Prototype - notat

Tittel: Iterasjon 4	
Forfattere: Prosjekt - gruppe 2	
Versjon: 1.0	Dato: 25.11.2024

Innhold

---

1 Introduksjon

I denne iterasjonen bygger vi videre på iterasjon 3. Her tar vi utgangspunkt i hvordan den siste iterasjon er formet, men endrer enkelte elementer samt funksjoner hos den. Hensikten med denne iterasjonen er å få tilnærmet et ferdig produkt, det vil si at det kan være såpass ferdig at den kan leveres til evt. brukere for testing. Følgende funksjoner er implementert med kode:

- Alfabet læring (A, B, C)
- Læring av dyrenavn (Ape, Ku)

Samt er det utformet en bedre ergonomisk boks som gjør at det er lettere for yngre brukere å ta i bruk verktøyet vårt. Det er også blitt byttet ut audio, dette for å fremme engasjement og motivasjon. Til slutt har vi implementert en batteriboks som gjør at man kan bruke denne uten å ha en pc eller noe kabler tilkoblet.

2 Metode og testmiljø

For å endre audion tok vi i bruk open-source plattformer som hadde bedre og mer realistiske menneske stemmer.

Måten prototypen ble testet er at vi selv tok på oss blind for øyene og prøvde deretter å bruke *Blindkeys* uten å se, på denne måten kan vi få svar på om denne kan bli brukt av ønsket målgruppe.

3 Test resultat

Resultatet ble som tenkt, nemlig at dette funket greit selv om man ikke kunne se hva en gjorde. Audio outputen ga en god tilbakemelding hvorvidt man får det til, og dersom man gjør feil. Ved å trykke på knappene kan man og bytte mellom de forskjellige modusene, disse er da enten å lære seg alfabetet, eller dyre modusen. Denne modusen går ut på at det spilles en dyrelyd, og deretter skal man skrive inn forbokstaven til det dyret man hører.

4 Begrensninger og tiltak

Etter omfattende testing av høytallersystemet kom det frem at DFplayeren sender ut en konstant spenning på ca 2.5v på utgangen. Denne vedvarende spenningen overopphetet og ødela høytalerelementet siden det gikk opp i limingen pga varme. Dette må løses i neste iterasjon. Det skal implementes et lavpassfilter for å fjerne den konstante DC komponenten på utgangen til DFplayeren for å unngå å ødelegge flere høyttalere. Lydopplevelsen fra høyttaleren er bra, men det burde sees på å sette inn en ekstra høyttaler for å løfte lydkvaliteten og opplevelsen ytterligere.

5 Konklusjon

Det kan konkluderes med at det er implementert et fungerende program for læring av alfabetet. Iterasjonen integrerte bokstavene a,b og c og feedback med audio fungerer etter hensikt. Det er og implementert et enkelt dyreprogram som er gjort mere engasjerende med å legge inn lyder fra dyr som stemmer overens med forbokstaven til dyret. Videre iterasjon må fokusere på å fremme engasjement ytterligere samt legge inn en løsning som forhindrer at flere høyttalere ødelegges samt implementere enda en høyttaler for å løfte kvaliteten og lydopplevelsen.

6 Vedlegg

Kode for iterasjon 4:

Main fil:

Listing 1: kode for iterasjon 4: main

```

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

void setup() {
  Serial.begin(115200);
  initialize_buttons();
  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 dyrefunksjonen:

Listing 2: kode for iterasjon 4: dyr

```
#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(2500); // 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();
    delay(1800);
    Serial.println("Riktig knapp trykket! Gaar videre til neste dyr");
    letter = 6;
  } else {
    feil_prov_igjen();
    delay(1800);
    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();
    delay(2000);
    Serial.println("Riktig knapp trykket! Gaar videre til neste dyr");
    letter = 7;
  } else {
    feil_prov_igjen();
    delay(2000);
    Serial.println("Proov igjen!");
    letter = 6;
  }
}
```

Header fil for alfabet funksjonen:

Listing 3: kode for iterasjon 4: alfabet

```

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

```
5
6  // Definering av knapper
7  #define NUM_BUTTONS 6 // Antall knapper
8  const uint8_t buttonPins[NUM_BUTTONS] = {2, 3, 4, 5, 6, 7}; // Pins
       for knapper
9
10 void initialize_buttons();
11 bool is_button_pressed(uint8_t pin);
12 void wait_for_start_alphabet();
13 void wait_for_start_animals();
14 void check_buttons_A();
15 void check_buttons_B();
16
17 // Initialisere alle knappene som pullup
18 void initialize_buttons() {
19   for (int i = 0; i < NUM_BUTTONS; i++) {
20     pinMode(buttonPins[i], INPUT_PULLUP);
21   }
22 }
23
24 // Funksjon for aa sjekke om en knapp er trykket
25 bool is_button_pressed(uint8_t pin) {
26   return digitalRead(pin) == LOW; // Lav betyr at knappen er trykket
27 }
28
29 // Funksjon for aa starte programmet
30 void wait_for_start_alphabet() {
31   if (!is_button_pressed(buttonPins[0])) { // Sjekker foorste knapp (
       pin 2)
32     startknapp_trykket();
33     Serial.println("Alfabet␣valgt,␣programmet␣begynner");
34     delay(2500); // For aa gi litt pusterom
35     letter = 1; // Bytt til neste case
36   }
37 }
38
39 // Funksjon for aa sjekke A
40 void check_buttons_A() {
41   // Definer hvilke knapper som skal vaere trykket (1) og ikke trykket
       (0)
42   const bool expectedState[NUM_BUTTONS] = {true, true, false, true,
       true, true}; // Tilpass til kravene for A
43
44   bool allCorrect = true;
45
46   for (int i = 0; i < NUM_BUTTONS; i++) {
47     // Sjekk om hver knapp er i forventet tilstand
48     if (is_button_pressed(buttonPins[i]) != expectedState[i]) {
49       allCorrect = false; // Hvis en knapp ikke er i riktig tilstand
50       break; // Avslutt sjekk tidlig
51     }
52   }
53
```

```
54    if (allCorrect) {
55      riktig_bra_jobba();
56      delay(1800);
57      Serial.println("Riktig knapp trykket! Gaar videre til B");
58      letter = 2;
59    } else {
60      feil_proov_igjen();
61      delay(1800);
62      Serial.println("Feil, proov igjen!");
63      letter = 1;
64    }
65
66    delay(1000); // Unngaa raske gjentakelser
67  }
68
69  // Funksjon for aa sjekke B
70  void check_buttons_B() {
71    // Definer hvilke knapper som skal vaere trykket (1) og ikke trykket
          (0)
72    const bool expectedState[NUM_BUTTONS] = {true, false, false, true,
         true, true};
73
74    bool allCorrect = true;
75
76    for (int i = 0; i < NUM_BUTTONS; i++) {
77      // Sjekk om hver knapp er i forventet tilstand
78      if (is_button_pressed(buttonPins[i]) != expectedState[i]) {
79        allCorrect = false; // Hvis en knapp ikke er i riktig tilstand
80        break; // Avslutt sjekk tidlig
81      }
82    }
83
84    if (allCorrect) {
85      riktig_bra_jobba();
86      delay(1800);
87      Serial.println("Riktige knapper trykket! gaar videre til C");
88      letter = 3;
89    } else {
90      feil_proov_igjen();
91      delay(1800);
92      Serial.println("Feil, proov igjen!");
93      letter = 2;
94    }
95
96    delay(1000); // Unngaa raske gjentakelser
97  }
98
99  // Funksjon for aa sjekke C
100 void check_buttons_C() {
101   // Definer hvilke knapper som skal vaere trykket (1) og ikke trykket
          (0)
102   const bool expectedState[NUM_BUTTONS] = {true, true, false, false,
         true, true}; // Tilpass til kravene for C
```

```
103
104   bool allCorrect = true;
105
106   for (int i = 0; i < NUM_BUTTONS; i++) {
107     // Sjekk om hver knapp er i forventet tilstand
108     if (is_button_pressed(buttonPins[i]) != expectedState[i]) {
109       allCorrect = false; // Hvis en knapp ikke er i riktig tilstand
110       break; // Avslutt sjekk tidlig
111     }
112   }
113
114   if (allCorrect) {
115     riktig_bra_jobba();
116     delay(1800);
117     Serial.println("Riktige knapper trykket! gaar videre til D");
118     letter = 4;
119   } else {
120     feil_proov_igjen();
121     delay(1800);
122     Serial.println("Feil, proov igjen!");
123     letter = 3;
124   }
125
126   delay(1000); // Unngaa raske gjentakelser
127 }
128
129 void done_alphabet(){
130   //spille av "ferdig audio"
131   letter = 0;
132 }
```

Header fil for audio

Listing 4: kode for iterasjon 4: audio

```
1
2  //Definisjoner start blind_keys_audio
3  #if (defined(ARDUINO_AVR_UNO) || defined(ESP8266))   // Using a soft
       serial port
4  #include <SoftwareSerial.h>
5  SoftwareSerial softSerial(/*rx =*/10, /*tx =*/11);
6  #define FPSerial softSerial
7  #else
8  #define FPSerial Serial1
9  #endif
10
11 DFRobotDFPlayerMini blind_keys_audio;
12 void printDetail(uint8_t type, int value);
13 //Definisjoner stopp blind_keys_audio
14
15 //Setup for blink_keys_audio
16 void setup_blind_keys_audio(){
17
```

```
 #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"));
 Serial.println(F("Starter blind_keys_audio"));

 if (!blind_keys_audio.begin(FPSerial, /*isACK = */true, /*doReset =
     */true)) {  //Use serial to communicate with mp3.
   Serial.println(F("Unable to begin:"));
   Serial.println(F("1.Sjekk koblinger"));
   Serial.println(F("2.Sett inn minnekort!"));
   while(true){
     delay(0); // Code to compatible with ESP8266 watch dog.
   }
 }
 Serial.println(F("blind_keys_audio klar til bruk"));

 blind_keys_audio.volume(30);  //Set volume value. From 0 to 30

}

//Lyder for bokstaver
void spill_av_a(){
 blind_keys_audio.play(8);
}

void spill_av_b(){
 blind_keys_audio.play(10);
}

void spill_av_c(){
 blind_keys_audio.play(12);
}

void startknapp_trykket(){
  blind_keys_audio.play(9);
}


// Generelle lyder
void feil_proov_igjen(){
  blind_keys_audio.play(19);
}

void riktig_bra_jobba(){
  blind_keys_audio.play(21);
}
```

```
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(){
80    blind_keys_audio.play(60); //mangler denne
81  }
82
83
84  // Dyreprogram lyder
85  void startknapp_trykket_dyreprogram(){
86    blind_keys_audio.play(15);
87  }
88
89  void hvilket_dyr_er_dette(){
90    blind_keys_audio.play(17);
91  }
92
93  void riktig_bra_jobba_dyr(){
94    blind_keys_audio.play(16);
95  }
96
97  void monkey_sound(){
98    blind_keys_audio.play(18);
99  }
100
101 void cow_sound(){
102   blind_keys_audio.play(13);
103 }
104 //Avspillingsfunksjoner slutt
105
106
107
108 //Funksjon for debugging, legges i loop
109 void debug_blind_keys_audio(){
110
111   if (blind_keys_audio.available()) {
112    printDetail(blind_keys_audio.readType(), blind_keys_audio.read());
          //Print the detail message from DFPlayer to handle different
          errors and states.
113  }
114 }
115 void printDetail(uint8_t type, int value){
116  switch (type) {
117    case TimeOut:
118      Serial.println(F("Time␣Out!"));
119      break;
120    case WrongStack:
```

```
121       Serial.println(F("Stack␣Wrong!"));
122       break;
123     case DFPlayerCardInserted:
124       Serial.println(F("Card␣Inserted!"));
125       break;
126     case DFPlayerCardRemoved:
127       Serial.println(F("Card␣Removed!"));
128       break;
129     case DFPlayerCardOnline:
130       Serial.println(F("Card␣Online!"));
131       break;
132     case DFPlayerUSBInserted:
133       Serial.println("USB␣Inserted!");
134       break;
135     case DFPlayerUSBRemoved:
136       Serial.println("USB␣Removed!");
137       break;
138     case DFPlayerPlayFinished:
139       Serial.print(F("Number:"));
140       Serial.print(value);
141       Serial.println(F("␣Play␣Finished!"));
142       break;
143     case DFPlayerError:
144       Serial.print(F("DFPlayerError:"));
145       switch (value) {
146         case Busy:
147           Serial.println(F("Card␣not␣found"));
148           break;
149         case Sleeping:
150           Serial.println(F("Sleeping"));
151           break;
152         case SerialWrongStack:
153           Serial.println(F("Get␣Wrong␣Stack"));
154           break;
155         case CheckSumNotMatch:
156           Serial.println(F("Check␣Sum␣Not␣Match"));
157           break;
158         case FileIndexOut:
159           Serial.println(F("File␣Index␣Out␣of␣Bound"));
160           break;
161         case FileMismatch:
162           Serial.println(F("Cannot␣Find␣File"));
163           break;
164         case Advertise:
165           Serial.println(F("In␣Advertise"));
166           break;
167         default:
168           break;
169       }
170       break;
171     default:
172       break;
173   }
```

```
174
175 }
```