PROJEKT: SIMULERING REACT-GAME

Kurs: Test, verifiering och certifiering

Johan Kämpe

2017-12-10

Mölk Utbildning
Mjukvaruutvecklare inbyggda system

Lärare: Tomas Berggren

Sammanfattning

Syftet med projektet är att få en ökad förståelse för hur simulering av hårdvara fungerar och hur det kan utföras. Målet är att kunna simulera ett hårdvaruprojekt i ett eller flera simuleringsprogram.

Simulering utfördes av ett redan färdigt projekt kallat *reactionGameRGBLed,* ett reaktionsspel för två personer. Simulering av spelet utfördes i två olika simuleringsprogram, TinkerCad och UnoArduSim.

Originalkällkoden för *reactionGameRGBLed* skrevs om i sin helhet för att kunna simuleras. Till skillnad från originalet användes en LCD-display för simulering i TinkerCad, då TinkerCad ej har stöd för den display som användes i spelet. Det valdes att också använda interrupts för detektion knapptryckningar.

För simulering i UnoArduSim användes serial-kommunikation i stället för en display.

Demonstrationsvideos spelades in av simuleringarna.

Innehållsförteckning

1 Inledning

Detta kapitel beskriver syftet med projektet, dess krav, och en beskrivning av det tidigare projekt som valdes att simuleras.

1.1 Syfte och mål

Syftet med projektet är att få en ökad förståelse för hur simulering av hårdvara fungerar och hur det kan utföras.

Målet är att utföra en simulering av ett tidigare färdigt hårdvaruprojekt i ett eller flera simuleringsverktyg.

1.2 Projektkrav

För betyget godkänt ska simulering av det tidigare projektet utföras i *minst* ett simuleringsverktyg, om simulering misslyckas ska utmaningar som uppstod beskrivas.

Projektet ska levereras med en **projektrapport** med innehåll:

- **Inledning**
- **Syfte**
- **Mål**
- Beskrivning av **metod**
- **Resultat**
- **Diskussionsdel**

Rapporten ska innehålla **bilagor**:

- **Kopplingsschema**
- **Källkod**
- **Demonstration av simuleringen som utfördes**, eventuellt en länk till en video-fil

Projektet bedöms utifrån dess kvalitet och svårighetsgrad.

Om fler än ett projekt väljs att simuleras, ska dessa ha egna rapporter.

1.3 Val av simuleringsverktyg

De verktyg som valdes att användas i simuleringsprojektet är **TinkerCad,** som är ett webbaserat simuleringsverktyg från bolaget Autodesk, samt datorprogrammet **UnoArduSim**, utvecklat av Stan Simmons.

TinkerCad valdes som huvudprogram för simulering, och beskrivs utförligare i rapporten.

1.4 Bakgrund

1.4.1 Beskrivning av tidigare projekt

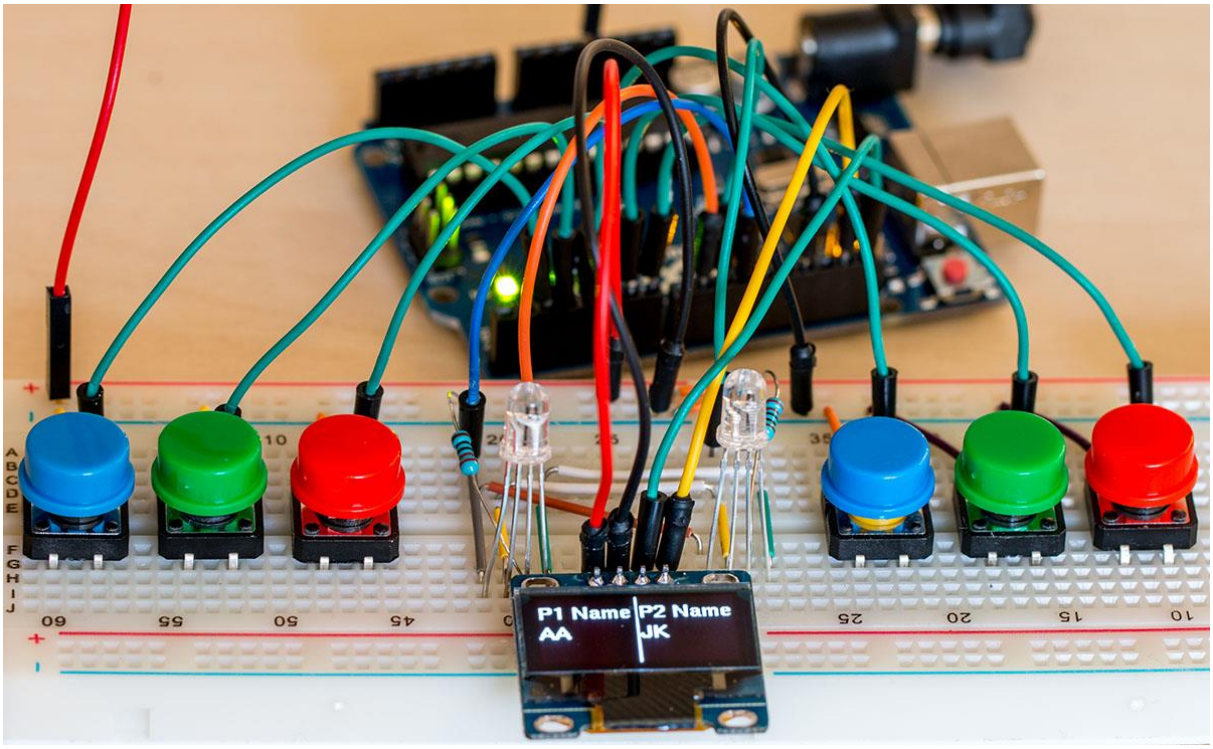
Det befintliga projekt som valdes för simulering kallas *reactionGameRGBLed*, och är ett reaktionsspel för två personer.

Projektet skapades 2016-11-25 av *Mjukvaruutvecklare Inbyggda System*-eleven Johan Kämpe, för att öva på programmering och utveckling i Arduino-miljön. Projektet var ett hobbyprojekt och användes inte i utbildningen.

Projektet bestod av följande komponenter:

Komponent	Antal
Tryckknapp röd	2 st.
Tryckknapp grön	2 st.
Tryckknapp blå	2 st.
Lysdiod RGB	2 st.
OLED display SSD1306	1 st.
Resistor 330Ω	2 st.
Breadboard	1 st.
Arduino UNO-utvecklingskort	1 st.

Även flertalet kablar och bygelkablar användes för att koppla samman komponenterna via breadboard-plattan.

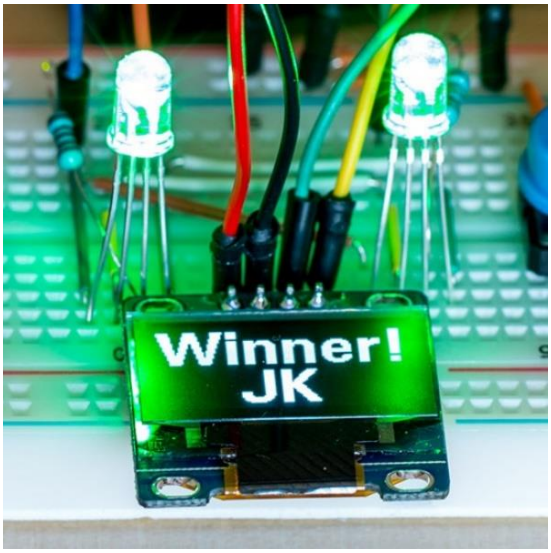


Figur 1 Fotografi på uppkopplat och startat reaktionsspel

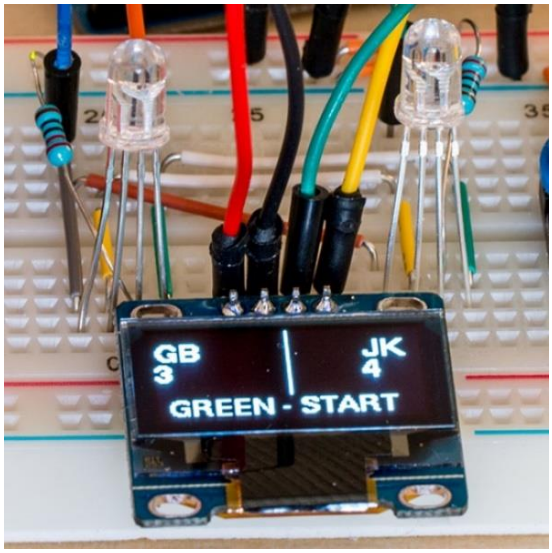
Regler för spelet:

- En av spelarna använder de vänstra tre knapparna, den andra spelaren de högra tre knapparna.
- En av spelarna trycker på sin gröna knapp för att starta en nedräkning från tre sekunder, som visas på displayen.
- När nedräkningen är klar tänds RGB-lysdioderna med en slumpvald färg, antingen röd, grön eller blå.
- Den spelare som först trycker på sin knapp med färg som matchar RGB-lysdioden vinner omgången.
- Om den spelare som trycker först har tryckt på fel färg, så går poängen till den andra spelaren.
- Resultat visas på displayen.

Vid uppstart av spelet ombeds också spelarna att mata in sina initialer med hjälp av knapparna. Blå och röd knapp väljer bokstav från alfabetet, och grön knapp bekräftar.



Figur 2 Displayinformation: vinnare



Figur 3 Displayinformation: poäng

En länk till en demonstrationsvideo av spelet finns i kapitlet Länkar.

1.5 Länkar

1.5.1 GitHub-länkar

Simuleringsprojektets GitHub-sida
https://github.com/GoblinDynamiteer/test-course-simulation-project

Orginalprojektets GitHub-sida
https://github.com/GoblinDynamiteer/arduino_misc/tree/master/reactionGameRGBLed

Kod för simulering i TinkerCad
https://github.com/GoblinDynamiteer/test-course-simulation-project/blob/master/react-game/code/refactored/react-game-refactored/react-game-refactored.cpp

Kod för simulering i UnoArduSim
https://github.com/GoblinDynamiteer/test-course-simulation-project/blob/master/react-game/code/uno_ardu_sim/uno_ardu_sim.cpp

1.5.2 YouTube-länkar

Demovideo av det fysiska reaktionsspelet
https://youtu.be/9Vmtv2STFm0

Simulering av reaktionsspelet i TinkerCad
https://youtu.be/hol5LOlpUuk

Simulering av reaktionsspelet i UnoArduSim
https://youtu.be/1ZH_kPwZ1Qg

Simuleringstest av knappar i TinkerCad
https://youtu.be/liGEypqYlbg

Simuleringstest av LCD i TinkerCad
https://youtu.be/g8J78O9m7_M

Simuleringstest av RGB-LED i TinkerCad
https://youtu.be/PVE1l9xOwBE

Reaktionsspel WIP, test av LCD och ISR
https://youtu.be/Jycv4DsDiec

1.5.3 Andra länkar

Simuleringsprojektet på TinkerCad
https://www.tinkercad.com/things/axWFTDi7dn8-reaction-game

1.6 Noteringar

Projektet *reactionGameRGBLed* som används för simuleringen hänvisas till i rapporten som *reaction-game.*

Metod för hur simulering utförs i UnoArduSim beskrivs ej i rapporten.

Koden för simulering i UnoArduSim finns ej som bilaga i rapporten, se länk i GitHub-länkar för att visa koden på projektets GitHub-sida.

2 Genomförande och resultat

2.1 Använd programvara

Programvaror som har använts i projektet

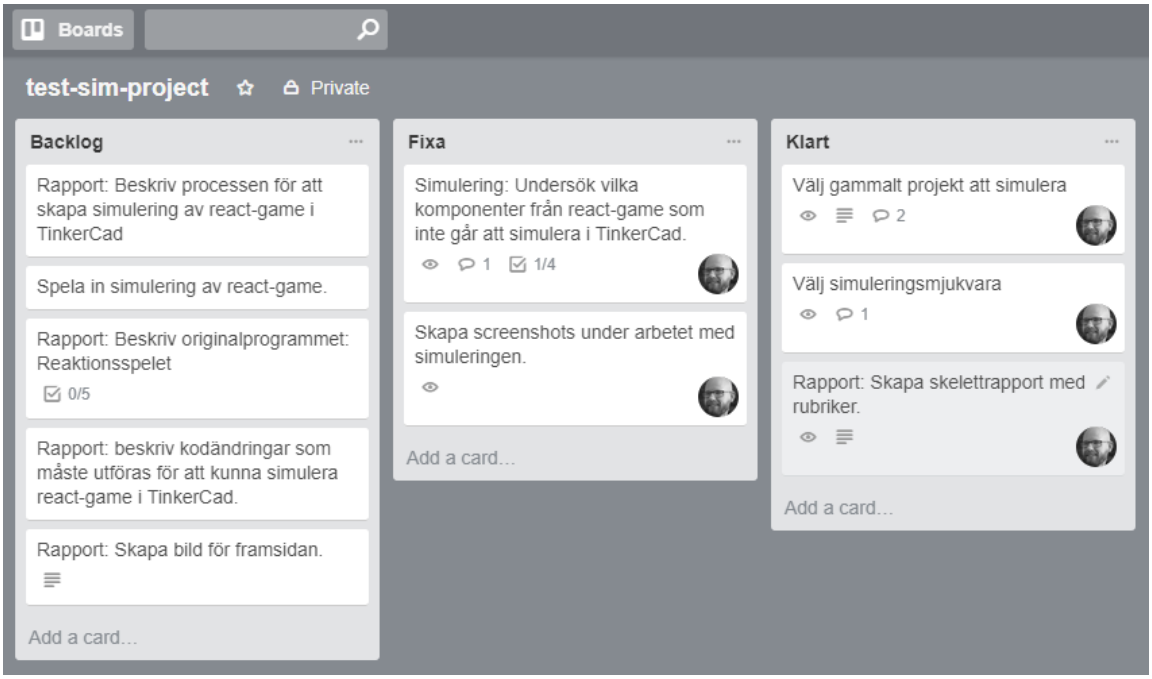
- **Microsoft Word 2016**
- **Autodesk TinkerCad**
  - Webbaserat simuleringsprogram med stöd för Arduino och vanliga komponenter.
- **UnoArduSim**
  - Simuleringsprogram för Arduino Uno, utvecklat av Stan Simmons
- **Atom text editor**
- **Git**
  - Versionshantering
- **Fritzing**
  - Program för att skapa kopplingsschema.
- **Icecream Screen Recorder**
  - Program för skärminspelning.
- **Trello**
  - Webbaserat planeringsverktyg.

2.2 Planering

Planering av projektet utfördes med planeringsverktyget *Trello*.

Uppgifter (kallade kort) att göra i projektet lades in till tavlan *Backlog* och flyttas sedan till *Fixa* när de påbörjas. När en uppgift är klar flyttas den till *Klart*.

Kort och kommentarer i Trello-projektet användes delvis som underlag till projektrapporten.

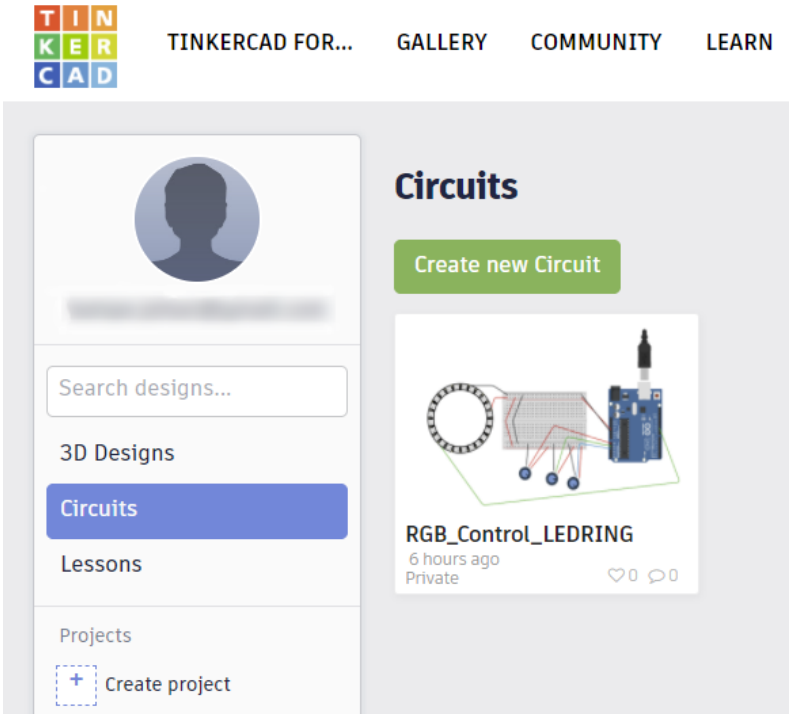


Figur 4 Planering med verktyget Trello

2.3 Metod

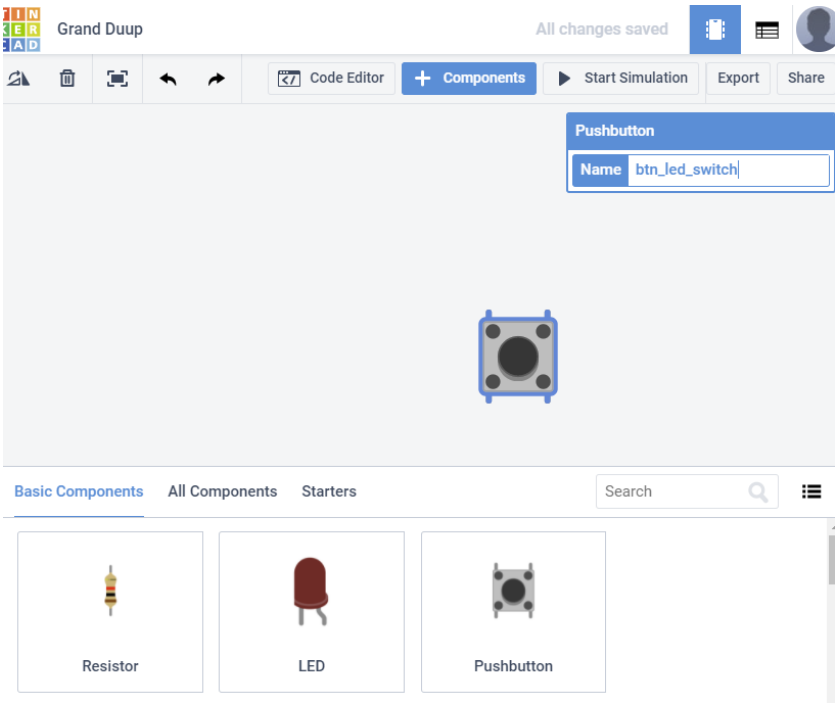
2.3.1 Skapande av simulationer med verktyget TinkerCad

För att skapa en ny kretssimulering i TinkerCad används undermenyn *Circuits* och sedan knappen *Create new circuit*, från TinkerCads huvudsida efter inloggning av användaren.



Figur 5 Undermenyn Circuits på TinkerCad

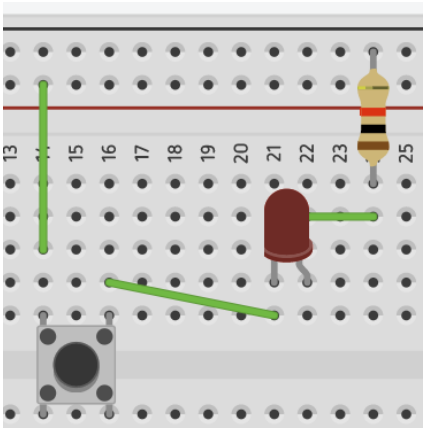
När ett nytt kretssimuleringsprojekt har skapats kan komponenter som ska användas läggas till med knappen *+ Components.* Komponenter visas i en lista och kan dras in till arbetsytan med muspekaren.



Figur 6 Tilläggning av tryckknapp i TinkerCad

Komponenter kan namnges, flyttas och raderas.

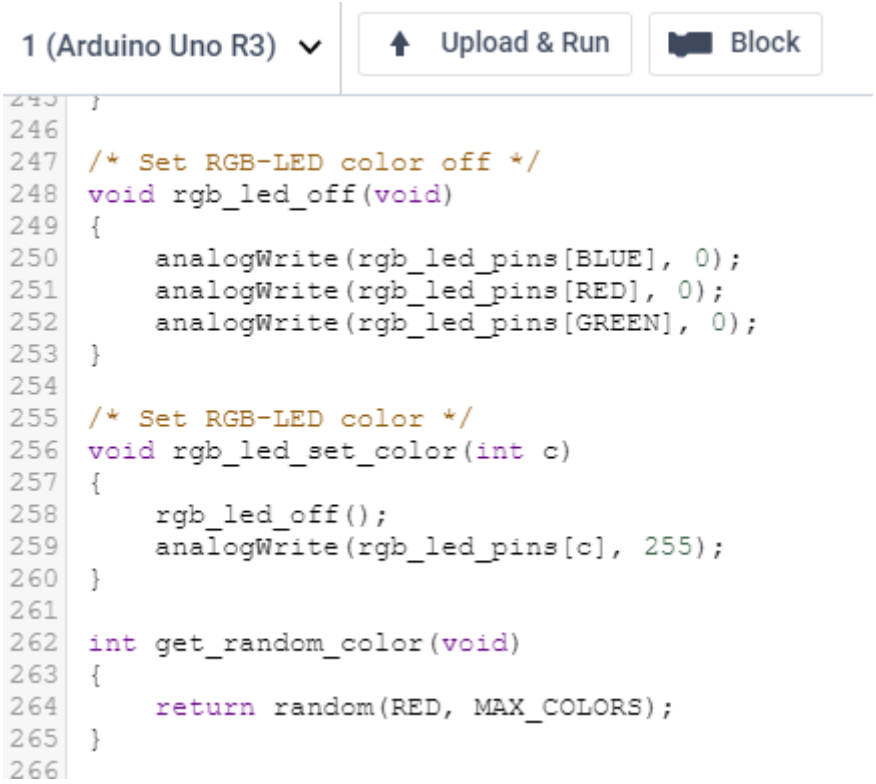
För att koppla samman komponenter med varandra används muspekaren på komponenternas anslutningar. En kabel skapas vid musklick.



Figur 7 Sammankopplade komponenter i TinkerCad, med gröna kablar

För att skriva eller klistra in kod till projektet används knappen *Code Editor*. För att starta simuleringen används knappen *Start Simulation* eller *Upload & Run.*

Figur 8 Knappen "Code Editor" på TinkerCad



Figur 9 Kod i kod-editor på TinkerCad

2.4 Test av komponenter från reaction-game i TinkerCad

Komponenter från *reaction-game* testades initialt en och en, för att fastställa vilka som skulle vara möjliga att simulera och om ursprungskällkoden behövde modifieras.

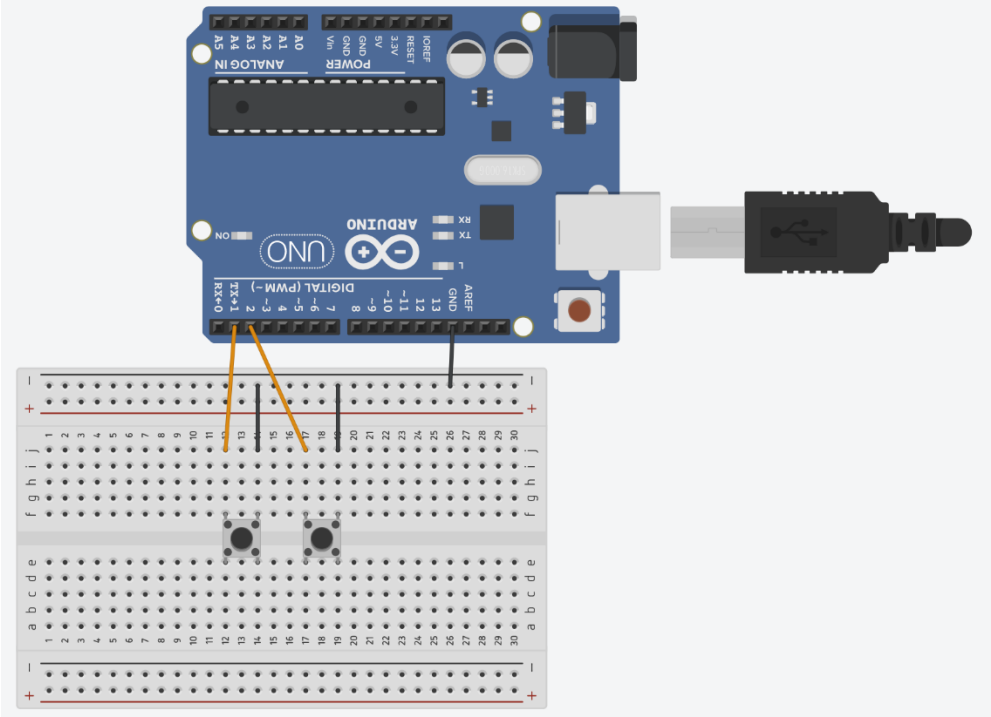
2.4.1 Test av knappar och Arduino UNO i TinkerCad

Reaction-game innehåller sex knappar och en Arduino UNO, för att säkerställa att dessa komponenter finns och fungerar i TinkerCad skapades en testkod, se Bilaga I: Testkod för knappar med bibliotek Button med TinkerCad.

I testkoden används två knappar, en för att tända lysdioden kopplad till pin 13 på en Arduino UNO, och en för att släcka den.

Likt orginalkoden för reaction-game användes biblioteket *Button* av Michael Adams för att skapa knappar.

Komponenter och kod lades in i ett nytt TinkerCad-projekt



Figur 10 TinkerCad-projekt med två knappar och en Arduino UNO

Vid simuleringsstart gavs följande fel:

fatal error: Button.h: No such file or directory

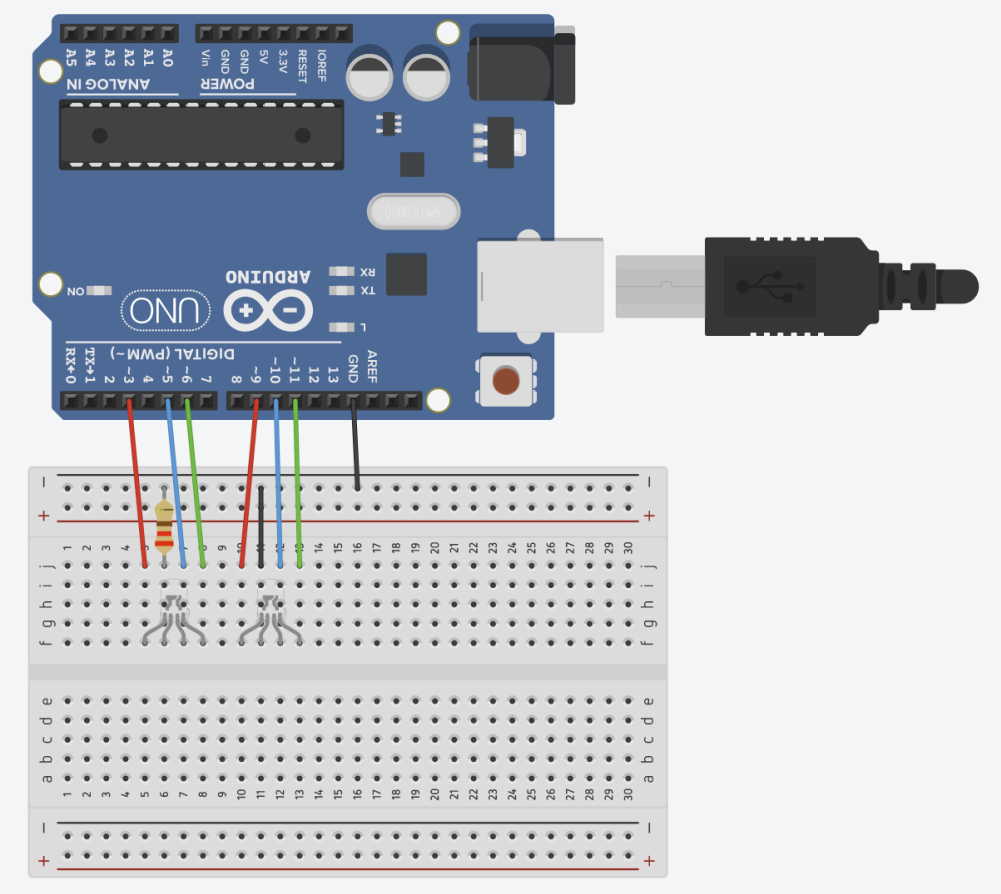
Således antogs det att biblioteket *Button* ej går att använda i TinkerCad. Testkoden byggdes om, se Bilaga II: Testkod för knappar utan bibliotek med TinkerCad, så att inget bibliotek för knappar används. Ingen hänsyn togs till *bouncing* i denna kod.

Med den modifierade koden startar simuleringen och knapparna fungerar som förväntat.

2.4.2 Test av RGB-Lysdioder och resistorer i TinkerCad

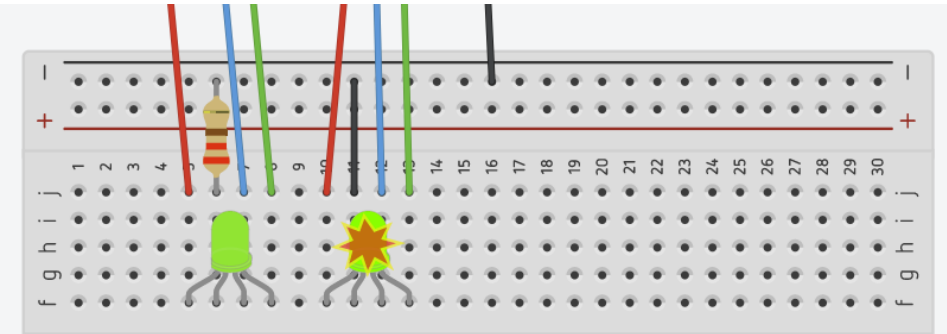
Reaction-game innehåller två RGB-lysdioder och två resistorer, för att säkerställa att dessa komponenter fungerar i TinkerCad skapades en testkod, se Bilaga III: Testkod för RGB-LED och resistor med TinkerCad.

Komponenter och kod lades in i ett nytt TinkerCad-projekt. Två RGB-lysdioder kopplades till en Arduino UNO, för den ena lysdioden kopplades en resistor med resistansen 220Ω mellan jord och lysdiodens gemensamma katod.



Figur 11 TinkerCad-projekt med två RGB-lysdioder, en resistor och en Arduino UNO

Simuleringen startar och lysdioderna cyklar mellan tre olika färger med olika styrka (PWM). Simuleringen indikerar att lysdioden utan resistor går sönder vid hög styrka.



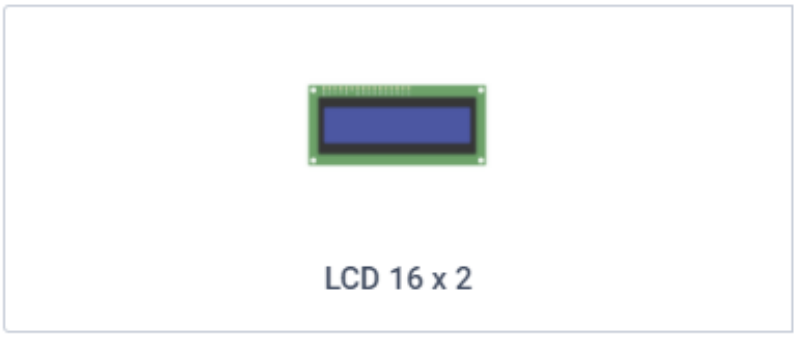
Figur 12 Indikation i TinkerCad, att en RGB-Lysdiod (den högra i bilden) har gått sönder

Detta visar att TinkerCad har möjlighet att simulera RGB-lysdioder och resistorer.

2.4.3 Test av display i TinkerCad

Reaction-game innehåller en SSD1306-display för att visa poängställning mellan spelare, nedräkning, samt inmatning av spelarnas initialer.

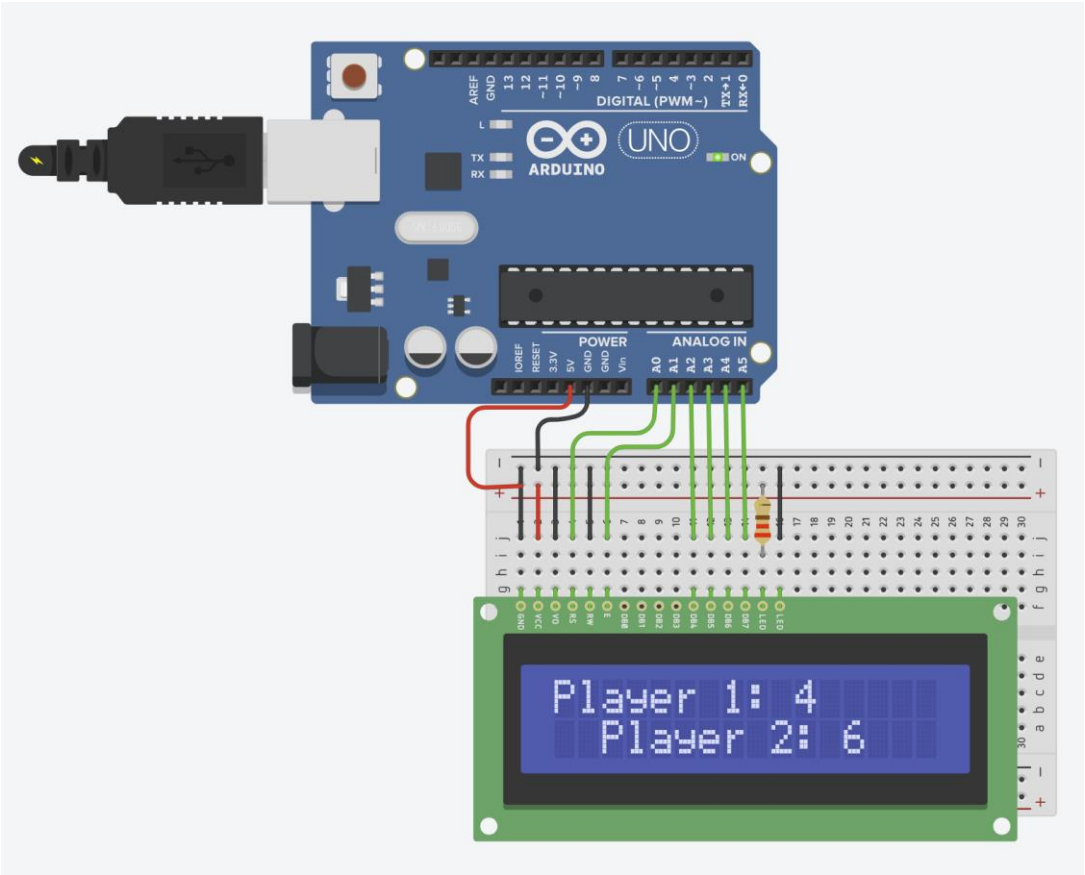
Vid inspektion av TinkerCads komponentlista visar det sig att det inte finns stöd för SSD1306, dock finns en annan display, kallad *LCD 16 x 2*. Denna väljs som ett lämpligt alternativ till SSD1306-displayen.



Figur 13 LCD 16 x 2-display i TinkerCads komponentlista

Testkod skapades för denna display, se Bilaga IV: Testkod för LCD med TinkerCad.

Komponenter och kod lades in i ett nytt TinkerCad-projekt.



Figur 14 TinkerCad-projekt med LCD-display och en Arduino UNO

Simuleringen fungerade i TinkerCad, LCD-displayen kommer att ersätta SSD1306-displayen i simuleringen av reaction-game.

2.5 Skapande av kod och simuleringsprojekt i TinkerCad

Koden för reaction-game skapades för över ett år sedan, den skrivas om i sin helhet, förutom de ändringar som behövde göras för att få den simuleringsbar i TinkerCad.

Koden för simulering i TinkerCad återfinns i Bilaga VII: Kod för simulering i TinkerCad.
Koden i bilagan saknar kommentarer, för fullständig version se länk i GitHub-länkar.

En video av den utförda simuleringen i TinkerCad kan ses på YouTube, se länk i YouTube-länkar.
Videon återfinns också på projektets GitHub-sida.

Vissa ändringar har utförts i spelet som simuleras, exempelvis går det att trycka på valfri knapp för att starta nedräkningen, till skillnad från enbart de gröna knapparna i originalutförandet.

Skillnader mellan den nya koden, och reaction-games originalkod:

- **Indentering**
  - Annan placering av klammerparenteser.
  - Mellanslag används i stället för tabb-tecken.

- **Variabler**
  - Bättre namngivning av variabelnamn.
  - Arrayer används för variabler av samma typ, exempel:

```
#define pinLEDRed 9
#define pinLEDGreen 10
#define pinLEDBlue 11
```

Figur 15 Variabler i ursprungskod

```
enum{ RED, GREEN, BLUE, MAX_COLORS };
const int rgb_led_pins[] = { 9, 10, 11 };
```

Figur 16 Variabler i ny kod

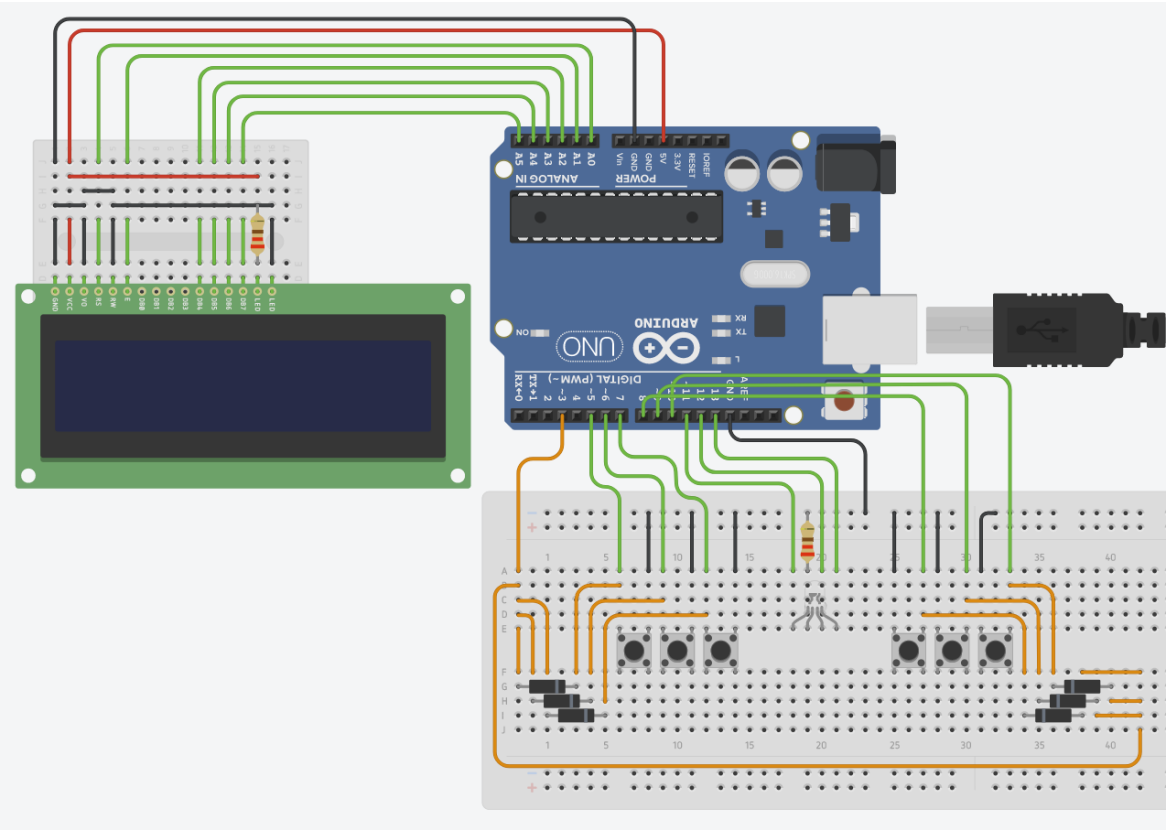
- **Bibliotek**
  - Egen funktionalitet för knapphantering används, i stället för att använda biblioteket *Button*.
  - *LiquidCrystal*-biblioteket används i stället för *U8glib*-biblioteket (för SSD1306-OLED).

- **Funktioner**
  - Bättre namngivning av funktioner.
  - Kommentarer som beskriver funktionerna.

- **Knappar**
  - I stället för att testa alla sex knappar kontinuerligt i programmets loop används en ISR som triggas av samtliga knappar. I ISR-funktionen kollas sedan vilken knapp som trycktes ned.

Under utvecklingen av den nya koden testades den kontinuerligt i TinkerCad, i ett projekt där den slutgiltiga hårdvaran hade byggts.

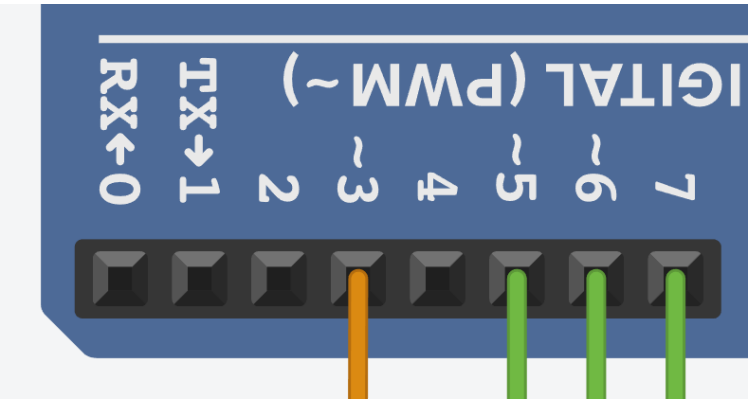
Kopplingsschema ses i Bilaga VI: Kopplingsschema.



Figur 17 Reaction-games hårdvara uppkopplad i TinkerCad

Då detektion av knapptryck valdes att hanteras av en ISR, kopplades även alla knappar till Arduino UNO-enhetens PIN3, via dioder. Dioderna förhindrar att en knapp triggar de andra knapparna som också är kopplade till PIN3.

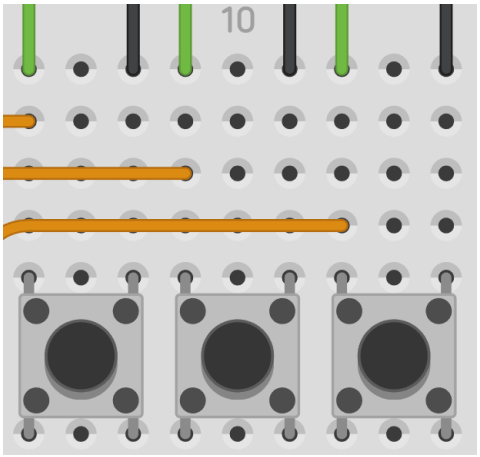
PIN3 används för trigger av ISR i koden.



Figur 18 Digitala anslutningar 0-7 på Arduino UNO i TinkerCad. PIN3 har en orange kabel ansluten till sig.

Alla anslutningar som är kopplade till knapparna, och interrupt-pin PIN3 är satta till läge INPUT_PULLUP. I detta läge används MCUns interna pull-up på anslutningarna. Anslutningarnas läge sätts med funktionen **pinMode.**

Knapparna är kopplade till jord, och jordar anslutningarna när de trycks ned.
ISR-funktionen är satt till att trigga när PIN3 jordas.



Figur 19 Anslutningar för knappar, orange kabel är kopplad till PIN3 för interrupt, svart kabel är jord

För att testa knapparnas läge, nedtryckt eller inte nedtryckt, används funktionen **digitalRead** på anslutningarna som knapparna är kopplade till. Om **digitalRead** ger värdet 0 är anslutningen jordad och knappen är nedtryckt.

```
pinMode(INT_PIN, INPUT_PULLUP);
attachInterrupt(
    digitalPinToInterrupt(INT_PIN), isr_button_click, FALLING);
```

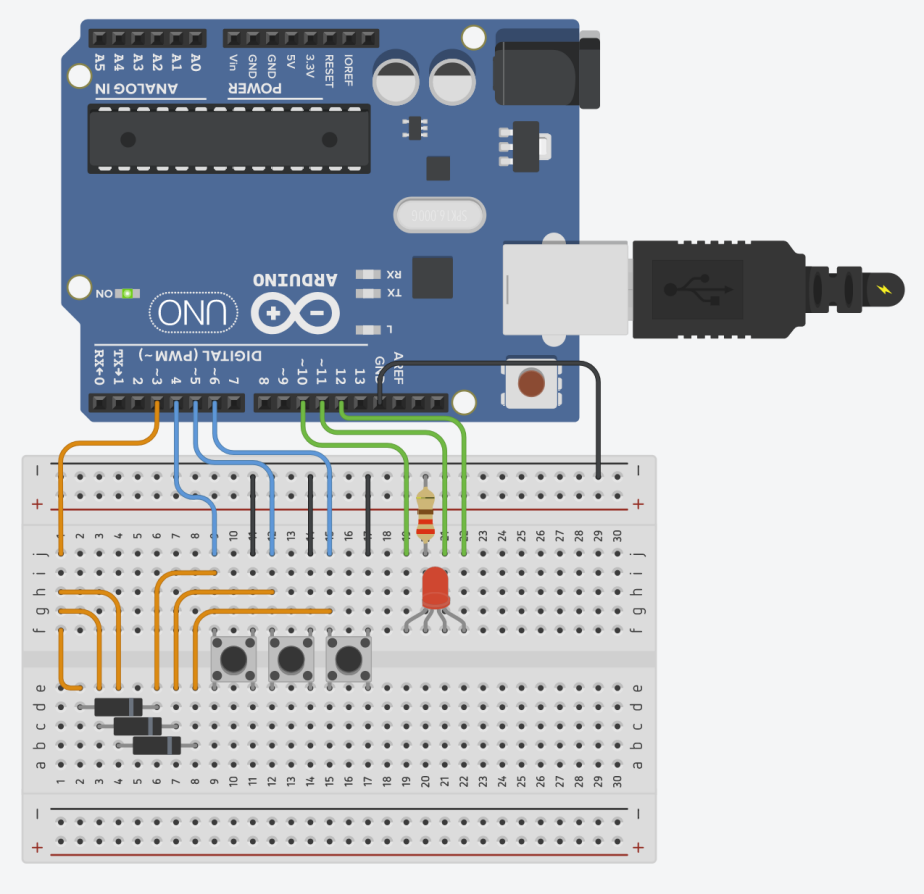
Figur 20 pinMode för interrupt pin, och inställning av ISR

```
for(int i = 0; i < MAX_BTN; i++)
{
    if(i < MAX_COLORS)
    {
        pinMode(rgb_led_pins[i], OUTPUT);
    }

    pinMode(btn_pins[i], INPUT_PULLUP);
}
```

Figur 21 pinMode för RGB-lysdiodanslutningar och knapp-anslutningar

För att säkerställa att TinkerCad hanterar ISR med funktionen **attachInterrupt**, och dioder, skapades ett nytt projekt. Vardera tryckknapp tänder en färg på RGB-lysdioden, röd, grön och blå respektive.

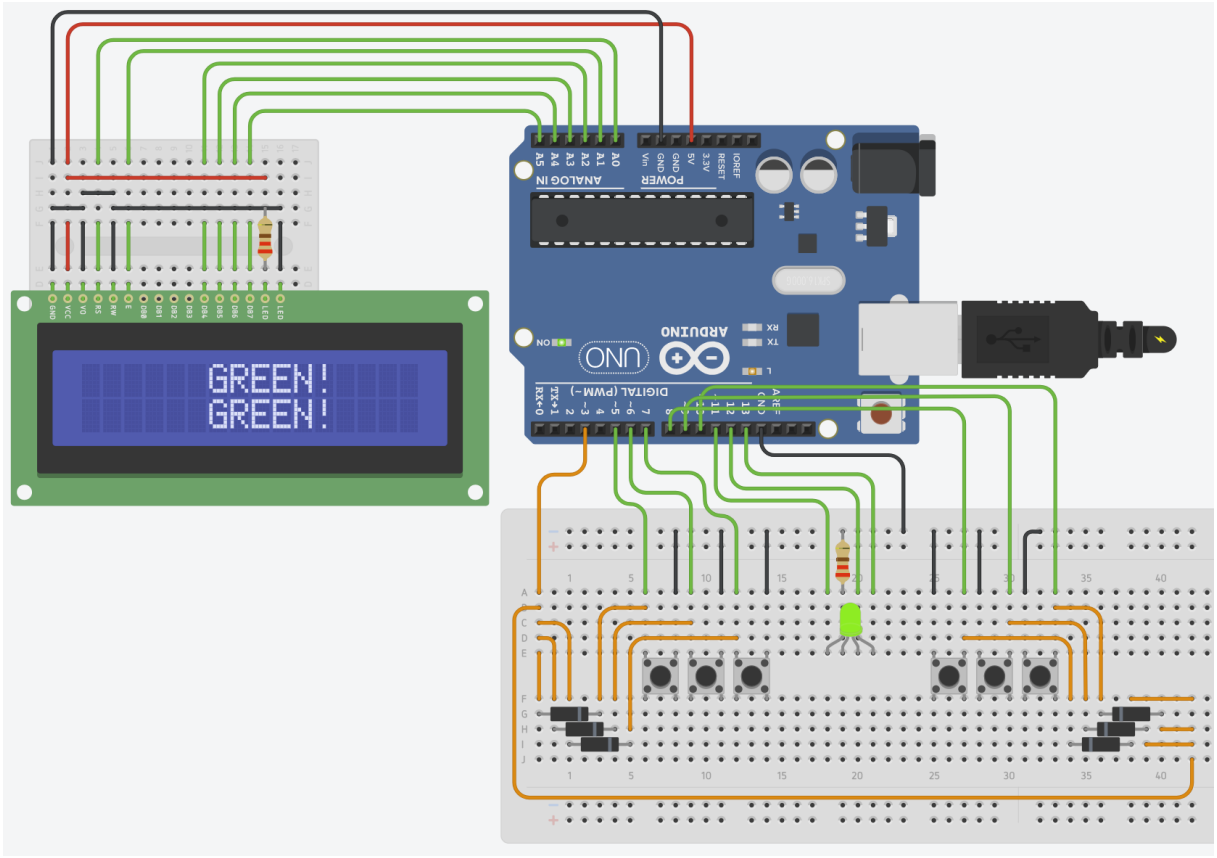


Figur 22 TinkerCad-projekt med ISR och dioder

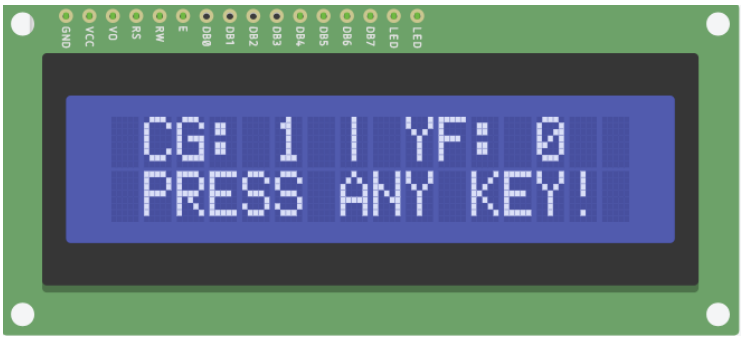
Kod för testet återges i Bilaga V: Testkod för ISR och dioder med TinkerCad.

Testet fungerade, vilket innebär att reaction-game kunde simuleras i TinkerCad:

Figur 23 Simulering av react-game i TinkerCad - LCD-Display: Inmatning av spelarnamn



Figur 24 Simulering av react-game i TinkerCad – Spelläge: Väntar på knapptryck för att avgöra vinnare

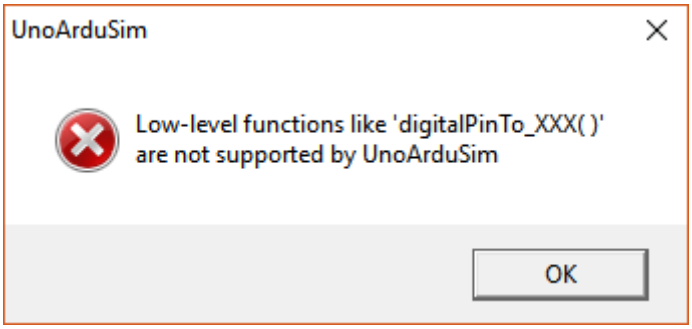


Figur 25 Simulering av react-game i TinkerCad - LCD-Display: Idle-läge, visar poängställning

2.6 Simulering i UnoArduSim

Simulering av reaction-game utfördes också i simuleringsverktyget UnoArduSim. Som bas för simulering användes koden för simuleringen i TinkerCad.

UnoArduSim saknar stöd för interrupts, LCD-displayer, samt diverse annan funktionalitet, således behövde koden ändras.



Figur 26 Felmeddelande i UnoArduSim, för avsaknad av stöd av lågnivå-funktioner

Exempel på ändringar som utfördes i koden för att möjliggöra simulering i UnoArduSim:

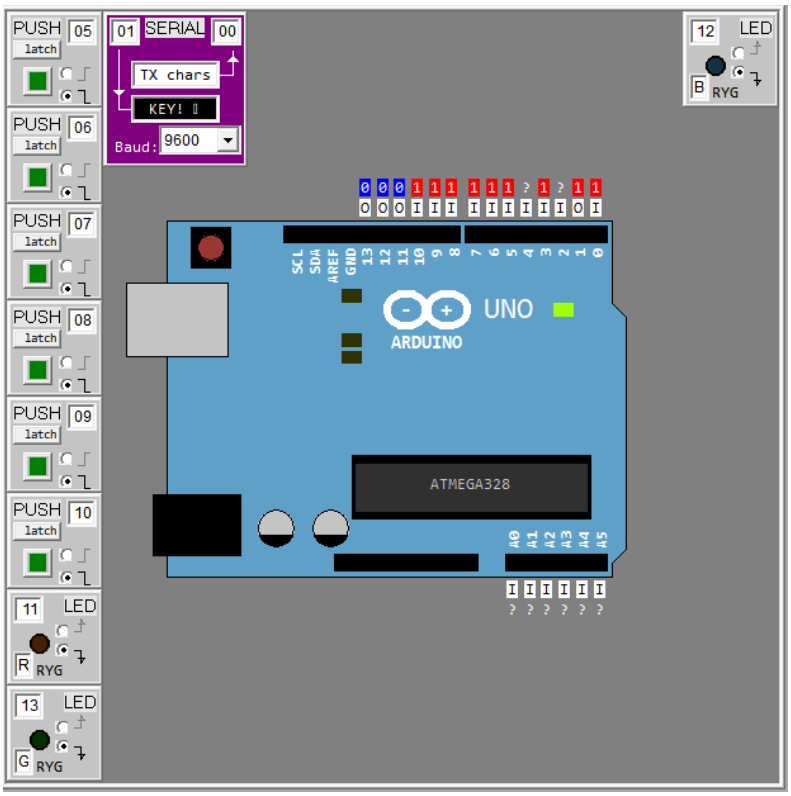
Fel / avsaknad av stöd	Åtgärd
Initiering av arraystorlekar kunde ej göras med macron. Exempel: **int array[NUM]**	Byte till hårdkodade heltal. Exempel: **int array[2]**

Avsaknad av stöd för ISR	Anropning av ISR manuellt innan varje kontroll om en knapp har blivit nedtryckt
	`while(!done)`
	`{`
	`  isr_button_click();`
	`  if(button_pressed)`
	`  {`
	Figur 27 Manuell anropning av ISR-funktion

Avsaknad av stöd för sammanläkning av strängar: Exempel **print(String1 + String2)**	Skriv ut strängar var för sig. Exempel **print(String1)** **print(String2)**
Avsaknad av stöd för LCD.	Använd **Serial.print()** för utskrift av data.
Funktionslika makron stöds ej.	Skriv om funktionslika macron till funktioner.

En länk till källkoden finns i GitHub-länkar.

Ett projekt skapades med sex knappar och tre lysdioder, i stället för LCD-display används USB Serial-kommunikation. Programmets serial monitor används för att visa data.



Figur 28 Komponenter i UnoArduSim

Figur 29 Serial Monitor i UnoArduSim

En demonstrationsvideo av simuleringen i UnoArduSim kan ses på Youtube, se YouTube-länkar.

3 Diskussion och slutsats

3.1 Utvärdering av TinkerCad

Nackdelar

Det webbaserade simuleringsverktyget TinkerCad upplevs vara riktat till projekt av mindre skala. Många vanliga komponenter och bibliotek saknas, exempelvis SSD1306-OLED displayen som användes i reaction-games originalutförande.

Ofta startade inte simuleringen i TinkerCad efter att projektets kod hade uppdaterats, en lösning på detta var att upprepade gånger försöka starta simuleringen eller prova igen vid ett senare tillfälle.

Detta problem upplevdes som mycket störande i arbetet med projektet.

TinkerCad upplevdes också vara långsamt.

Fördelar

TinkerCad upplevs som användarvänligt, komponenter läggs till och flyttas med drag-and-drop vilket kan kännas naturligt.

Verktyget upplevs också vara estetiskt tilltalande.

3.2 Utvärdering av UnoArduSim

Nackdelar

Likt TinkerCad verkar UnoArduSim sakna vissa vanliga komponenter, och koden behövde specialskrivas för att möjliggöra simulering.

Fördelar

Programmet upplevs som mycket snabbt, speciellt i jämförelse med TinkerCad. En klar fördel är också att koden kompileras och körs vid varje ny ändring av den.

3.3 Simulering

Simulering upplevs vara ett bra sätt att testa hårdvaruprojekt innan en fysisk produkt eller prototyp är byggd.

4 Bilagor

Bilaga I: Testkod för knappar med bibliotek Button med TinkerCad

```
#define PIN_BTN_LED_ON 1
#define PIN_BTN_LED_OFF 2
#define PIN_ONBOARD_LED 13

#include <Button.h>

/* Button object */
Button btn_led_on(PIN_BTN_LED_ON);
Button btn_led_off(PIN_BTN_LED_OFF);

void setup(void)
{
   btn_led_on.begin();
   btn_led_off.begin();
   pinMode(PIN_ONBOARD_LED, OUTPUT);
}

void loop(void)
{
   static bool led_state = false;
   static bool led_state_last = false;

   if(btn_led_on.pressed())
   {
      led_state = true;
   }

   if(btn_led_off.pressed())
   {
      led_state = false;
   }

   if(led_state != led_state_last)
   {
      digitalWrite(PIN_ONBOARD_LED, led_state);
      led_state_last = led_state;
   }

   delay(1);
}
```

Bilaga II: Testkod för knappar utan bibliotek med TinkerCad

```
#define PIN_BTN_LED_ON 1
#define PIN_BTN_LED_OFF 2
#define PIN_ONBOARD_LED 13
#define BUTTON_PRESSED 0

void setup(void)
{
   pinMode(PIN_BTN_LED_ON, INPUT_PULLUP);
   pinMode(PIN_BTN_LED_OFF, INPUT_PULLUP);
   pinMode(PIN_ONBOARD_LED, OUTPUT);
}

void loop(void)
{
   static bool led_state = false;
   static bool led_state_last = false;

   if(digitalRead(PIN_BTN_LED_ON) == BUTTON_PRESSED)
   {
      led_state = true;
   }

   if(digitalRead(PIN_BTN_LED_OFF) == BUTTON_PRESSED)
   {
      led_state = false;
   }

   if(led_state != led_state_last)
   {
      digitalWrite(PIN_ONBOARD_LED, led_state);
      led_state_last = led_state;
   }

   delay(1);
}
```

Bilaga III: Testkod för RGB-LED och resistor med TinkerCad

```
const byte pins_rgbled_1[] = { 3,  5, 6 };
const byte pins_rgbled_2[] = { 9, 10, 11 };

enum{ RED, BLUE, GREEN, COLORS_MAX };

#define PWM_MAX 255

void setup(void)
{
   for(int i = 0; i < 3; i++)
   {
      pinMode(pins_rgbled_1[i], OUTPUT);
      pinMode(pins_rgbled_2[i], OUTPUT);
   }
}

void loop(void)
{
   for(int color = RED; color < COLORS_MAX; color++)
   {
      for(int pwm = PWM_MAX; pwm >= 0; pwm--)
      {
         analogWrite(pins_rgbled_1[color], pwm);
         analogWrite(pins_rgbled_2[color], pwm);
         delay(30);
      }
   }
}
```

Bilaga IV: Testkod för LCD med TinkerCad

```
#include <LiquidCrystal.h>

enum
{
   LCD_RS = A0,
   LCD_EN = A1,
   LCD_D4 = A2,
   LCD_D5 = A3,
   LCD_D6 = A4,
   LCD_D7 = A5
};

LiquidCrystal lcd(
   LCD_RS, LCD_EN, LCD_D4, LCD_D5, LCD_D6, LCD_D7);

void lcd_print_string(String s, int col, int row, bool clear);

void setup()
{
   lcd.begin(16, 2);
}

void loop()
{
   for(int i = 0; i < 10; i++)
   {
      lcd_print_string("Player 1: " + String(i),     0, 0, true);
      lcd_print_string("Player 2: " + String(i + 2), 0, 1, false);
      delay(1000);
   }
}

void lcd_print_string(String s, int col, int row, bool clear)
{
   if(clear)
   {
      lcd.clear();
   }

   lcd.setCursor(col, row);
   lcd.print(s);
}
```

Bilaga V: Testkod för ISR och dioder med TinkerCad

```
#define INT_PIN 3
#define BUTTON_STATUS_DOWN 0
#define ERROR -1

enum { RED, GREEN, BLUE, COLOR_MAX, BTN_MAX = COLOR_MAX };
const int rgb_led_pin[] = { 10, 11, 12 };
const int rgb_btn_pin[] = { 4, 5, 6 };

bool button_pressed;

void setup()
{
  for(int i = 0; i < BTN_MAX; i++)
  {
    pinMode(rgb_led_pin[i], OUTPUT);
    pinMode(rgb_btn_pin[i], INPUT_PULLUP);
  }

  pinMode(INT_PIN, INPUT_PULLUP);
  attachInterrupt(
    digitalPinToInterrupt(INT_PIN), isr_button_click, FALLING);

  button_pressed = false;
  Serial.begin(9600);
}

void loop()
{
  if(button_pressed)
  {
    int color = get_pressed_button();
    Serial.println("Setting color: " + String(color));
    set_rgb_led(color);
    button_pressed = false;
  }

  delay(1);
}

void isr_button_click(void)
{
  Serial.println("Int!");
  button_pressed = true;
}
```

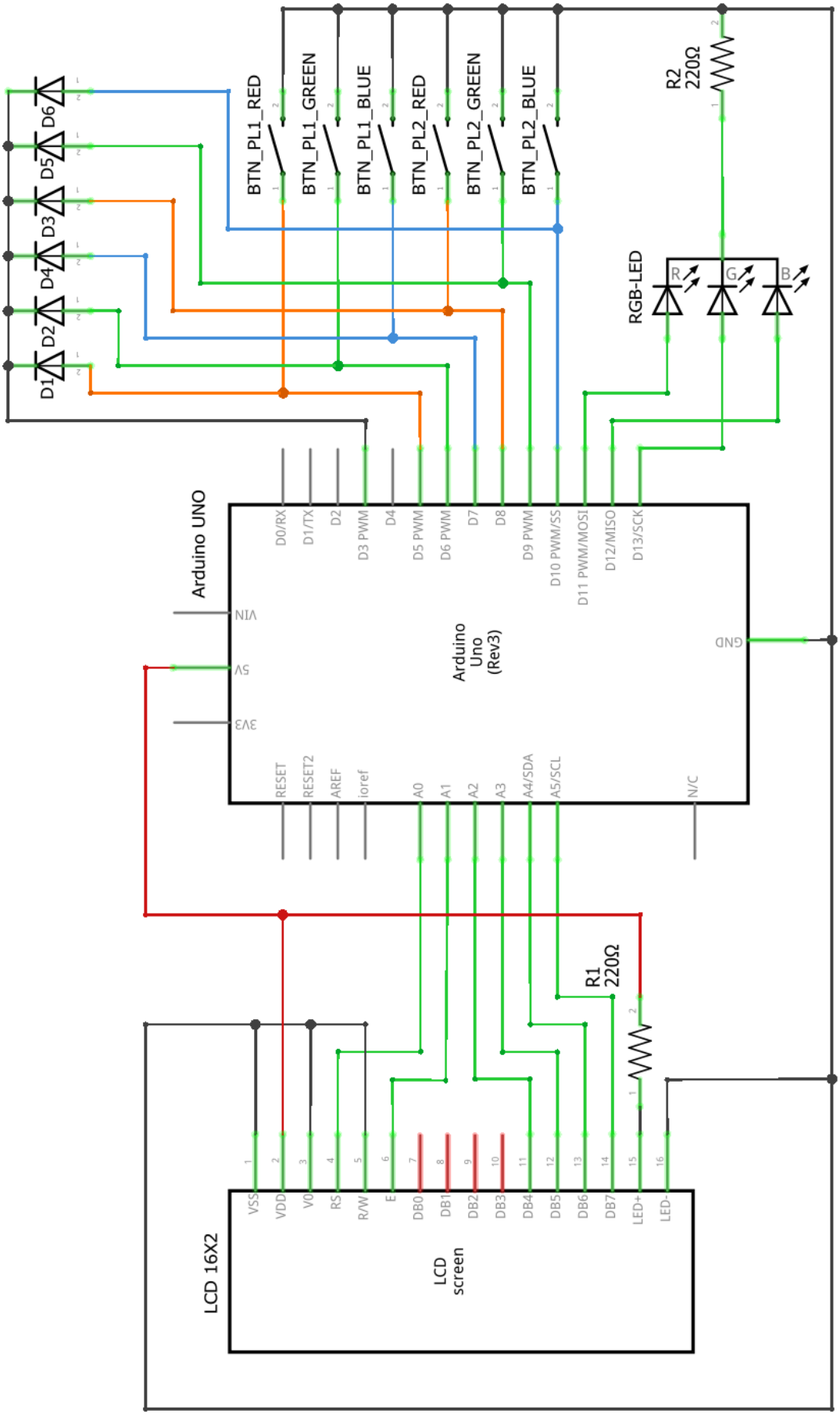
```
int get_pressed_button(void)
{
   for(int i = 0; i < BTN_MAX; i++)
   {
      if(digitalRead(rgb_btn_pin[i]) == BUTTON_STATUS_DOWN)
      {
         Serial.println("Btn press: " + String(i));
         return i;
      }
   }

   return ERROR;
}

void set_rgb_led(int color)
{
   for(int i = 0; i < COLOR_MAX; i++)
   {
      digitalWrite(rgb_led_pin[i], LOW);
   }

   digitalWrite(rgb_led_pin[color], HIGH);
}
```

Bilaga VI: Kopplingsschema



Bilaga VII: Kod för simulering i TinkerCad

```
#include <Arduino.h>
#include <LiquidCrystal.h>

enum{ RED, GREEN, BLUE, MAX_COLORS };

enum{ BTN_PL1_RED, BTN_PL1_GREEN, BTN_PL1_BLUE,
    BTN_PL2_RED, BTN_PL2_GREEN, BTN_PL2_BLUE,
    MAX_BUTTONS };

enum{ PLAYER_1, PLAYER_2, MAX_PLAYERS };

enum
{
  LCD_RS = A0,
  LCD_EN = A1,
  LCD_D4 = A2,
  LCD_D5 = A3,
  LCD_D6 = A4,
  LCD_D7 = A5
};

#define INT_PIN           3
#define NO_BUTTON_PRESSED   -1
#define UP              true
#define DOWN             false
#define LCD_COLS          16
#define LCD_ROWS           2
#define COUNTDOWN_NUMBER    3

#define BTN_TO_COLOR(b)     (b >= MAX_COLORS ? b - 3 : b)
#define BTN_TO_PLAYER(b)    (b > 2 ? PLAYER_2 : PLAYER_1 )
#define INC_ONE_MAX(v, m)   (v + 1 > m ? m : v + 1)
#define LCD_MIDDLE(s_len)   (LCD_COLS / 2 - s_len / 2)

const int rgb_led_pins[] =  { 11, 13, 12 };
const int btn_pins[] =      { 5, 6, 7, 8, 9, 10 };

bool button_pressed;
int last_button_pressed;

int player_score[MAX_PLAYERS];
String player_name[MAX_PLAYERS];

LiquidCrystal lcd(
  LCD_RS, LCD_EN, LCD_D4, LCD_D5, LCD_D6, LCD_D7);

void game_mode_set_names(void);
bool game_mode_idle(void);
void game_mode_countdown(void);
int  game_mode_play(void);
void rgb_led_off(void);
```

```
void rgb_led_set_color(int);
int  rgb_get_random_color(void);
void isr_button_click(void);
void lcd_print_string(String, int, int, bool);
int  button_get_which_pressed(void);
void player_name_cycle_char(int, bool, int);

void setup()
{
   randomSeed(analogRead(0));

   pinMode(INT_PIN, INPUT_PULLUP);

   for(int i = 0; i < MAX_BUTTONS; i++)
   {
      if(i < MAX_COLORS)
      {
         pinMode(rgb_led_pins[i], OUTPUT);
      }

      pinMode(btn_pins[i], INPUT_PULLUP);
   }

   attachInterrupt(
      digitalPinToInterrupt(INT_PIN), isr_button_click, FALLING);

   player_name[PLAYER_1].reserve(2);
   player_name[PLAYER_2].reserve(2);
   player_score[PLAYER_1] = 0;
   player_score[PLAYER_2] = 0;

   lcd.begin(LCD_COLS, LCD_ROWS);

   rgb_led_off();
   button_pressed = false;

   game_mode_set_names();
}
```

```
void loop()
{
   while(!game_mode_idle());

   game_mode_countdown();

   int winner = game_mode_play();

   lcd_print_string("WINNER: ", 0, 0, true);
   lcd_print_string(player_name[winner] + "!", 0, 1, false);

   player_score[winner]++;

   delay(2000);
   rgb_led_off();
   button_pressed = false;
}

void isr_button_click(void)
{
   if(!button_pressed)
   {
      last_button_pressed = button_get_which_pressed();
      button_pressed = true;
   }
}
```

```
void game_mode_set_names(void)
{
   player_name[PLAYER_1] = player_name[PLAYER_2] = "AA";
   int char_index[MAX_PLAYERS] = { 0, 0 };
   bool done = false;

   lcd_print_string("SET NAME P1: " + player_name[PLAYER_1], 0, 0, true);
   lcd_print_string("SET NAME P2: " + player_name[PLAYER_2], 0, 1, false);

   while(!done)
   {
      if(button_pressed)
      {
         int p = BTN_TO_PLAYER(last_button_pressed);

         switch(BTN_TO_COLOR(last_button_pressed))
         {
            case RED:
               player_name_cycle_char(p, DOWN, char_index[p]);
               break;

            case GREEN:
               char_index[p] = INC_ONE_MAX(char_index[p], 2);
               break;

            case BLUE:
               player_name_cycle_char(p, UP, char_index[p]);
               break;

            default:
               break;
         }

         lcd_print_string(
            "SET NAME P1: " + player_name[PLAYER_1], 0, 0, true);
         lcd_print_string(
            "SET NAME P2: " + player_name[PLAYER_2], 0, 1, false);

         button_pressed = false;
         done = (char_index[PLAYER_1] == 2 && char_index[PLAYER_2] == 2);
      }
   }
}
```

```
bool game_mode_idle(void)
{
  static bool dots = false;

  lcd_print_string(
    player_name[PLAYER_1] + ": " + player_score[PLAYER_1] + " | " +
    player_name[PLAYER_2] + ": " + player_score[PLAYER_2], 1, 0, true);

  lcd_print_string(dots ? "*PRESS ANY KEY!*" : " PRESS ANY KEY! ",
    0, 1, false);

  dots = !dots;

  if(button_pressed)
  {
    button_pressed = false;
    return true;
  }

  delay(500);

  return false;
}

void game_mode_countdown(void)
{
  static const String message = "GET READY!";
  int count = COUNTDOWN_NUMBER;

  lcd_print_string(message, LCD_MIDDLE(message.length()), 0, true);
  delay(1000);

  while(count)
  {
    lcd_print_string(String(count--), LCD_COLS/2, 1, false);
    delay(1000);
  }

  button_pressed = false;
}
```

```
int game_mode_play(void)
{
  static const String color_name[MAX_COLORS] =
  {
    "RED", "GREEN", "BLUE"
  };

  int c = rgb_get_random_color();
  int winner;

  rgb_led_set_color(c);

  lcd_print_string(
    color_name[c] + "!", LCD_MIDDLE(color_name[c].length()), 0, true);
  lcd_print_string(
    color_name[c] + "!", LCD_MIDDLE(color_name[c].length()), 1, false);

  while(!button_pressed);

  int player_to_press_first = BTN_TO_PLAYER(last_button_pressed);
  int button_color_pressed =  BTN_TO_COLOR(last_button_pressed);

  return (button_color_pressed == c ?
    player_to_press_first : !player_to_press_first);
}

void player_name_cycle_char(int player, bool direction, int index)
{
  char c = player_name[player][index];

  if(direction == UP)
  {
    c = c + 1 > 'Z' ? 'A' : c + 1;
  }

  else
  {
    c = c - 1 < 'A' ? 'Z' : c - 1;
  }

  player_name[player][index] = c;
}
```

```
int button_get_status(int button)
{
   return !digitalRead(btn_pins[button]);
}

int button_get_which_pressed(void)
{
   for(int i = 0; i < MAX_BUTTONS; i++)
   {
      if(button_get_status(i))
      {
         return i;
      }
   }

   return NO_BUTTON_PRESSED;
}

void rgb_led_off(void)
{
   analogWrite(rgb_led_pins[BLUE], 0);
   analogWrite(rgb_led_pins[RED], 0);
   analogWrite(rgb_led_pins[GREEN], 0);
}

void rgb_led_set_color(int c)
{
   rgb_led_off();
   analogWrite(rgb_led_pins[c], 255);
}

int rgb_get_random_color(void)
{
   return random(RED, MAX_COLORS);
}

void lcd_print_string(String s, int col, int row, bool clear)

{
   if(clear)
   {
      lcd.clear();
   }

   lcd.setCursor(col, row);
   lcd.print(s);
}
```