|  |
| --- |
| 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 4](#_Toc500696208)

[1.1 Syfte och mål 4](#_Toc500696209)

[1.2 Projektkrav 4](#_Toc500696210)

[1.3 Val av simuleringsverktyg 4](#_Toc500696211)

[1.4 Bakgrund 5](#_Toc500696212)

[1.4.1 Beskrivning av tidigare projekt 5](#_Toc500696213)

[1.5 Länkar 7](#_Toc500696214)

[1.5.1 GitHub-länkar 7](#_Toc500696215)

[1.5.2 YouTube-länkar 7](#_Toc500696216)

[1.5.3 Andra länkar 7](#_Toc500696217)

[1.6 Noteringar 8](#_Toc500696218)

[2 Genomförande och resultat 9](#_Toc500696219)

[2.1 Använd programvara 9](#_Toc500696220)

[2.2 Planering 9](#_Toc500696221)

[2.3 Metod 10](#_Toc500696222)

[2.3.1 Skapande av simulationer med verktyget TinkerCad 10](#_Toc500696223)

[2.4 Test av komponenter från reaction-game i TinkerCad 12](#_Toc500696224)

[2.4.1 Test av knappar och Arduino UNO i TinkerCad 12](#_Toc500696225)

[2.4.2 Test av RGB-Lysdioder och resistorer i TinkerCad 13](#_Toc500696226)

[2.4.3 Test av display i TinkerCad 14](#_Toc500696227)

[2.5 Skapande av kod och simuleringsprojekt i TinkerCad 15](#_Toc500696228)

[2.6 Simulering i UnoArduSim 20](#_Toc500696229)

[3 Diskussion och slutsats 22](#_Toc500696230)

[3.1 Utvärdering av TinkerCad 22](#_Toc500696231)

[3.2 Utvärdering av UnoArduSim 22](#_Toc500696232)

[3.3 Simulering 22](#_Toc500696233)

[4 Bilagor 23](#_Toc500696234)

# Inledning

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

## 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.

## 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.

## 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.

## Bakgrund

### Beskrivning av tidigare projekt

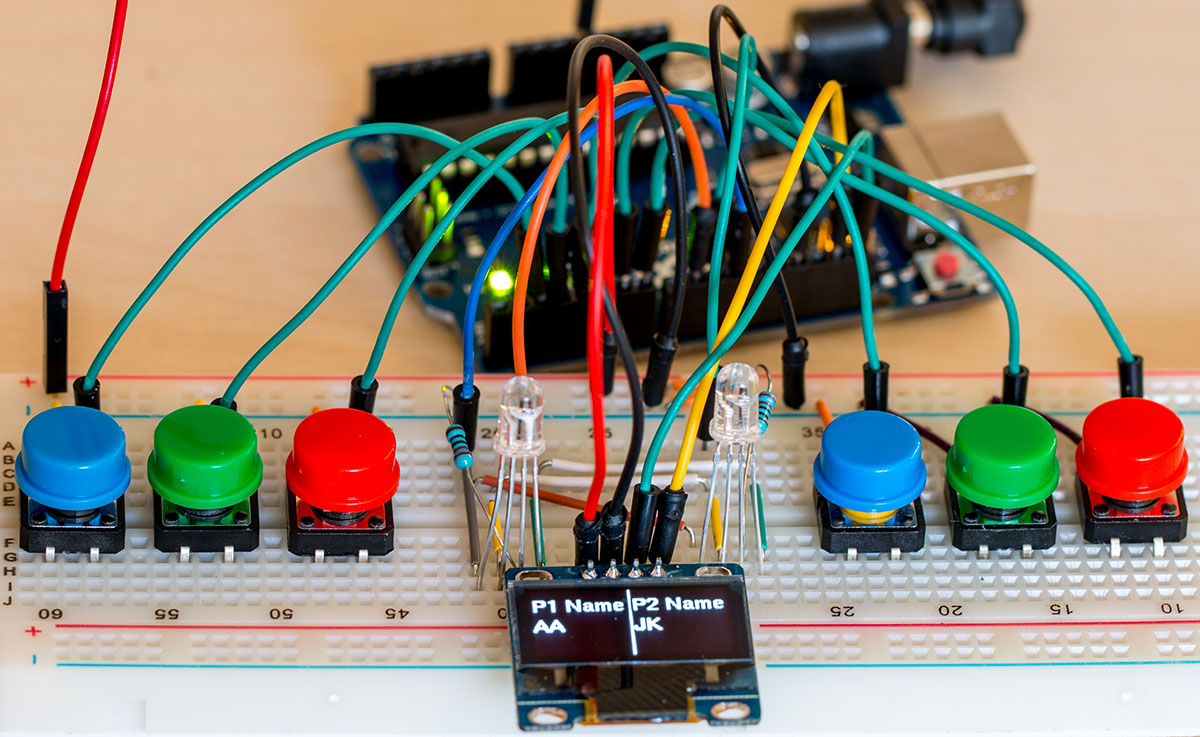
Det befintliga projekt som valdes för simulering kallas *reactionGameRGBLed*, och är ett reaktionsspel för två personer.

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 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](#_Länkar).

## Länkar

### 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>

### 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>

### Andra länkar

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

## 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](#_GitHub-länkar) för att visa koden på projektets GitHub-sida.

# Genomförande och resultat

## Använd programvara

Programvaror som har använts i projektet

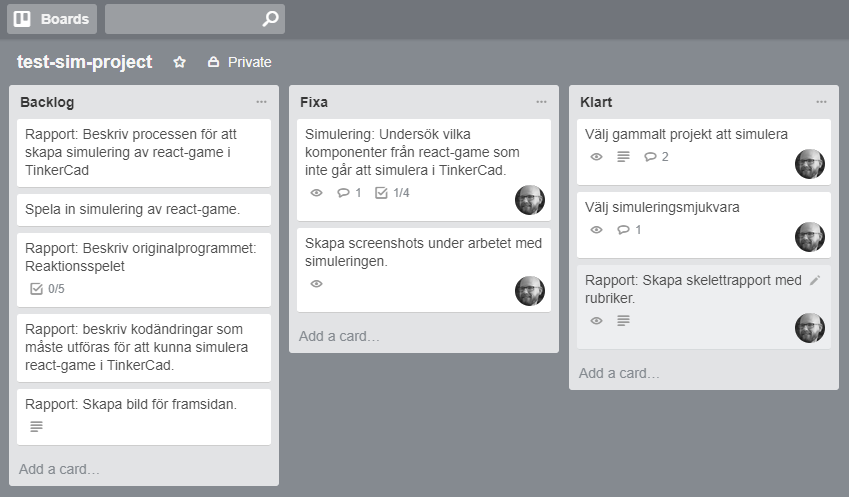
* **Microsoft Word 2016**
* **Autodesk TinkerCad**
  + Webbaserat simuleringsprogram med stöd för Arduino och vanliga komponenter.
* **UnoArduSim**
  + Simuleringsprogram för Arduino Uno, utvecklat av Stan Simmons
* **Atom text editor**
* **Git**
  + Versionshantering
* **Fritzing**
  + Program för att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

## 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.

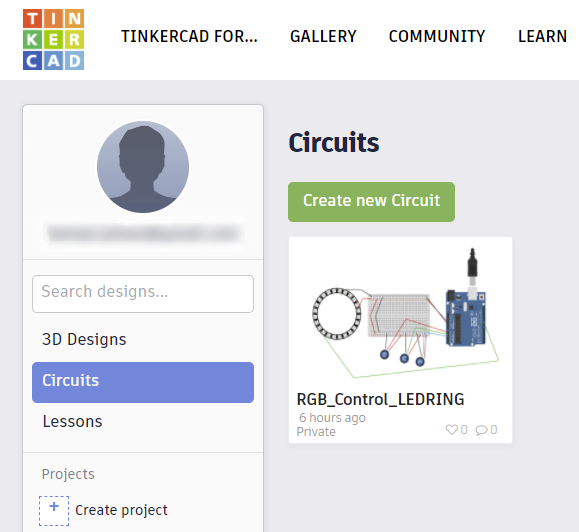


Figur Planering med verktyget Trello

## Metod

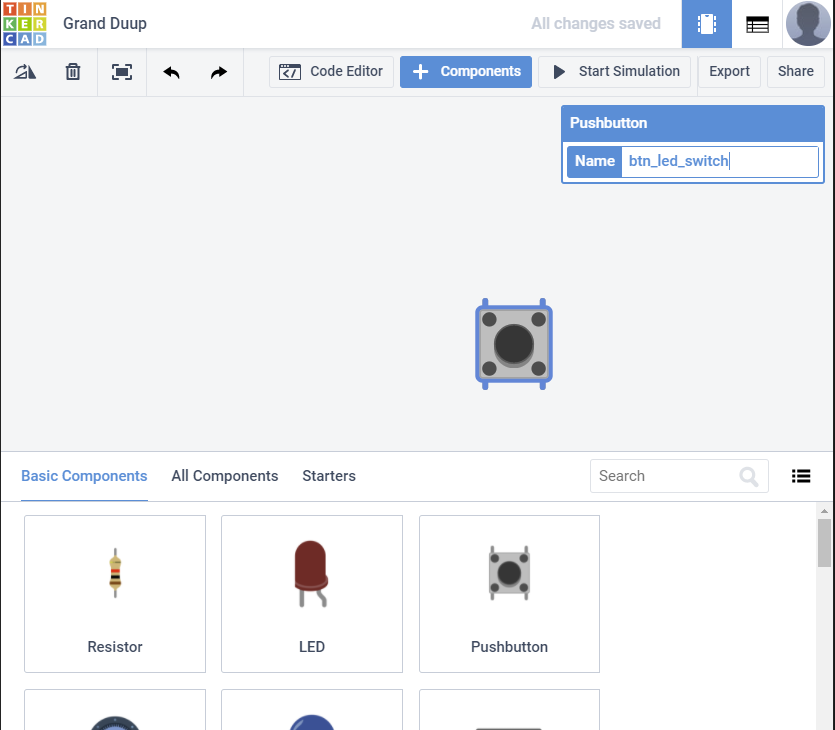
### Skapande av simulationer med verktyget TinkerCad

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 Undermenyn Circuits på TinkerCad

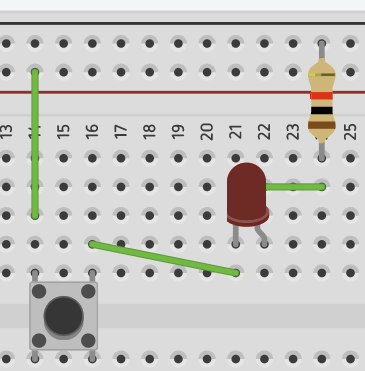
När ett nytt kretssimuleringsprojekt har skapats kan komponenter som ska användas läggas till med knappen *+ Components.* Komponenter visas i en lista och kan dras in till arbetsytan med muspekaren.



Figur 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.

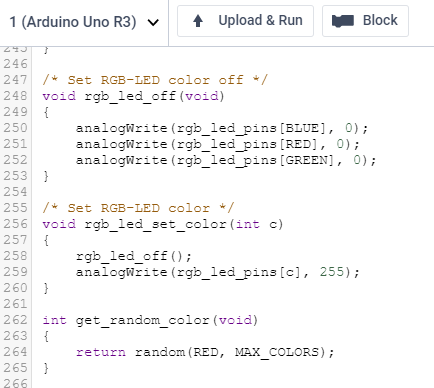


Figur Sammankopplade komponenter i TinkerCad, med gröna kablar

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 Knappen "Code Editor" på TinkerCad



Figur Kod i kod-editor på TinkerCad

## 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.

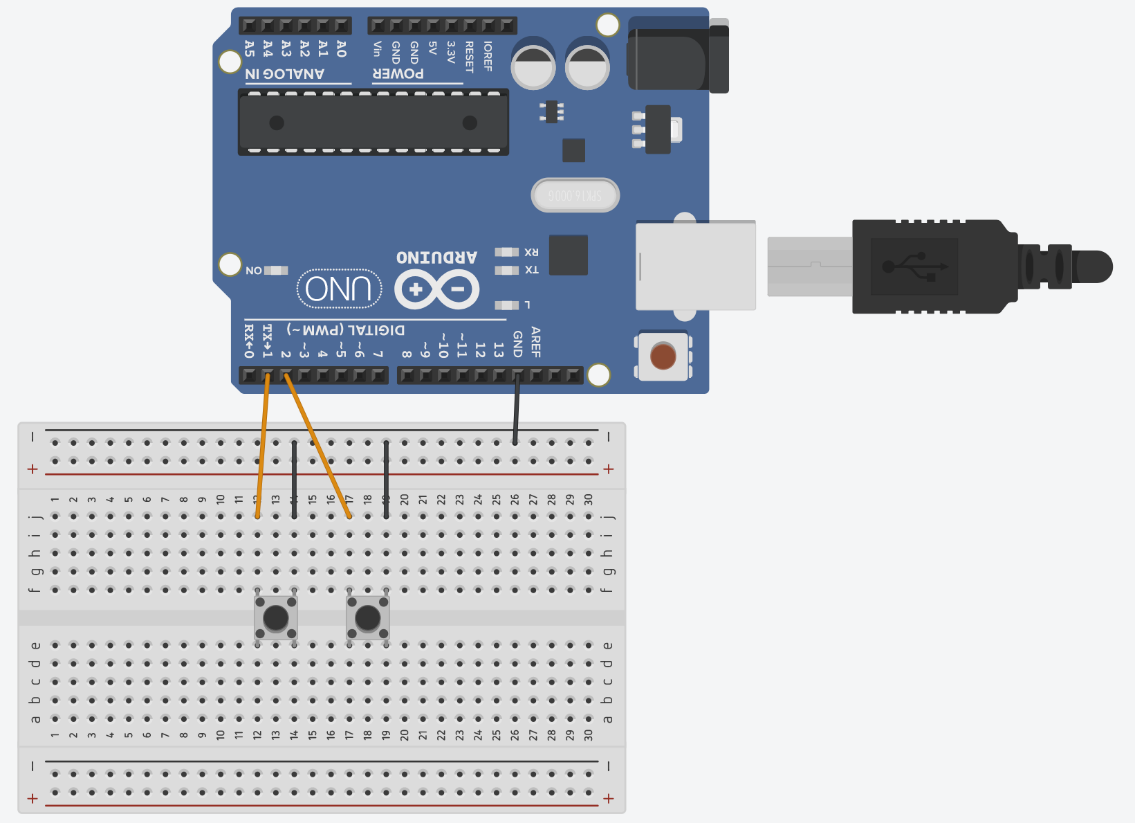
### Test av knappar och Arduino UNO i TinkerCad

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](#_Bilaga_I:_Testkod).

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 TinkerCad-projekt med två knappar och en Arduino UNO

Vid simuleringsstart gavs följande fel:

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

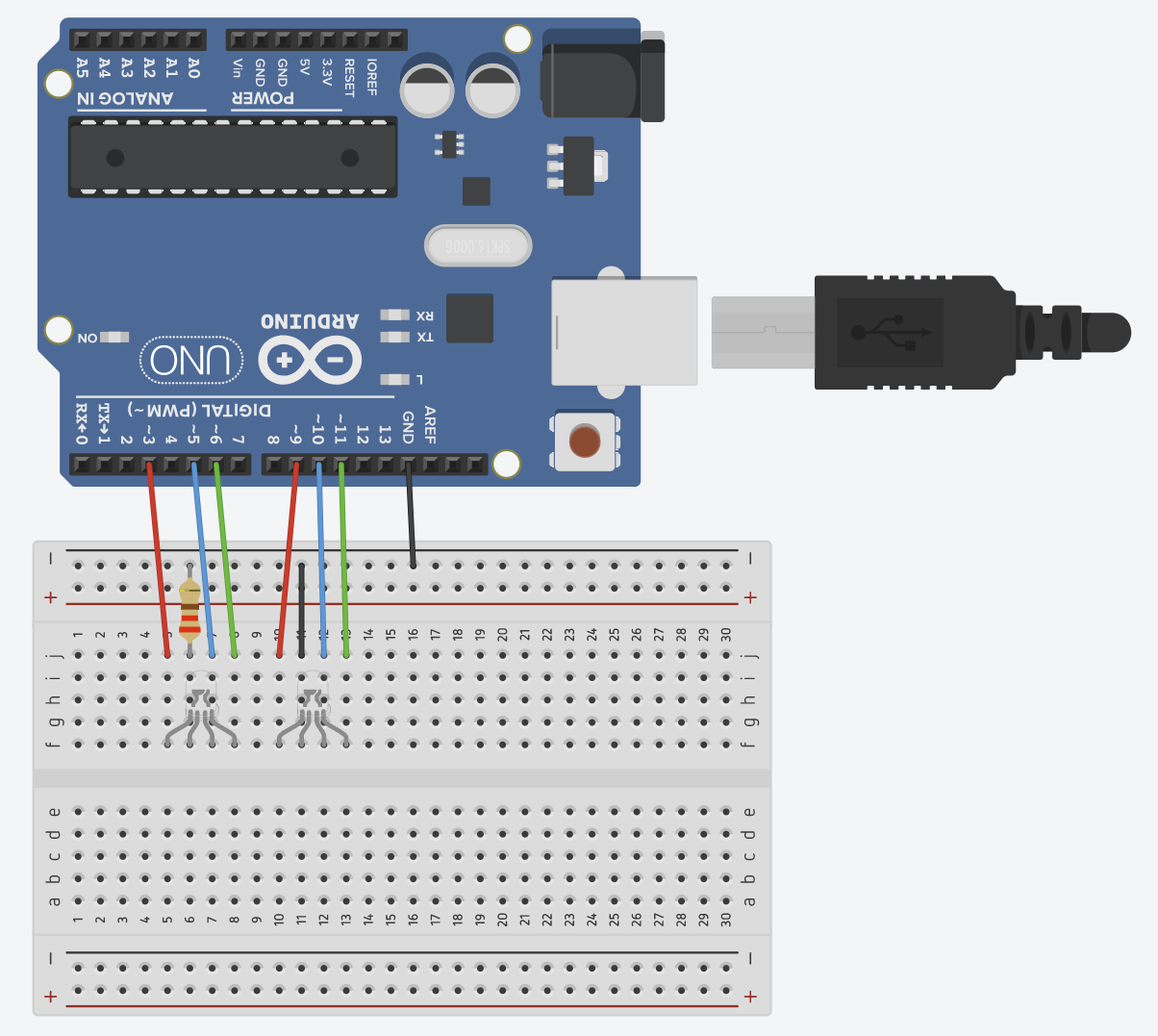
Således antogs det att biblioteket *Button* ej går att använda i TinkerCad. Testkoden byggdes om, se [Bilaga II: Testkod för knappar utan bibliotek med TinkerCad](#_Bilaga_II:_Testkod), så att inget bibliotek för knappar används. Ingen hänsyn togs till *bouncing* i denna kod.

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

### Test av RGB-Lysdioder och resistorer i TinkerCad

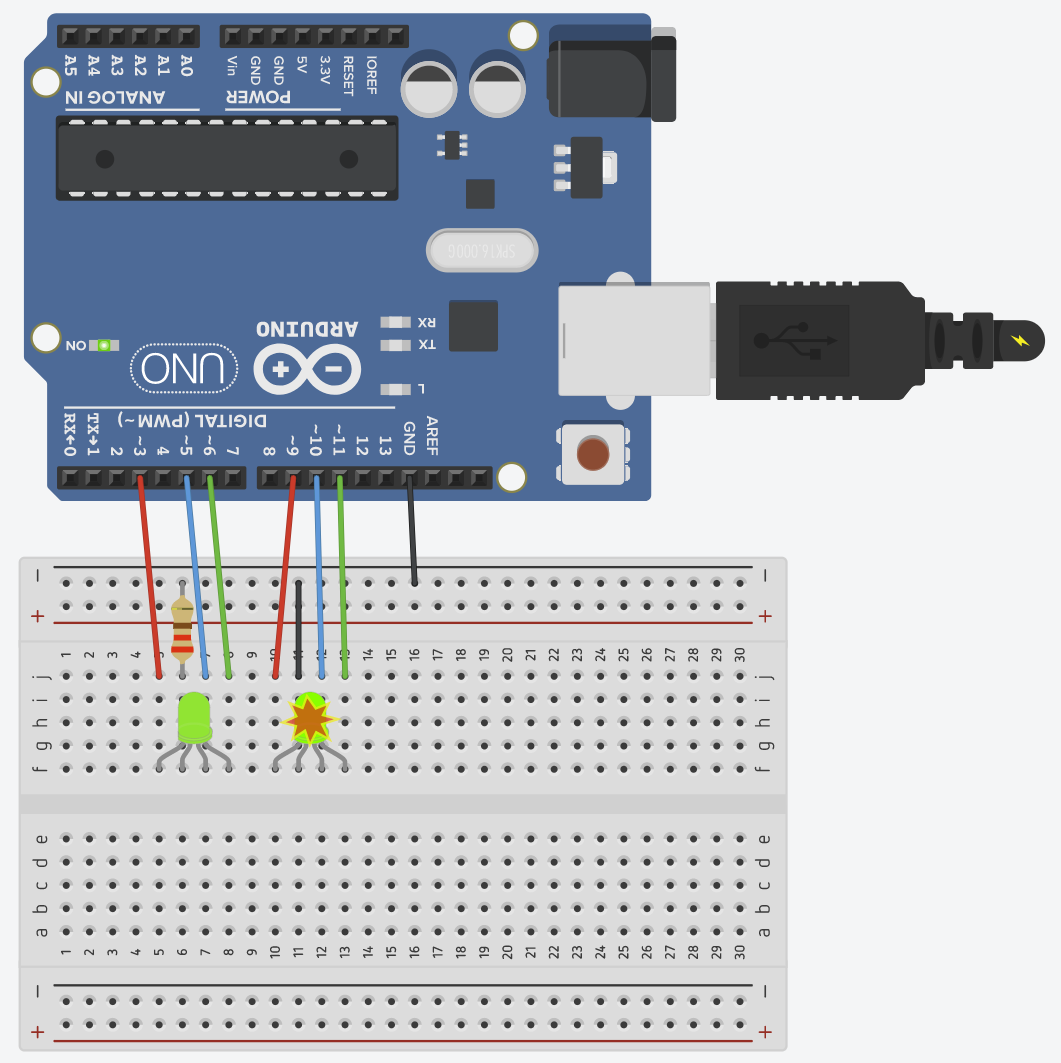
Reaction-game innehåller två RGB-lysdioder och två resistorer, för att säkerställa att dessa komponenter fungerar i TinkerCad skapades en testkod, se [Bilaga III: Testkod för RGB-LED och resistor med TinkerCad](#_Bilaga_3:_Testkod).

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 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.



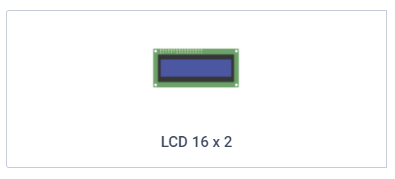
Figur Indikation i TinkerCad, att en RGB-Lysdiod (den högra i bilden) har gått sönder

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

### Test av display i TinkerCad

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

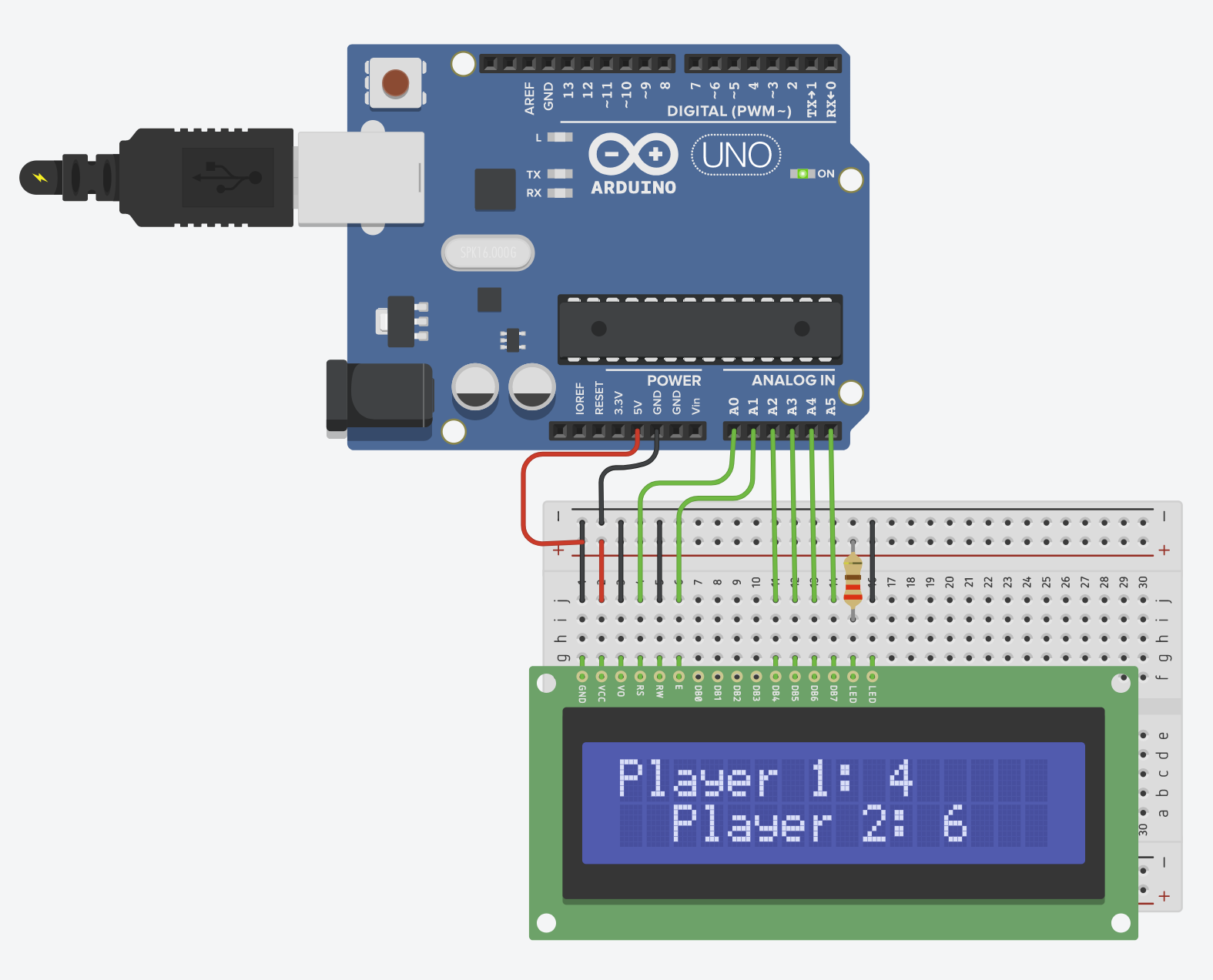
Vid inspektion av TinkerCads komponentlista visar det sig att det inte finns stöd för SSD1306, dock finns en annan display, kallad *LCD 16 x 2*. Denna väljs som ett lämpligt alternativ till SSD1306-displayen.



Figur LCD 16 x 2-display i TinkerCads komponentlista

Testkod skapades för denna display, se [Bilaga IV: Testkod för LCD med TinkerCad](#_Bilaga_IV:_Testkod).

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



Figur 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.

## 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.

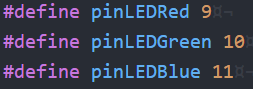
Koden för simulering i TinkerCad återfinns i [Bilaga VII: Kod för simulering i TinkerCad](#_Bilaga_VII:_Kod).   
Koden i bilagan saknar kommentarer, för fullständig version se länk i [GitHub-länkar](#_GitHub-länkar).

En video av den utförda simuleringen i TinkerCad kan ses på YouTube, se länk i [YouTube-länkar](#_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:



Figur Variabler i ursprungskod

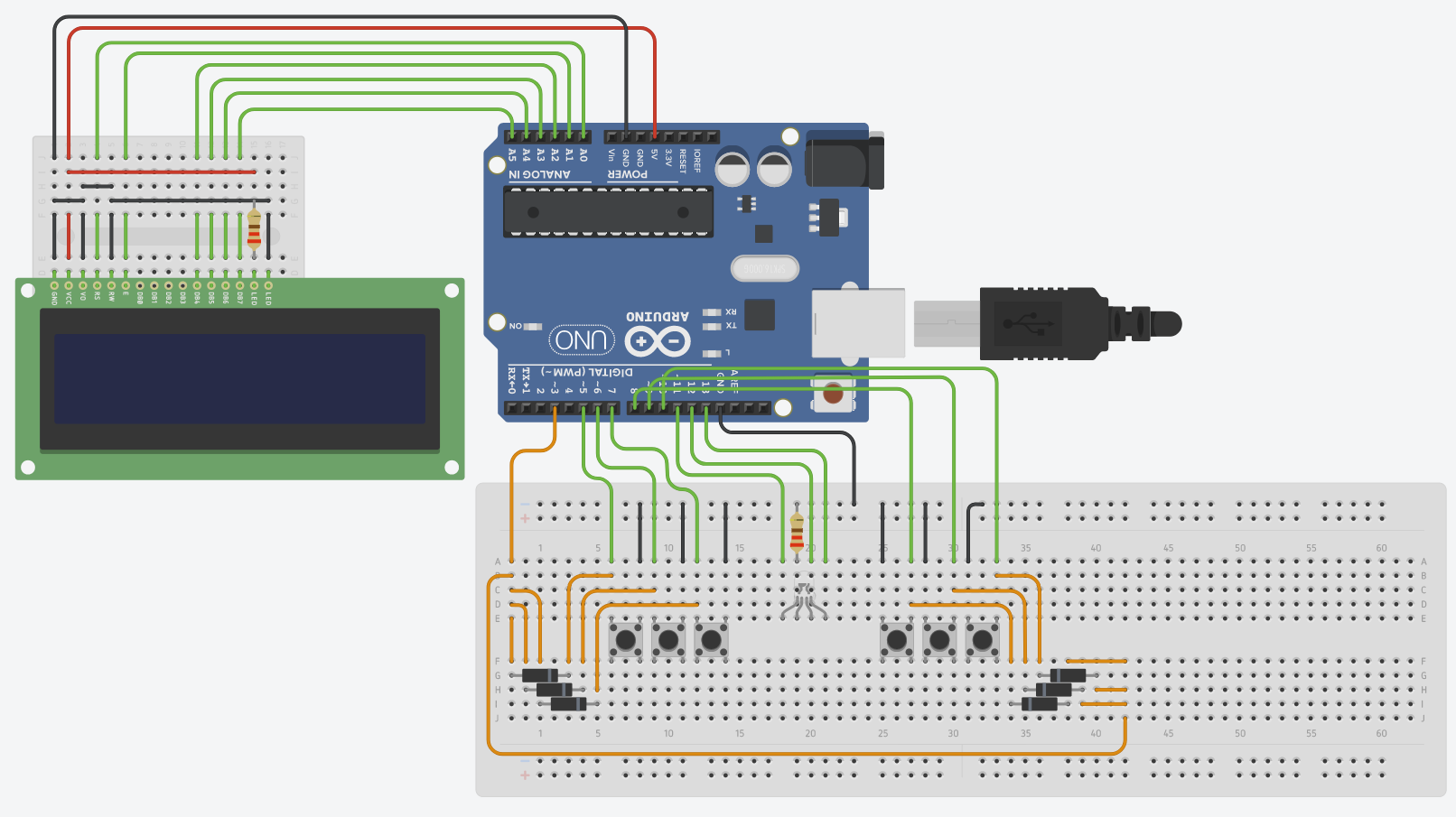


Figur Variabler i ny kod

* **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.

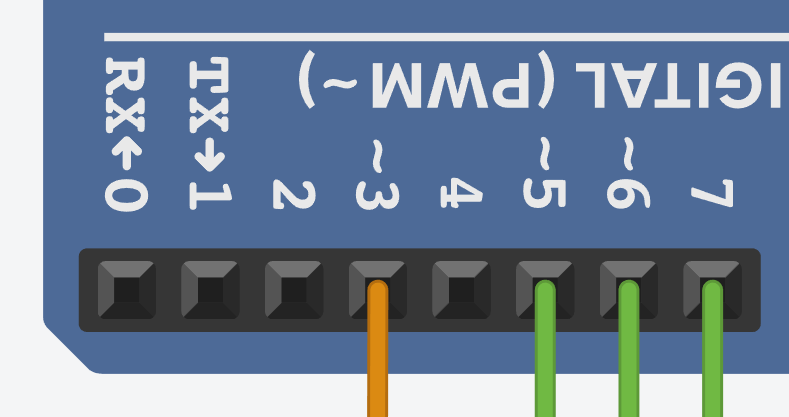
Kopplingsschema ses i [Bilaga VI: Kopplingsschema](#_Bilaga_VI:_Kopplingsschema).



Figur 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.

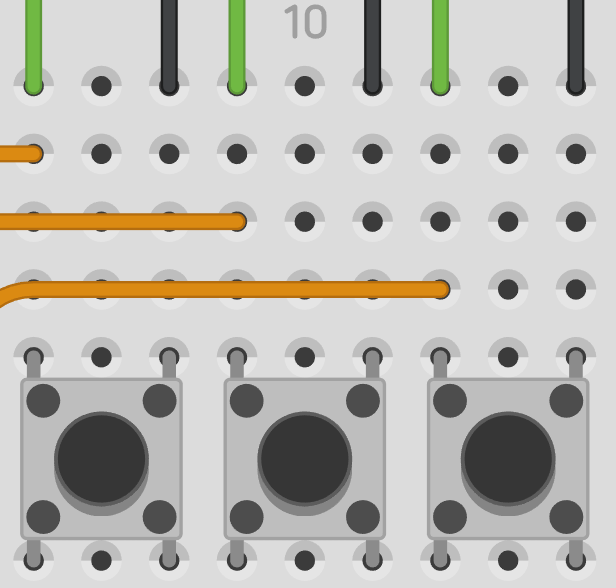
PIN3 används för trigger av ISR i koden.



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

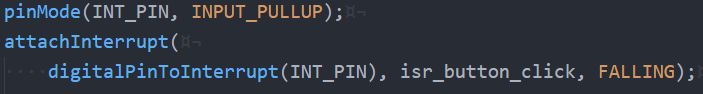
Alla anslutningar som är kopplade till knapparna, och interrupt-pin PIN3 är satta till läge INPUT\_PULLUP. I detta läge används MCUns interna pull-up på anslutningarna. Anslutningarnas läge sätts med funktionen **pinMode.**

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

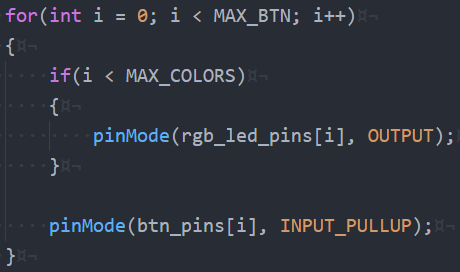


Figur Anslutningar för knappar, orange kabel är kopplad till PIN3 för interrupt, svart kabel är jord

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.

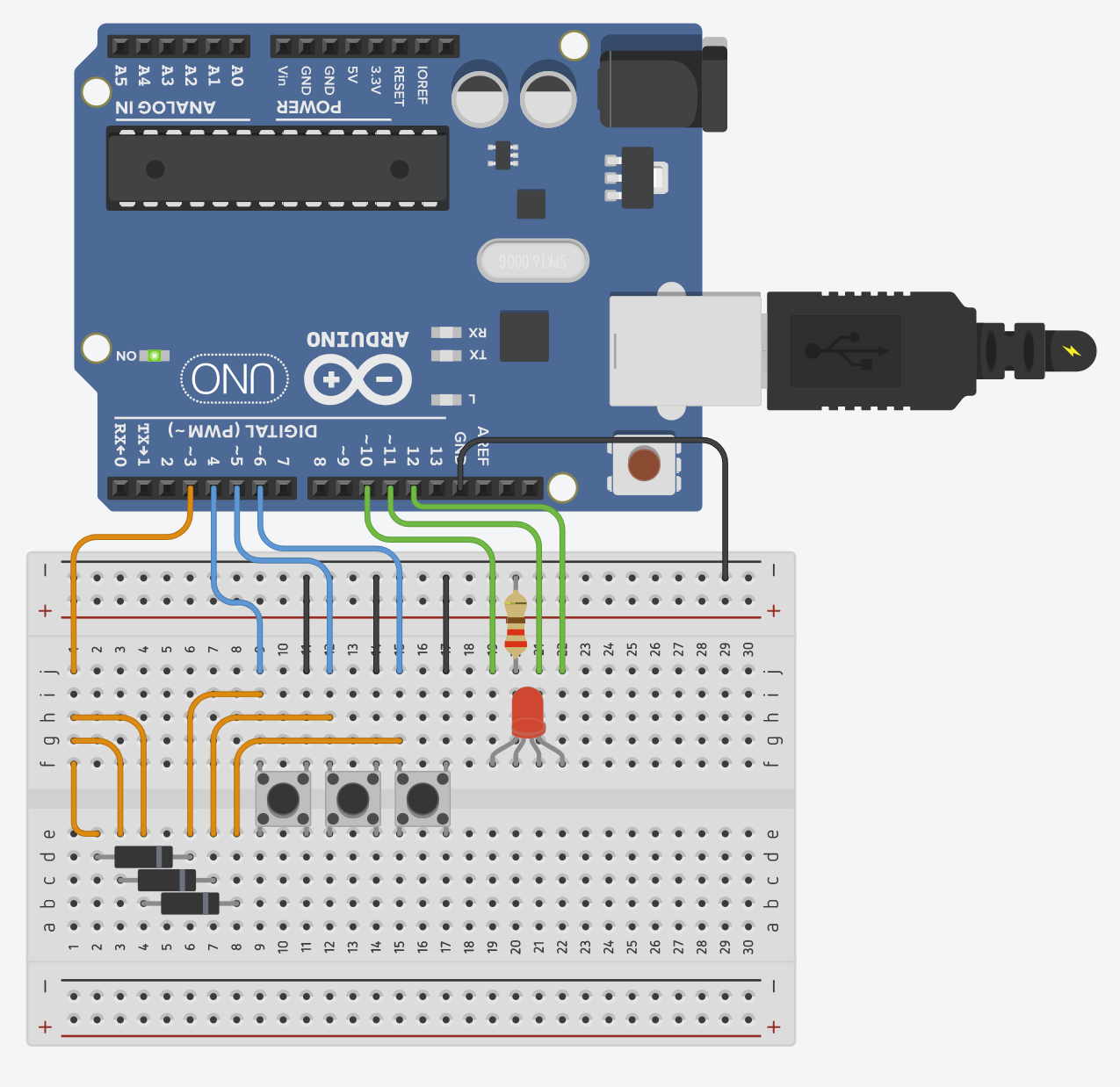


Figur pinMode för interrupt pin, och inställning av ISR



Figur 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.



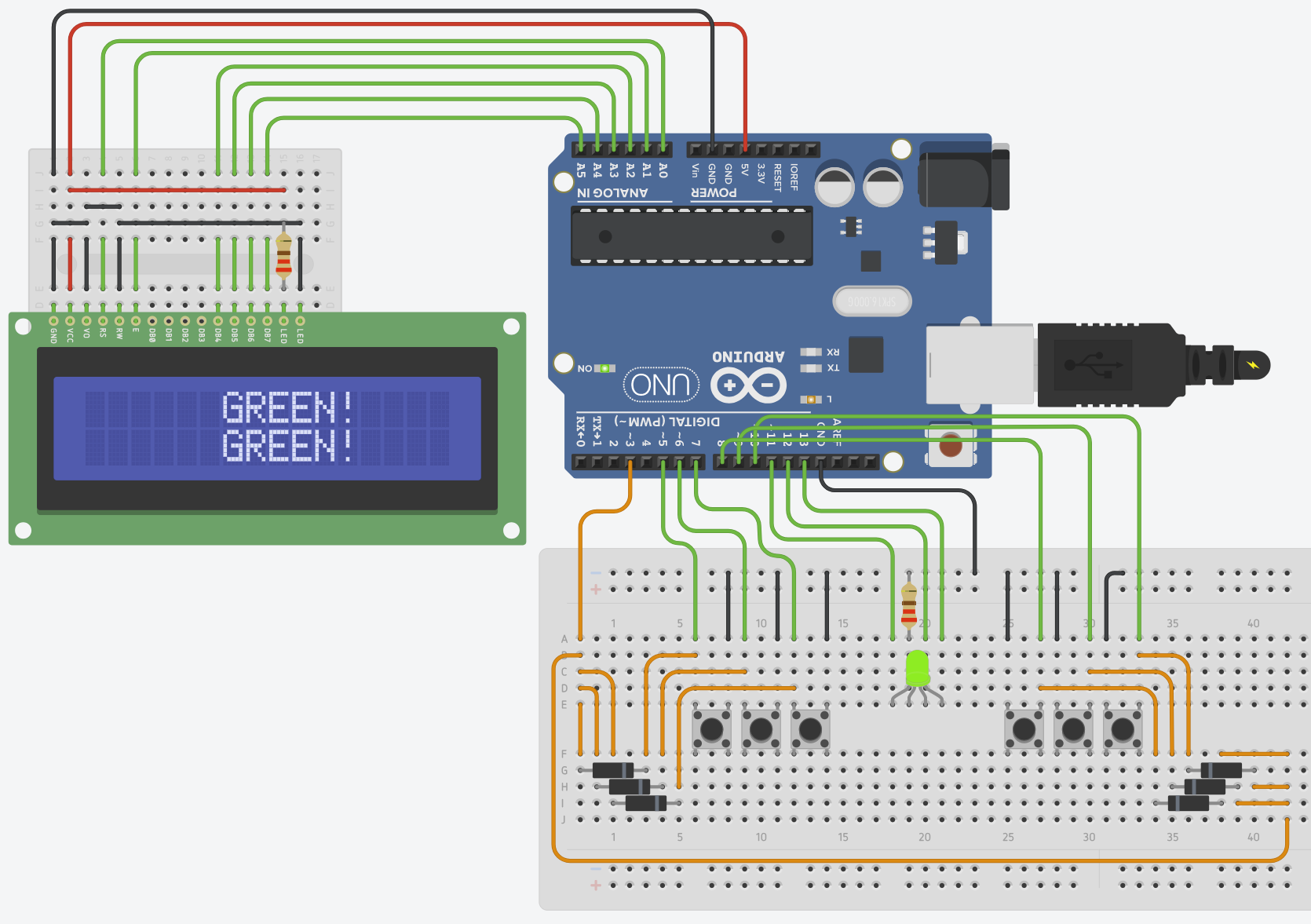
Figur TinkerCad-projekt med ISR och dioder

Kod för testet återges i [Bilaga V: Testkod för ISR och dioder med TinkerCad](#_Bilaga_V:_Testkod).

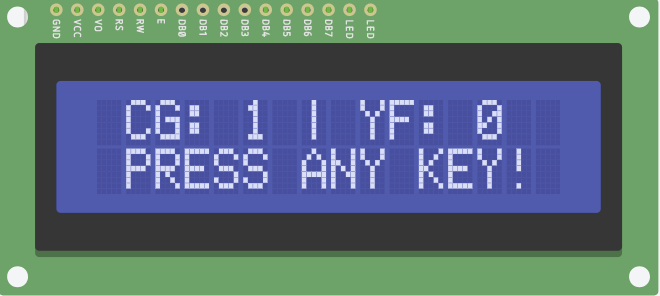
Testet fungerade, vilket innebär att reaction-game kunde simuleras i TinkerCad:



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



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

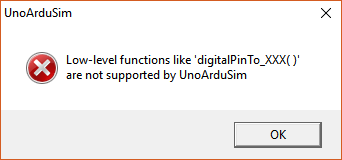


Figur Simulering av react-game i TinkerCad - LCD-Display: Idle-läge, visar poängställning

## 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.



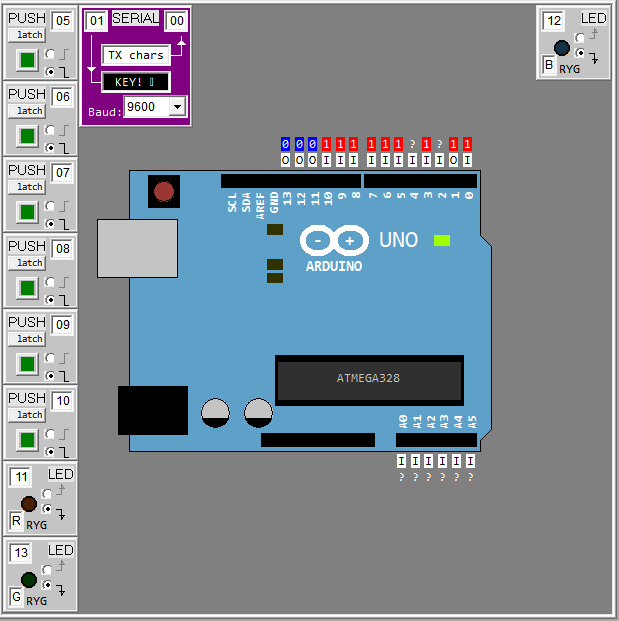
Figur Felmeddelande i UnoArduSim, för avsaknad av stöd av lågnivå-funktioner

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

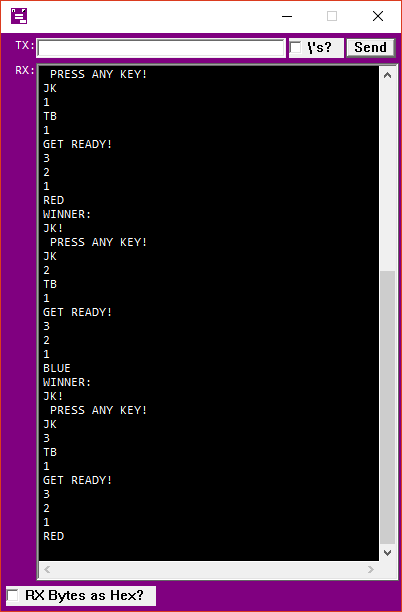
|  |  |
| --- | --- |
| Fel / avsaknad av stöd | Åtgärd |
| Initiering av arraystorlekar kunde ej göras med macron. Exempel: int array[NUM] | Byte till hårdkodade heltal. Exempel:  **int array[2]** |
| Avsaknad av stöd för ISR | Anropning av ISR manuellt innan varje kontroll om en knapp har blivit nedtryckt    Figur 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](#_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 Komponenter i UnoArduSim



Figur 29 Serial Monitor i UnoArduSim

En demonstrationsvideo av simuleringen i UnoArduSim kan ses på Youtube, se [YouTube-länkar](#_YouTube-länkar).

# Diskussion och slutsats

## 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.

## 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.

## Simulering

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

# 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++)

{

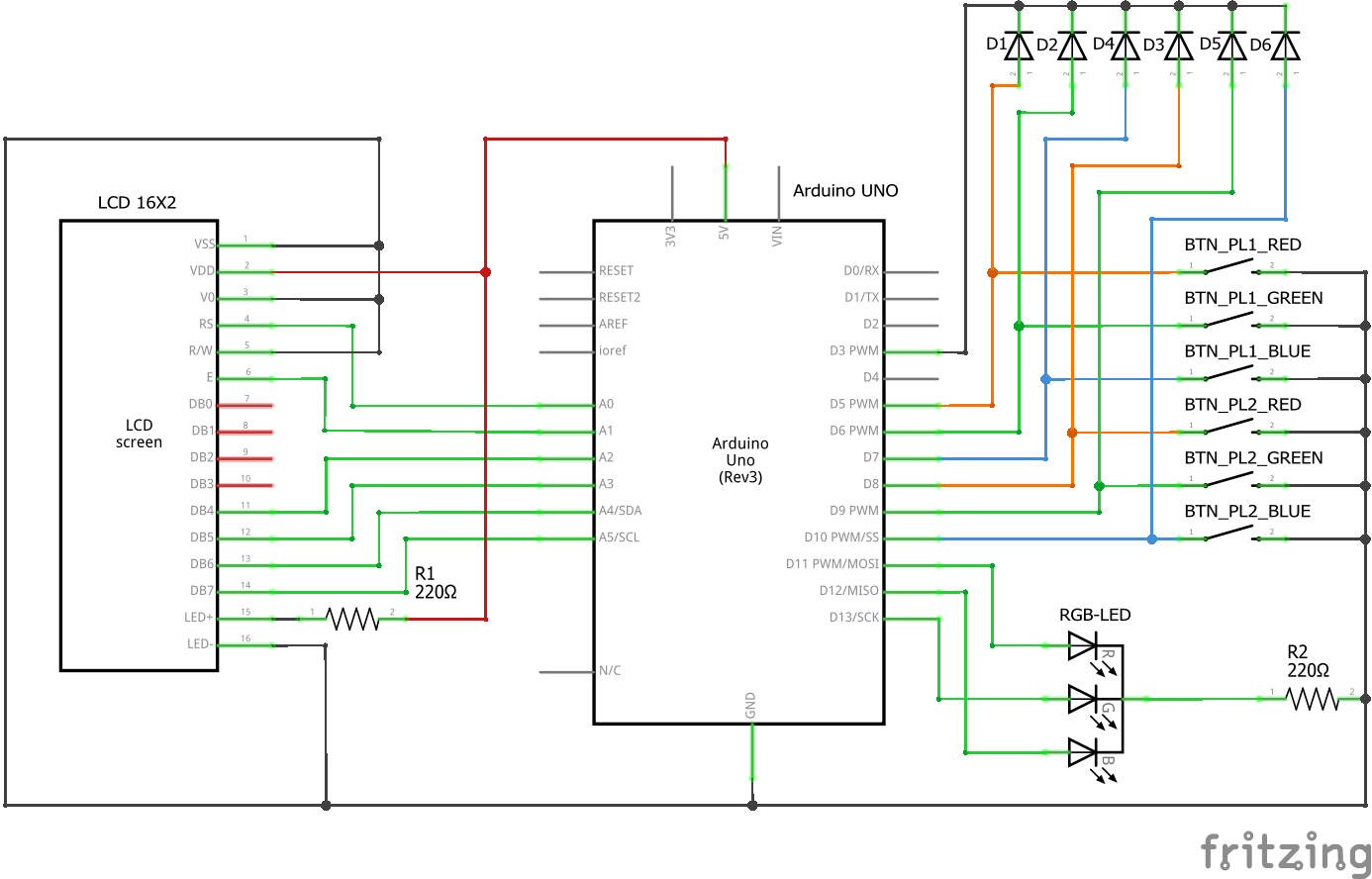
digitalWrite(rgb\_led\_pin[i], LOW);

}

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);

}