Prototype - notat

Tittel: Iterasjon 5	
Forfattere: Prosjekt - gruppe 2	
Versjon: 1.0	Dato: 28.11.2024

Innhold

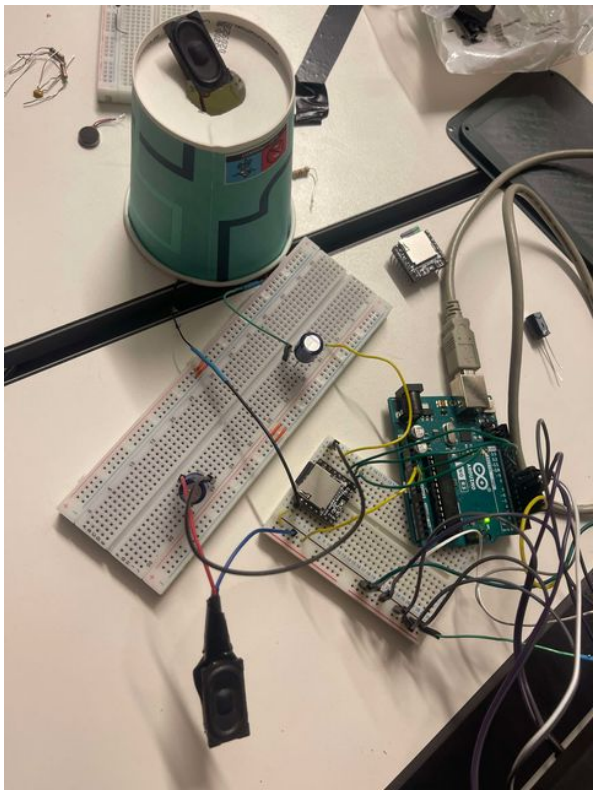
---

1 Introduksjon

Iterasjonen bygger videre på iterasjon 4. Denne iterasjonen fokuserer på å implementere høypassfilter, en ekstra høyttaler og en mer stimulerende brukeropplevelse. Det skal implementeres en vibrator som skal dure 2 ganger ved feil typing, og 1 gang dersom svaret er riktig.

2 Metode og testmiljø

En kondensator på 220 µF er satt i serie med høyttaleren for å blokkere DC komponenten på utgangen. Frekvensen må over 226Hz for at utgangssignalet ikke skal være dempet av betydning. Dette resulterer i at lavere frekvenser slik som DC komponenten og lydsignal som tilsvarer mørkere basstoner ikke slipper igjennom. En vibrasjonsmodul er festet på innsiden av casen og kodet til å vibrere ved feil/rett input fra bruker. Bilde fra testing kan sees under:



Figur 1: Testing av oppsett

3 Test resultat

Filteret fungerer som tiltenkt og blokkerer DC komponenten og har ingen hørbar forskjell på lydvaliteten. En høyttaler til med samme filteroppsett er implementert på utgang nr 2 på DFplayermini. Dette løftet lydopplevelsen ytterligere og ga mere dybde til lydopplevelsen. Det oppleves også mer stimulerende og engasjerende med en ekstra høyttaler, samt vibrering som bekreftelse eller avkreftelse. Vibrasjonsmodulen er festet på innsiden av casen i taket, slik at den er mest mulig i ¨kontakt¨ med fingrene/hånda som ligger an mot casen.

4 Begrensninger og tiltak

Boksen som skal huse elektronikken og være brukervennlig for barn er ikke ferdig montert. Den må være lukket og robust med en lett tilgjengelig av/på knapp. Denne skal i utgangspunktet bare monteres, men siden den ikke er ferdig 3d-printet ligger casen under begrensninger. Vibrasjonsmodulen må festes på en bedre måte, slik den er montert er det stor sjanse for at den rister løs.

5 Konklusjon

De implementerte lavpassfiltrene fungerte som tiltenkt. Den konstante spenningen som tidligere lå på utgangen slipper ikke gjennom kondensatoren. Det er også implementert en ekstra fungerende høyttaler med suksess. Begge er omfattende testet og ingen av de går varme. Vibrasjonsmpdulen er testet og fungerer stabilt etter sin hensikt. To kjappe vibreringer for feil og en kort vibrering for rett.

6 Vedlegg

Kode for iterasjon 5:

Main fil:

Listing 1: kode for iterasjon 5: main

```
#include <DFRobotDFPlayerMini.h>
#include <Arduino.h>
#include "animals_braille.h"

void setup() {
  Serial.begin(115200);
  initialize_buttons();
  pinMode(vibrate_pin, OUTPUT);
  setup_blind_keys_audio();
  blind_keys_er_klar(); //Audio: klar til bruk
  Serial.println("Programmet er klart");
  delay(1000);
}

void loop() {
  //debug_blind_keys_audio();
  switch (letter) {
    case 0:
      //velg_program();
      delay(1000);
      wait_for_start_alphabet();
      wait_for_start_animals();
      break;
    case 1: //Start alfabet
      Serial.println("Trykk inn for A:");
      spill_av_a();
      delay(1000);
      check_buttons_A();
      break;
    case 2:
      Serial.println("Trykk inn for B:");
      spill_av_b();
      delay(1000);
      check_buttons_B();
```

```
      break;
    case 3:
      Serial.println("Trykk inn for C:");
      spill_av_c();
      delay(1000);
      check_buttons_C();
      break;
    case 4: // Alfabet ferdig, nullstiller
      Serial.println("Du er ferdig, godt jobbet!");
      du_er_ferdig();
      letter = 0;
      delay(1000);
      break;
    case 5: //Start dyreprogram
      Serial.println("Hvilket dyr er dette?");
      hvilket_dyr_er_dette();
      delay(2000);
      monkey_sound();
      delay(2000);
      animal_sounds_monkey();
      break;
    case 6:
      Serial.println("Hvilket dyr er dette?");
      hvilket_dyr_er_dette();
      delay(2000);
      cow_sound();
      delay(2000);
      animal_sounds_cow();
      break;
    case 7: //dyreprogram ferdig, nullstiller
      Serial.println("Du er ferdig, godt jobbet!");
      du_er_ferdig();
      letter = 0;
      delay(1000);
      break;
    default:
      break;
  }
}
```

Header fil for dyre funksjon:

Listing 2: kode for iterasjon 5: dyrer

```

#include "buttons_braille.h"

void wait_for_start_animals(){
    if (!is_button_pressed(buttonPins[5])) { // Sjekker siste knapp (
        pin 7)
    startknapp_trykket_dyreprogram();
    Serial.println("Dyreprogram valgt, programmet begynner");
    delay(3000); // For aa gi litt pusterom
```

```
    letter = 5; // Bytt til neste case
  }
}

void animal_sounds_monkey(){
    // Definer hvilke knapper som skal vaere trykket (1) og ikke
       trykket (0)
  const bool expectedState[NUM_BUTTONS] = {true, true, false, true,
     true, true}; // Tilpass til kravene for A

  bool allCorrect = true;

  for (int i = 0; i < NUM_BUTTONS; i++) {
    // Sjekk om hver knapp er i forventet tilstand
    if (is_button_pressed(buttonPins[i]) != expectedState[i]) {
      allCorrect = false; // Hvis en knapp ikke er i riktig tilstand
      break; // Avslutt sjekk tidlig
    }
  }

  if (allCorrect) {
    riktig_bra_jobba_dyr();
    vibrate_right();
    delay(1800);
    Serial.println("Riktig knapp trykket! Gaar videre til neste dyr");
    letter = 6;
  } else {
    feil_prov_igjen();
    delay(1400);
    vibrate_wrong();
    Serial.println("Proov igjen!");
    letter = 5;
  }
}

void animal_sounds_cow(){
    // Definer hvilke knapper som skal vaere trykket (1) og ikke
       trykket (0)
  const bool expectedState[NUM_BUTTONS] = {false, true, false, true,
     true, true}; // Tilpass til kravene for A

  bool allCorrect = true;

  for (int i = 0; i < NUM_BUTTONS; i++) {
    // Sjekk om hver knapp er i forventet tilstand
    if (is_button_pressed(buttonPins[i]) != expectedState[i]) {
      allCorrect = false; // Hvis en knapp ikke er i riktig tilstand
      break; // Avslutt sjekk tidlig
    }
  }

  if (allCorrect) {
    riktig_bra_jobba_dyr();
```

```
    vibrate_right();
    delay(1800);
    Serial.println("Riktig knapp trykket! Gaar videre til neste dyr");
    letter = 7;
  } else {
    feil_prov_igjen();
    delay(1400);
    vibrate_wrong();
    Serial.println("Proov igjen!");
    letter = 6;
  }
}
```

Header fil for alfabet funksjon:

Listing 3: kode for iterasjon 5: alfabet

```

#include "blind_keys_audio.h"
// Definering av bokstaver
int8_t letter = 0;

// Definering av knapper
#define NUM_BUTTONS 6 // Antall
//#define NUM_VIBRATORS 1
const uint8_t buttonPins[NUM_BUTTONS] = {2, 3, 4, 5, 6, 7}; // Pins for knapper
//const uint8_t vibrator[NUM_VIBRATORS] = {}

const uint8_t vibrate_pin = 8;

void initialize_buttons();
bool is_button_pressed(uint8_t pin);
void wait_for_start();
void check_buttons_A();
void check_buttons_B();

// Initialisere alle knappene som pullup
void initialize_buttons() {
  for (int i = 0; i < NUM_BUTTONS; i++) {
    pinMode(buttonPins[i], INPUT_PULLUP);
  }
}

// Funksjon for aa sjekke om en knapp er trykket
bool is_button_pressed(uint8_t pin) {
  return digitalRead(pin) == LOW; // Lav betyr at knappen er trykket
}

// Funksjon for aa starte programmet
void wait_for_start_alphabet() {
  if (!is_button_pressed(buttonPins[0])) { // Sjekker forste knapp (pin 2)
```

```
35      startknapp_trykket();
36      Serial.println("Startknapp trykket, programmet begynner");
37      delay(3000); // For aa gi litt pusterom
38      letter = 1; // Bytt til neste case
39    }
40  }
41
42  void vibrate_right(){
43      digitalWrite(vibrate_pin, HIGH);
44      delay(200);
45      digitalWrite(vibrate_pin, LOW);
46  }
47
48  void vibrate_wrong(){
49      digitalWrite(vibrate_pin, HIGH);
50      delay(100);
51      digitalWrite(vibrate_pin, LOW);
52      delay(100);
53      digitalWrite(vibrate_pin, HIGH);
54      delay(100);
55      digitalWrite(vibrate_pin, LOW);
56  }
57
58
59  // Funksjon for aa sjekke A
60  void check_buttons_A() {
61    // Definer hvilke knapper som skal vaere trykket (1) og ikke trykket
        (0)
62    const bool expectedState[NUM_BUTTONS] = {true, true, false, true,
        true, true}; // Tilpass til kravene for A
63
64    bool allCorrect = true;
65
66    for (int i = 0; i < NUM_BUTTONS; i++) {
67      // Sjekk om hver knapp er i forventet tilstand
68      if (is_button_pressed(buttonPins[i]) != expectedState[i]) {
69        allCorrect = false; // Hvis en knapp ikke er i riktig tilstand
70        break; // Avslutt sjekk tidlig
71      }
72    }
73
74    if (allCorrect) {
75      riktig_bra_jobba();
76      vibrate_right();
77      delay(1800);
78      Serial.println("Riktig knapp trykket! Gaar videre til B");
79      letter = 2;
80    } else {
81      feil_prov_igjen();
82      delay(1400);
83      vibrate_wrong();
84      Serial.println("Feil, proov igjen!");
85      letter = 1;
```

```
  }

  delay(1000); // Unngaa raske gjentakelser
}

// Funksjon for aa sjekke B
void check_buttons_B() {
  // Definer hvilke knapper som skal vaere trykket (1) og ikke trykket (0)
  const bool expectedState[NUM_BUTTONS] = {true, false, false, true, true, true};

  bool allCorrect = true;

  for (int i = 0; i < NUM_BUTTONS; i++) {
    // Sjekk om hver knapp er i forventet tilstand
    if (is_button_pressed(buttonPins[i]) != expectedState[i]) {
      allCorrect = false; // Hvis en knapp ikke er i riktig tilstand
      break; // Avslutt sjekk tidlig
    }
  }

  if (allCorrect) {
    riktig_bra_jobba();
    vibrate_right();
    delay(1800);
    Serial.println("Riktige knapper trykket! gaar videre til C");
    letter = 3;
  } else {
    feil_prov_igjen();
    delay(1400);
    vibrate_wrong();
    Serial.println("Feil, prov igjen!");
    letter = 2;
  }

  delay(1000); // Unngaa raske gjentakelser
}


// Funksjon for aa sjekke C
void check_buttons_C() {
  // Definer hvilke knapper som skal vaere trykket (1) og ikke trykket (0)
  const bool expectedState[NUM_BUTTONS] = {true, true, false, false, true, true}; // Tilpass til kravene for C

  bool allCorrect = true;

  for (int i = 0; i < NUM_BUTTONS; i++) {
    // Sjekk om hver knapp er i forventet tilstand
    if (is_button_pressed(buttonPins[i]) != expectedState[i]) {
      allCorrect = false; // Hvis en knapp ikke er i riktig tilstand
```

```
      break; // Avslutt sjekk tidlig
    }
  }

  if (allCorrect) {
    riktig_bra_jobba();
    vibrate_right();
    delay(1800);
    Serial.println("Riktige knapper trykket! gaar videre til D");
    letter = 4;
  } else {
    feil_prov_igjen();
    delay(1400);
    vibrate_wrong();
    Serial.println("Feil, prov igjen!");
    letter = 3;
  }

  delay(1000); // Unngaa raske gjentakelser
}
```

Header fil for audio:

Listing 4: kode for iterasjon 5: audio

```

//Definisjoner start blind_keys_audio
#if (defined(ARDUINO_AVR_UNO) || defined(ESP8266))   // Using a soft
    serial port
#include <SoftwareSerial.h>
SoftwareSerial softSerial(/*rx =*/10, /*tx =*/11);
#define FPSerial softSerial
#else
#define FPSerial Serial1
#endif

DFRobotDFPlayerMini blind_keys_audio;
void printDetail(uint8_t type, int value);
//Definisjoner stopp blind_keys_audio

//Setup for blink_keys_audio
void setup_blind_keys_audio(){

 #if (defined ESP32)
 FPSerial.begin(9600, SERIAL_8N1, /*rx =*/D3, /*tx =*/D2);
#else
 FPSerial.begin(9600);
#endif

 Serial.begin(115200);

 Serial.println();
 Serial.println(F("DFRobot DFPlayer Mini Demo"));
```

```
28  Serial.println(F("Starter blind_keys_audio"));
29
30  if (!blind_keys_audio.begin(FPSerial, /*isACK = */true, /*doReset =
        */true)) {  //Use serial to communicate with mp3.
31    Serial.println(F("Unable to begin:"));
32    Serial.println(F("1.Sjekk koblinger"));
33    Serial.println(F("2.Sett inn minnekort!"));
34    while(true){
35      delay(0); // Code to compatible with ESP8266 watch dog.
36    }
37  }
38  Serial.println(F("blind_keys_audio klar til bruk"));
39
40  blind_keys_audio.volume(30);  //Set volume value. From 0 to 30
41
42 }
43
44 //Lyder for bokstaver
45 void spill_av_a(){
46  blind_keys_audio.play(8);
47 }
48
49 void spill_av_b(){
50  blind_keys_audio.play(10);
51 }
52
53 void spill_av_c(){
54  blind_keys_audio.play(12);
55 }
56
57 void startknapp_trykket(){
58   blind_keys_audio.play(9);
59 }
60
61
62 // Generelle lyder
63 void feil_prov_igjen(){
64   blind_keys_audio.play(19);
65 }
66
67 void riktig_bra_jobba(){
68   blind_keys_audio.play(21);
69 }
70
71 void blind_keys_er_klar(){
72   blind_keys_audio.play(11);
73 }
74
75 void du_er_ferdig(){
76   blind_keys_audio.play(14);
77 }
78
79 void velg_program(){
```

```
  blind_keys_audio.play(60); //mangler denne
}


// Dyreprogram lyder
void startknapp_trykket_dyreprogram(){
  blind_keys_audio.play(15);
}

void hvilket_dyr_er_dette(){
  blind_keys_audio.play(17);
}

void riktig_bra_jobba_dyr(){
  blind_keys_audio.play(16);
}

void monkey_sound(){
  blind_keys_audio.play(18);
}

void cow_sound(){
  blind_keys_audio.play(13);
}
//Avspillingsfunksjoner slutt



//Funksjon for debugging, legges i loop
void debug_blind_keys_audio(){

  if (blind_keys_audio.available()) {
   printDetail(blind_keys_audio.readType(), blind_keys_audio.read());
       //Print the detail message from DFPlayer to handle different
       errors and states.
 }
}
void printDetail(uint8_t type, int value){
 switch (type) {
   case TimeOut:
     Serial.println(F("Time Out!"));
     break;
   case WrongStack:
     Serial.println(F("Stack Wrong!"));
     break;
   case DFPlayerCardInserted:
     Serial.println(F("Card Inserted!"));
     break;
   case DFPlayerCardRemoved:
     Serial.println(F("Card Removed!"));
     break;
   case DFPlayerCardOnline:
     Serial.println(F("Card Online!"));
```

```
      break;
    case DFPlayerUSBInserted:
      Serial.println("USB␣Inserted!");
      break;
    case DFPlayerUSBRemoved:
      Serial.println("USB␣Removed!");
      break;
    case DFPlayerPlayFinished:
      Serial.print(F("Number:"));
      Serial.print(value);
      Serial.println(F("␣Play␣Finished!"));
      break;
    case DFPlayerError:
      Serial.print(F("DFPlayerError:"));
      switch (value) {
        case Busy:
          Serial.println(F("Card␣not␣found"));
          break;
        case Sleeping:
          Serial.println(F("Sleeping"));
          break;
        case SerialWrongStack:
          Serial.println(F("Get␣Wrong␣Stack"));
          break;
        case CheckSumNotMatch:
          Serial.println(F("Check␣Sum␣Not␣Match"));
          break;
        case FileIndexOut:
          Serial.println(F("File␣Index␣Out␣of␣Bound"));
          break;
        case FileMismatch:
          Serial.println(F("Cannot␣Find␣File"));
          break;
        case Advertise:
          Serial.println(F("In␣Advertise"));
          break;
        default:
          break;
      }
      break;
    default:
      break;
  }

}
```