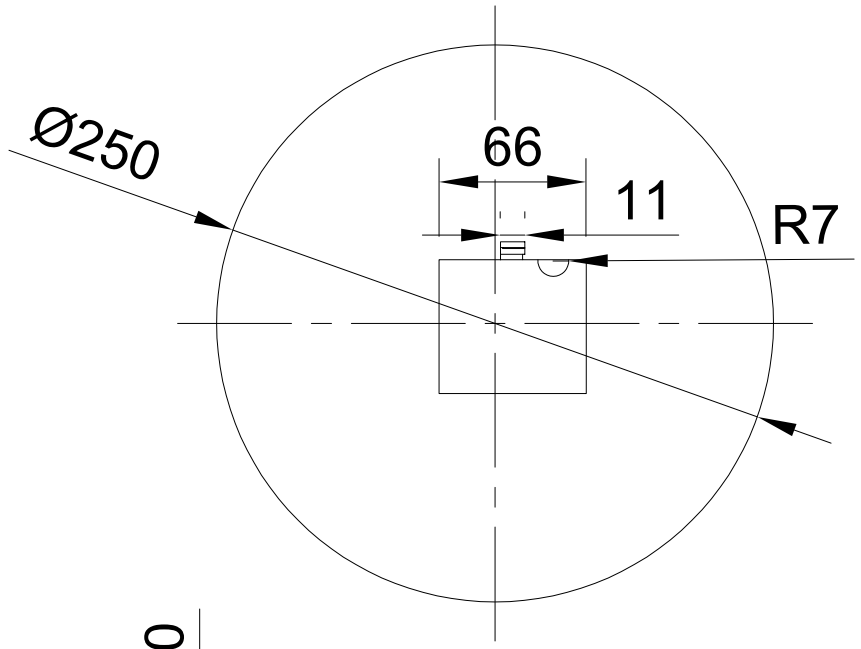


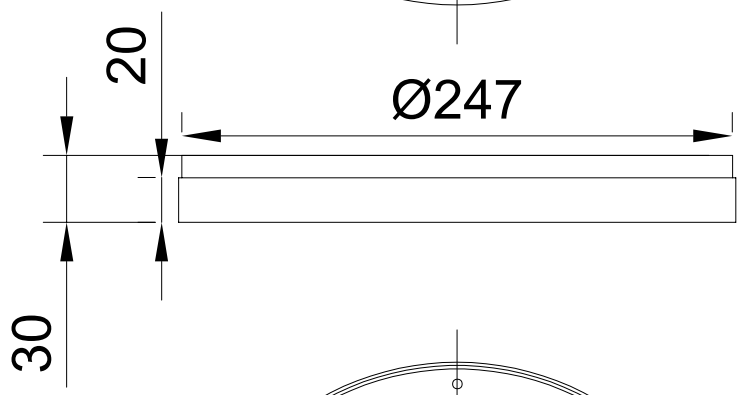


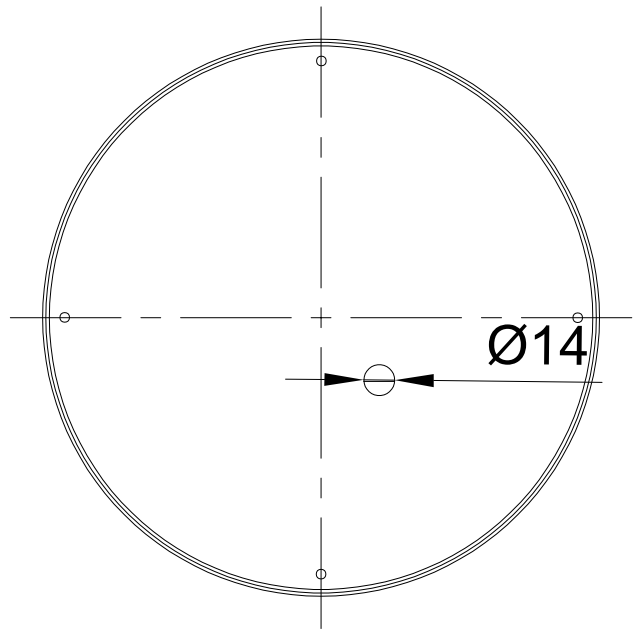


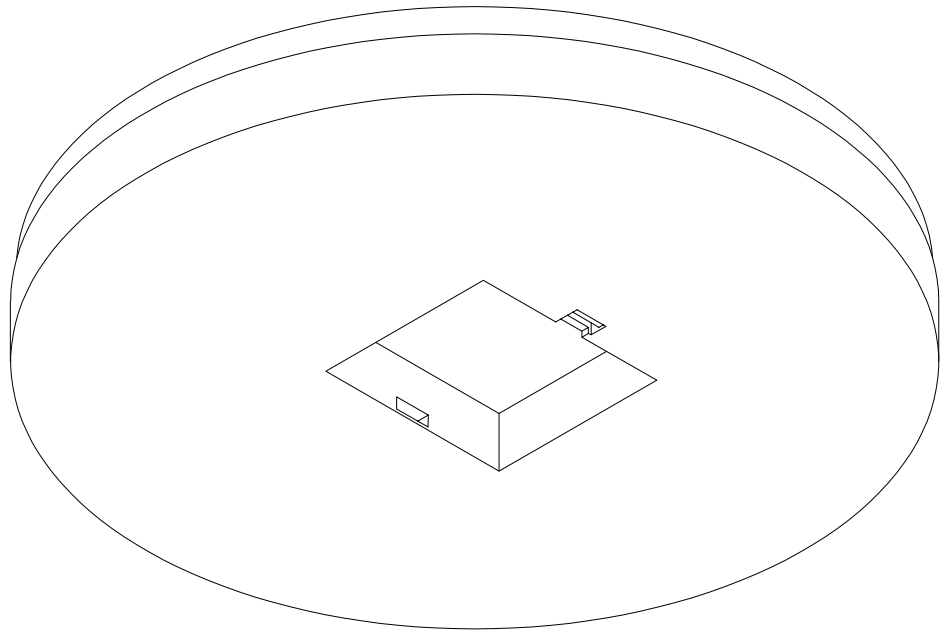


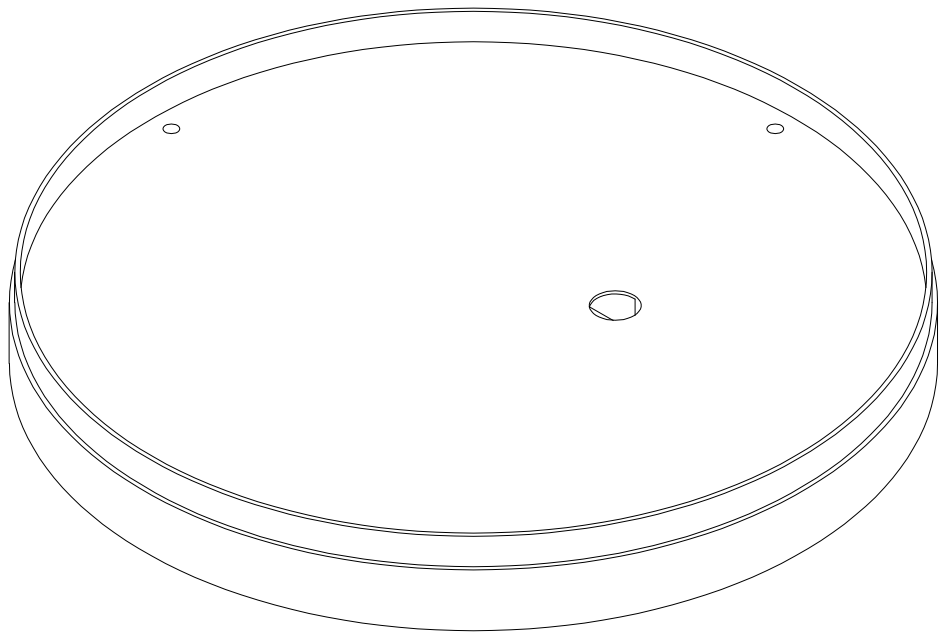
Dept.	Technical reference	Created by	Approved by
		Adam Gal 16/12/2025	
		Document type	Document status
		Title: Button	DWG No.: 40
			Rev.: 1.0 / Date of issue: 2025.12.16 / Sheet: 1/1



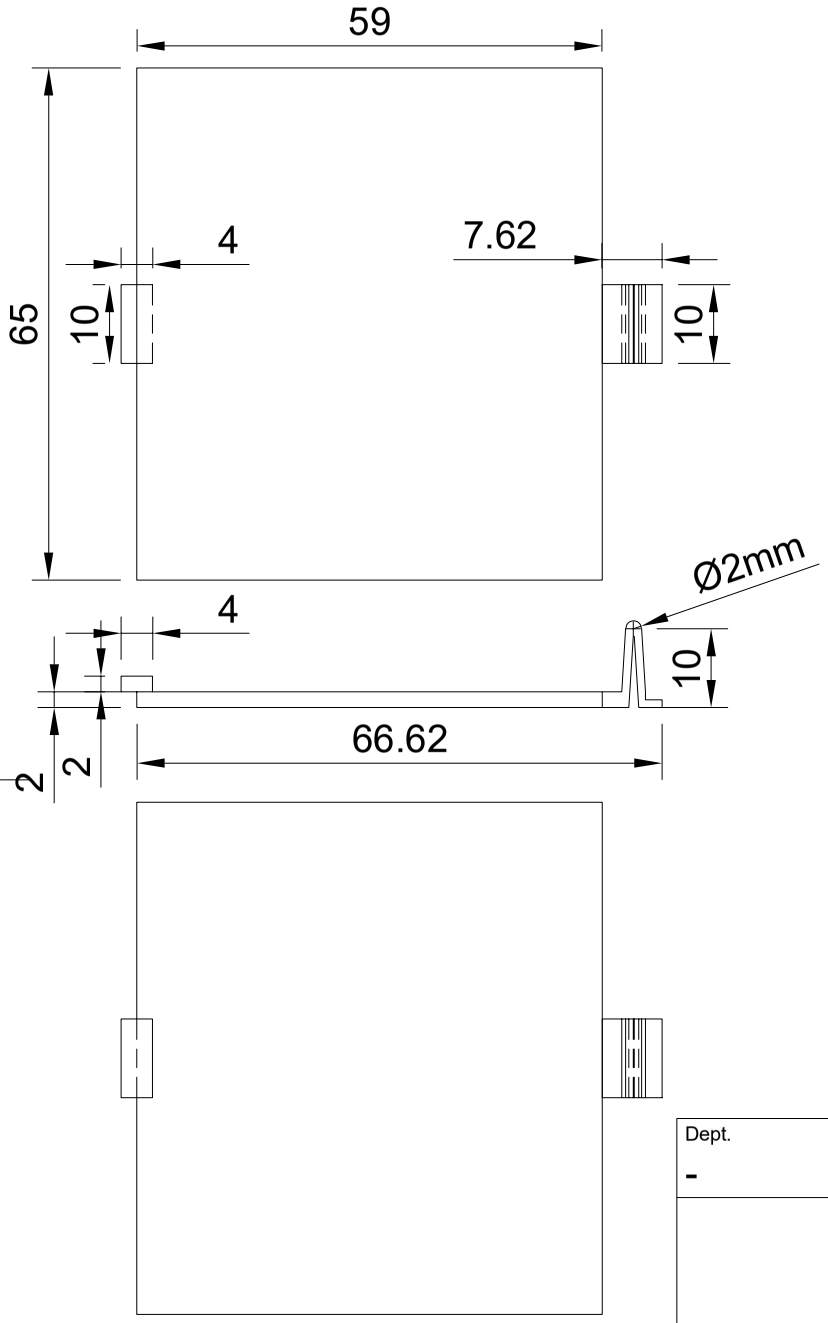


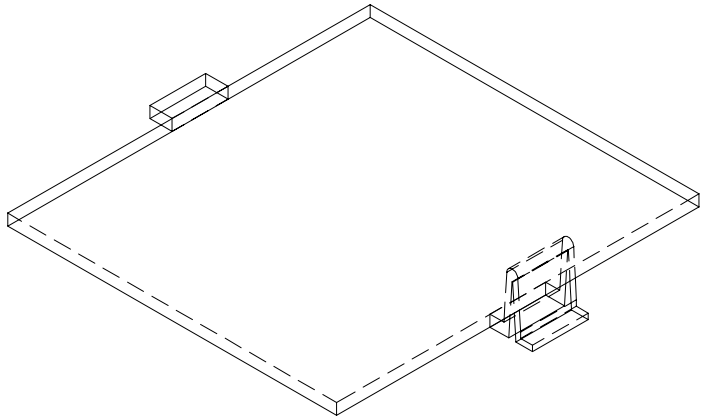


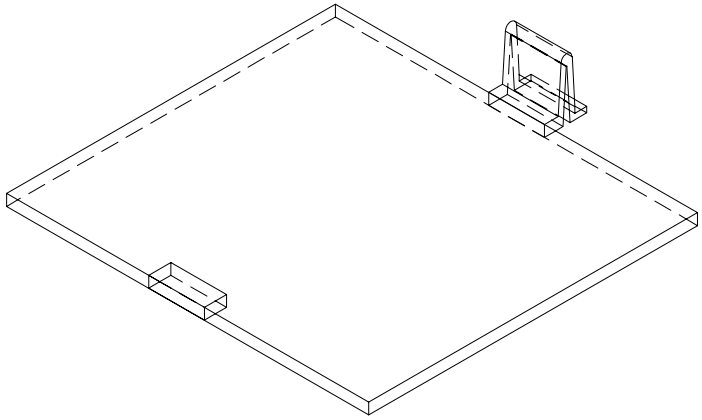




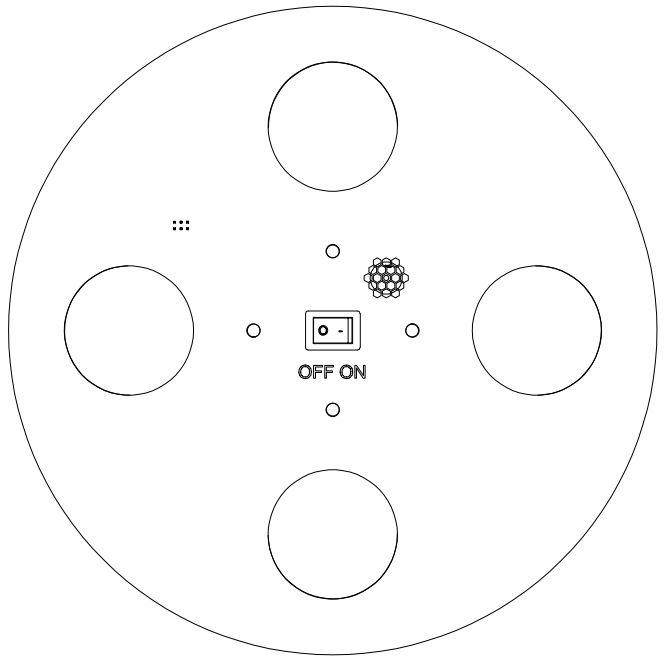
Dept.	Technical reference	Created by	Approved by
-		Adam Gal 27/11/2025	-
		Document type	Document status: -
		Title: Bottom_Plastic	DWG No.: 10
			Rev.: 1.0 / Date of issue: 2025.11.27 / Sheet: 1/1

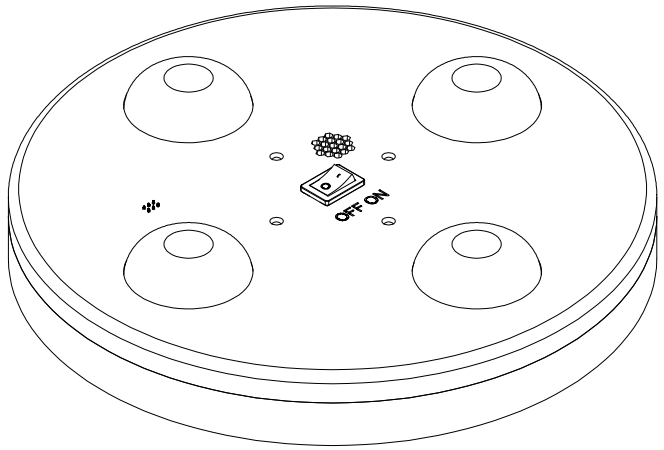


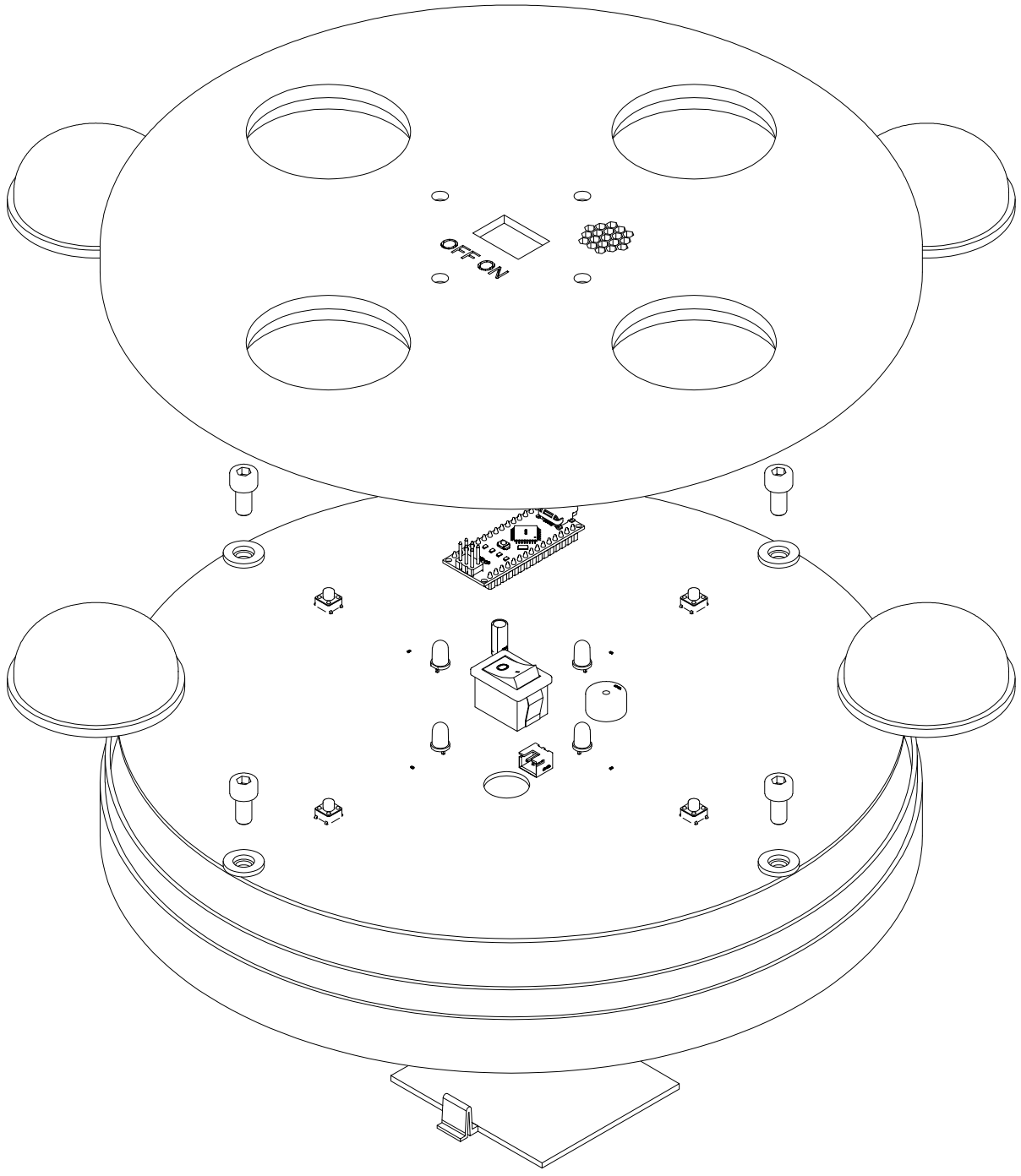


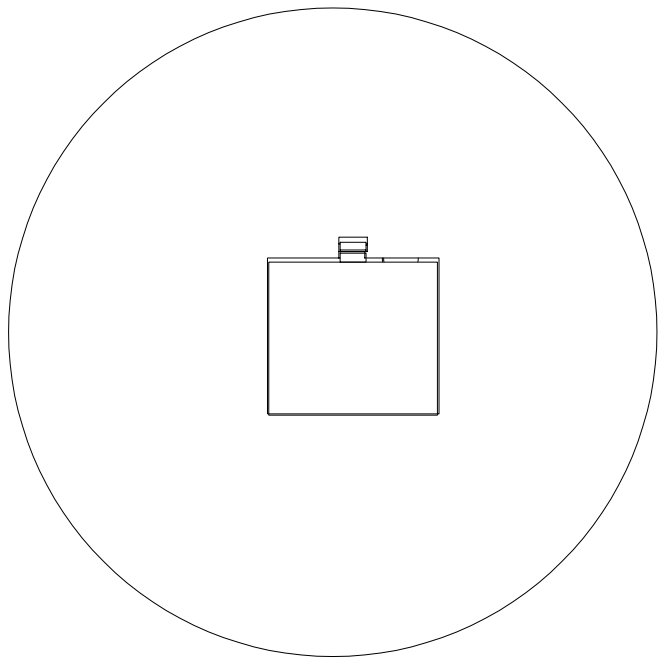


Dept.	Technical reference	Created by	Approved by
-		Adam Gal 27/11/2025	-
		Document type	Document status
			-
		Title	DWG No.
		Latch_Plastic	30
			Rev. 1.0 / Date of issue 2025.11.27 / Sheet 1/1







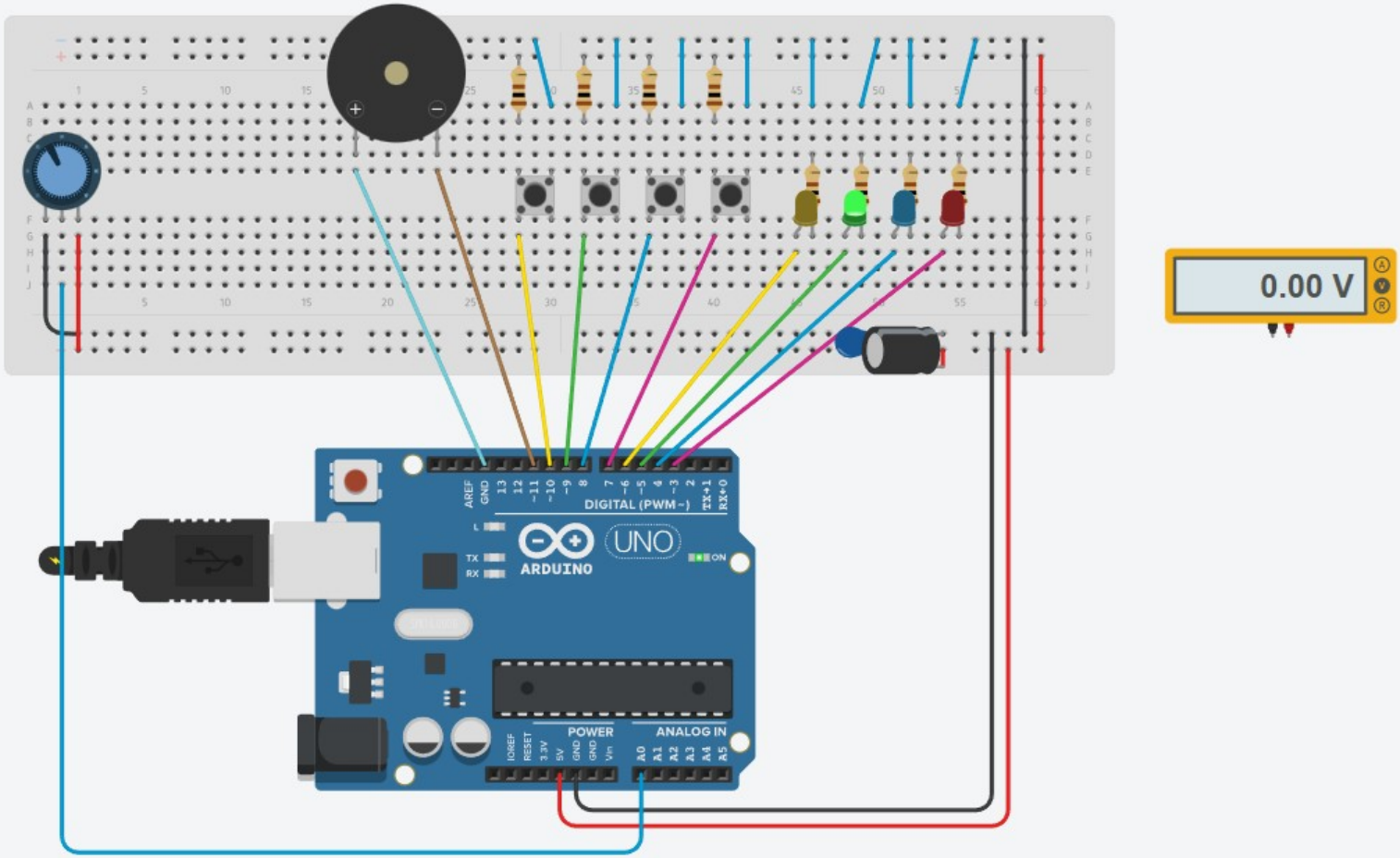


Dept.	Technical reference	Created by	Approved by
-		Adam Gal 27/11/2025	-
		Document type	Document status
			-
		Title	DWG No.
		Assembly	50
		Rev. 1.0	Date of issue 2025.11.27 / Sheet 1/1

All changes saved

Simulator time: 00:00:13

Code




```
1  //DATE: 2025.10.08
2  //Author: Gal Adam
3  // ====================================
4  // SIMON GAME - Arduino Nano
5  // ====================================
6  // A memory game where players repeat an
7  // - 4 colored buttons with LEDs
8  // - Buzzer for audio feedback
9  // - Green button (index 2) is used to st
10
11 // ----- PIN CONFIGURATION -----
12 const int BUTTON_PINS[4] = {10, 9, 8,7};
13 const int LED_PINS[4] = {3, 4, 5, 6};
14 const int BUZZER_PIN = 11;
15
16 // ----- AUDIO TONES -----
17 const int TONES[4] = {310, 209, 415, 252}
18
19 // ----- GAME VARIABLES -----
20 const int MAX_SEQUENCE = 100;
21 int sequence[MAX_SEQUENCE];
22 int sequenceLength = 0;
23 int userInputIndex = 0;
24
25 // ----- GAME STATES -----
26 enum GameState {
27   WELCOME,         // Show startup animati
28   WAIT_FOR_START, // Wait for player to p
29   PLAY_SEQUENCE,  // Display the sequence
30   USER_INPUT,      // Wait for player to r
31   GAME_OVER        // Show game over anima
32 };
33 GameState gameState = WELCOME;
34
35 // ----- DEBOUNCE TIMING -----
36 unsigned long lastButtonPress = 0;
37 const unsigned long debounceDelay = 200;
38
39 // ====================================
40 // SETUP
41 // ====================================
42 void setup() {
43   // Configure button pins with internal
44   for (int i = 0; i < 4; i++) {
45     pinMode(BUTTON_PINS[i], INPUT_PULLUP)
46
```

Serial Monitor


V1.0
Adam Gal
2025 ©
Vin
VCC