Prototype - notat

Tittel: Iterasjon 3	
Forfattere: Prosjekt - gruppe 2	
Versjon: 1.0	Dato: 10.11.2024

Innhold

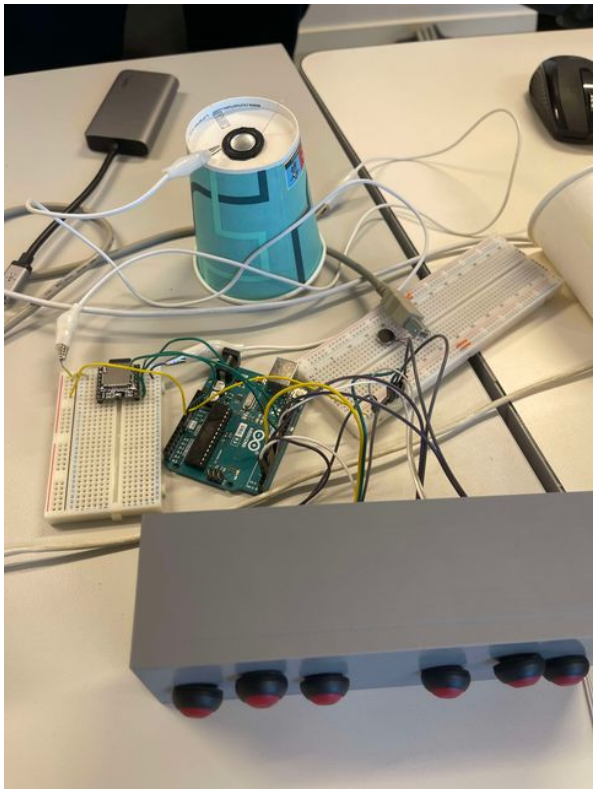
---

1 Introduksjon

Denne prototypen er en utviklet iterasjon fra iterasjon 2. I forrige iterajon benyttet vi oss av AVR128DB48 og lagde en kode for hvordan man kan styre knappene. I denne iterasjonen er fokuset på å få knappene til å virke med audio. For audio er det benyttet DF miniplayer. Da det viste seg at det var ekstremt krevende å få avspilt audio igjennom AVR med denne audiospilleren byttet vi til en Arduino Uno. Dette ga oss mulighet til å spille av lyd som ønsket. Hensikten med denne iterasjon er at det skal gi instrukser i form av audio, samt tilbakemeldinger som kan gi en veiledning for hvordan man skriver inn rett bokstav.

2 Metode og testmiljø

Måten dette ble testet på er at vi lastet opp ferdigstilt kode til Arduino Uno, og deretter sjekket vi hvordan knappene og audio delene funket sammen. Vedlagt bilde av testingen:



3 Test resultat

Testingen viste seg til å være vellyket, Ved å starte opp og trykke inn bestemte knapper fikk man ut den audio filen man ønsket.

4 Begrensninger og tiltak

Ved å se på bilde fra testingen kan man se et improvisert høyttalersystem, dette er noe vi gjorde midlertidig. Det funket greit men er ikke noe som vi kan beholde, dette designet skal bli brukt av barn og en enkel papp kopp kommer ikke til å holde.

En annen aspekt er audio delen, slik den er nå er den ikke veldig motiverende å høre på. Dermed vil et tiltak være å få inn audio som kan virke motiverende og engasjerende.

Samt ønsker vi flere funksjoner som gjør at *BlindKeys* et verktøy som har flere funksjoner, da det kan bli veldig ensidig og kjedelig for brukerne med de samme funskjonene hele tiden.

Designet av boks er ikke ergonomisk, eller tilpasset komponentene. Det er derfor nødt til å designe en ny boks som skal ha komponenter og ledninger inni seg. Samt må det designes slik at små barnehender enkelt kan bruke verktøyet vårt.

5 Konklusjon

Designet virker som ønsket, ved å kombinere audio og bruker input i form av knappetrykk kan man skape et verktøy som gjør det mulig for blinde barn å lære seg blindeskrift.

6 Vedlegg

Kode for iterasjon 3:

Main fil:

Listing 1: kode for iterasjon 3: main

```
#include <DFRobotDFPlayerMini.h>
#include <Arduino.h>
#include "buttons_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:
      wait_for_start();
      break;
    case 1:
      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;
    default:
      // Eventuell videre logikk
      break;
  }
}
```

Header fil for knappene:

Listing 2: kode for iterasjon 3: buttons

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

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

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 a 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 a starte programmet
void wait_for_start() {
  if (!is_button_pressed(buttonPins[0])) { // Sjekker forste knapp (pin 2)
    startknapp_trykket();
    Serial.println("Startknapp trykket, programmet begynner");
    delay(2500); // For a gi litt pusterom
    letter = 1; // Bytt til neste case
  }
}
```

```
/*
void wait_for_start_animals(){
    if (!is_button_pressed(buttonPins[5])) { // Sjekker siste knapp (pin 7)
    startknapp_trykket();
    Serial.println("Startknapp trykket, programmet begynner");
    delay(2500); // For a gi litt pusterom
    letter = 1; // Bytt til neste case
  }
}
*/


// Funksjon for a sjekke A
void check_buttons_A() {
  // Definer hvilke knapper som skal vere 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();
    delay(1800);
    Serial.println("Riktig knapp trykket! Gar videre til B");
    letter = 2;
  } else {
    feil_prov_igjen();
    delay(1800);
    Serial.println("Feil, prov igjen!");
    letter = 1;
  }

  delay(1000); // Unnga raske gjentakelser
}

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

  bool allCorrect = true;
```

```
84
85    for (int i = 0; i < NUM_BUTTONS; i++) {
86      // Sjekk om hver knapp er i forventet tilstand
87      if (is_button_pressed(buttonPins[i]) != expectedState[i]) {
88        allCorrect = false; // Hvis en knapp ikke er i riktig tilstand
89        break; // Avslutt sjekk tidlig
90      }
91    }
92
93    if (allCorrect) {
94      riktig_bra_jobba();
95      delay(1800);
96      Serial.println("Riktige knapper trykket! gar videre til C");
97      letter = 3;
98    } else {
99      feil_prov_igjen();
100     delay(1800);
101     Serial.println("Feil, prov igjen!");
102     letter = 2;
103   }
104
105   delay(1000); // Unnga raske gjentakelser
106 }
107
108
109 // Funksjon for a sjekke C
110 void check_buttons_C() {
111   // Definer hvilke knapper som skal vere trykket (1) og ikke trykket
        (0)
112   const bool expectedState[NUM_BUTTONS] = {true, true, false, false,
        true, true}; // Tilpass til kravene for C
113
114   bool allCorrect = true;
115
116   for (int i = 0; i < NUM_BUTTONS; i++) {
117     // Sjekk om hver knapp er i forventet tilstand
118     if (is_button_pressed(buttonPins[i]) != expectedState[i]) {
119       allCorrect = false; // Hvis en knapp ikke er i riktig tilstand
120       break; // Avslutt sjekk tidlig
121     }
122   }
123
124   if (allCorrect) {
125     riktig_bra_jobba();
126     delay(1800);
127     Serial.println("Riktige knapper trykket! gar videre til D");
128     letter = 4;
129   } else {
130     feil_prov_igjen();
131     delay(1800);
132     Serial.println("Feil, prov igjen!");
133     letter = 3;
134   }
```

```
135
136   delay(1000); // Unnga raske gjentakelser
137 }
```

Header fil for audio:

Listing 3: kode for iterasjon 3: audio

```
1  //Definisjoner start blind_keys_audio
2  #if (defined(ARDUINO_AVR_UNO) || defined(ESP8266))   // Using a soft
       serial port
3  #include <SoftwareSerial.h>
4  SoftwareSerial softSerial(/*rx =*/10, /*tx =*/11);
5  #define FPSerial softSerial
6  #else
7  #define FPSerial Serial1
8  #endif
9
10 DFRobotDFPlayerMini blind_keys_audio;
11 void printDetail(uint8_t type, int value);
12 //Definisjoner stopp blind_keys_audio
13
14
15
16
17 //Setup for blink_keys_audio
18 void setup_blind_keys_audio(){
19
20  #if (defined ESP32)
21  FPSerial.begin(9600, SERIAL_8N1, /*rx =*/D3, /*tx =*/D2);
22 #else
23  FPSerial.begin(9600);
24 #endif
25
26  Serial.begin(115200);
27
28  Serial.println();
29  Serial.println(F("DFRobot␣DFPlayer␣Mini␣Demo"));
30  Serial.println(F("Starter␣blind_keys_audio"));
31
32  if (!blind_keys_audio.begin(FPSerial, /*isACK = */true, /*doReset =
       */true)) {  //Use serial to communicate with mp3.
33    Serial.println(F("Unable␣to␣begin:"));
34    Serial.println(F("1.Sjekk␣koblinger"));
35    Serial.println(F("2.Sett␣inn␣minnekort!"));
36    while(true){
37      delay(0); // Code to compatible with ESP8266 watch dog.
38    }
39  }
40  Serial.println(F("blind_keys_audio␣klar␣til␣bruk"));
41
42  blind_keys_audio.volume(30);  //Set volume value. From 0 to 30
43
```

```
}

//Avspillingsfunksjoner start
void spill_av_a(){
 blind_keys_audio.play(11);
}

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

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

void feil_prov_igjen(){
  blind_keys_audio.play(8);
}

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

void startknapp_trykket(){
  blind_keys_audio.play(13);
}

void blind_keys_er_klar(){
  blind_keys_audio.play(14);
}
//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!"));
```

```
95        break;
96      case DFPlayerCardRemoved:
97        Serial.println(F("Card␣Removed!"));
98        break;
99      case DFPlayerCardOnline:
100       Serial.println(F("Card␣Online!"));
101       break;
102     case DFPlayerUSBInserted:
103       Serial.println("USB␣Inserted!");
104       break;
105     case DFPlayerUSBRemoved:
106       Serial.println("USB␣Removed!");
107       break;
108     case DFPlayerPlayFinished:
109       Serial.print(F("Number:"));
110       Serial.print(value);
111       Serial.println(F("␣Play␣Finished!"));
112       break;
113     case DFPlayerError:
114       Serial.print(F("DFPlayerError:"));
115       switch (value) {
116         case Busy:
117           Serial.println(F("Card␣not␣found"));
118           break;
119         case Sleeping:
120           Serial.println(F("Sleeping"));
121           break;
122         case SerialWrongStack:
123           Serial.println(F("Get␣Wrong␣Stack"));
124           break;
125         case CheckSumNotMatch:
126           Serial.println(F("Check␣Sum␣Not␣Match"));
127           break;
128         case FileIndexOut:
129           Serial.println(F("File␣Index␣Out␣of␣Bound"));
130           break;
131         case FileMismatch:
132           Serial.println(F("Cannot␣Find␣File"));
133           break;
134         case Advertise:
135           Serial.println(F("In␣Advertise"));
136           break;
137         default:
138           break;
139       }
140       break;
141     default:
142       break;
143   }
144
145 }
```