

<h1>Prototype - notat</h1>	
Tittel: Iterasjon 3	
Forfattere: Prosjekt - gruppe 2	
Versjon: 1.0	Dato: 10.11.2024

Innhold

1	Introduksjon	1
2	Metode og testmiljø	2
3	Test resultat	2
4	Begrensninger og tiltak	2
5	Konklusjon	3
6	Vedlegg	3
A	Audio kode	3
B	Main fil	4

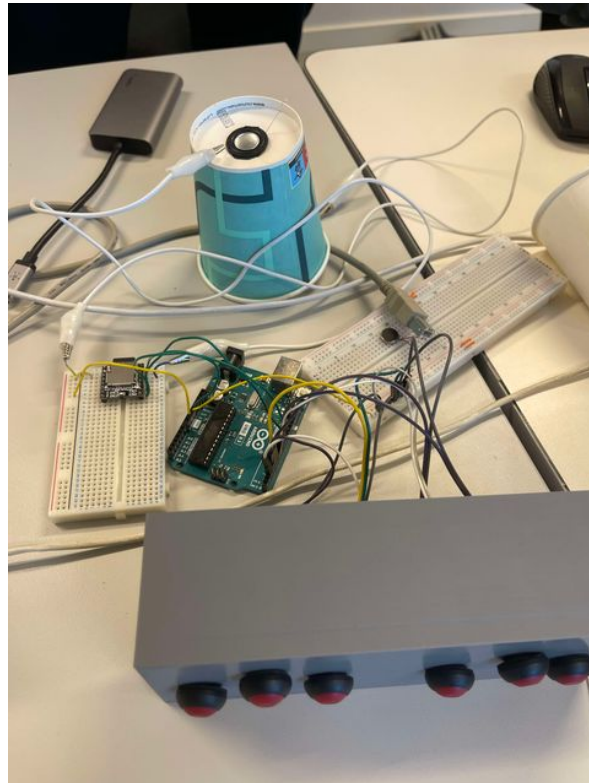
1 Introduksjon

Denne prototypen er en utviklet iterasjon fra iterasjon 2. I forrige iterasjon 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 instruksjoner i form av audio, samt tilbakemeldinger som kan gi en veiledning for hvordan man skriver inn rett bokstav.

2 Metode og testmiljø

For å generere audio brukte vi en python kode for å kunne konvertere tekst til lyd. Koden for audio generering ligger vedlagt her: A.

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



3 Test resultat

Testingen viste seg til å være vellykket, 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 fungerte 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. De genererte audio filene høres veldig robotisk ut. 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 funksjonene 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

A Audio kode

```
1     from gtts import gTTS
2 import os
3
4 # Sette riktig katalog der filene skal lagres
5 directory = "C:/Users/nicho/NTNU/Dokumenter/SKOLE/Ingenior_prosjekt/
6           H2024-prosjekt/Create_audio_files"
7
8 # Sjekk om katalogen finnes
9 if os.path.exists(directory):
10     os.chdir(directory) # Endre til ønsket katalog
11     print(f"Katalogen er nå: {os.getcwd()}")
12 else:
13     print(f"Katalogen finnes ikke: {directory}")
14
15 # Liste med Norske bokstaver
16 norwegian_letters = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J',
17                     'K',
18                     'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U',
19                     'V',
20                     'W', 'X', 'Y', 'Z', 'AE', 'O', 'AA']
21
22 # Lager en separat .wav fil for hver bokstav
23 for letter in norwegian_letters:
24     tts = gTTS(text=letter, lang='no')
25     filename = f"{letter}.mp3"
26     tts.save(filename)
27     print(f"Saved {filename}")
```

B Main fil

Listing 1: kode for iterasjon 3: main

```
1  #include <DFRobotDFPlayerMini.h>
2  #include <Arduino.h>
3  #include "buttons_braille.h"
4
5  void setup() {
6      Serial.begin(115200);
7      initialize_buttons();
8      setup_blind_keys_audio();
9      blind_keys_er_klar(); //Audio: klar til bruk
10     Serial.println("Programmet_er_klart");
11     delay(1000);
12 }
13
14 void loop() {
15     //debug_blind_keys_audio();
16     switch (letter) {
17         case 0:
18             wait_for_start();
19             break;
20         case 1:
21             Serial.println("Trykk_inn_for_A:");
22             spill_av_a();
23             delay(1000);
24             check_buttons_A();
25             break;
26         case 2:
27             Serial.println("Trykk_inn_for_B:");
28             spill_av_b();
29             delay(1000);
30             check_buttons_B();
31             break;
32         case 3:
33             Serial.println("Trykk_inn_for_C:");
34             spill_av_c();
35             delay(1000);
36             check_buttons_C();
37             break;
38         default:
39             // Eventuelt videre logikk
40             break;
41     }
42 }
```

Header fil for knappene:

Listing 2: kode for iterasjon 3: buttons

```
1  #include "blind_keys_audio.h"
2  // Definerer av bokstaver
```

```

3 int8_t letter = 0;
4
5 // Definerer av knapper
6 #define NUM_BUTTONS 6 // Antall knapper
7 const uint8_t buttonPins[NUM_BUTTONS] = {2, 3, 4, 5, 6, 7}; // Pins
   for knapper
8
9 void initialize_buttons();
10 bool is_button_pressed(uint8_t pin);
11 void wait_for_start();
12 void check_buttons_A();
13 void check_buttons_B();
14
15 // Initialisere alle knappene som pullup
16 void initialize_buttons() {
17     for (int i = 0; i < NUM_BUTTONS; i++) {
18         pinMode(buttonPins[i], INPUT_PULLUP);
19     }
20 }
21
22 // Funksjon for a sjekke om en knapp er trykket
23 bool is_button_pressed(uint8_t pin) {
24     return digitalRead(pin) == LOW; // Lav betyr at knappen er trykket
25 }
26
27 // Funksjon for a starte programmet
28 void wait_for_start() {
29     if (!is_button_pressed(buttonPins[0])) { // Sjekker forste knapp (
        pin 2)
30         startknapp_trykket();
31         Serial.println("Startknapp trykket, programmet begynner");
32         delay(2500); // For a gi litt pusterom
33         letter = 1; // Bytt til neste case
34     }
35 }
36 /*
37 void wait_for_start_animals(){
38     if (!is_button_pressed(buttonPins[5])) { // Sjekker siste knapp (
        pin 7)
39         startknapp_trykket();
40         Serial.println("Startknapp trykket, programmet begynner");
41         delay(2500); // For a gi litt pusterom
42         letter = 1; // Bytt til neste case
43     }
44 }
45 */
46
47 // Funksjon for a sjekke A
48 void check_buttons_A() {
49     // Definer hvilke knapper som skal vere trykket (1) og ikke trykket
        (0)
50     const bool expectedState[NUM_BUTTONS] = {true, true, false, true,

```

```

    true, true}; // Tilpass til kravene for A
52
53 bool allCorrect = true;
54
55 for (int i = 0; i < NUM_BUTTONS; i++) {
56     // Sjekk om hver knapp er i forventet tilstand
57     if (is_button_pressed(buttonPins[i]) != expectedState[i]) {
58         allCorrect = false; // Hvis en knapp ikke er i riktig tilstand
59         break; // Avslutt sjekk tidlig
60     }
61 }
62
63 if (allCorrect) {
64     riktig_bra_jobba();
65     delay(1800);
66     Serial.println("Riktig_knapp_trykket!_Gar_videre_til_B");
67     letter = 2;
68 } else {
69     feil_prov_igjen();
70     delay(1800);
71     Serial.println("Feil,_prov_igjen!");
72     letter = 1;
73 }
74
75 delay(1000); // Unnga raske gjentakelser
76 }
77
78 // Funksjon for a sjekke B
79 void check_buttons_B() {
80     // Definer hvilke knapper som skal være trykket (1) og ikke trykket
81     // (0)
82     const bool expectedState[NUM_BUTTONS] = {true, false, false, true,
83         true, true};
84
85     bool allCorrect = true;
86
87     for (int i = 0; i < NUM_BUTTONS; i++) {
88         // Sjekk om hver knapp er i forventet tilstand
89         if (is_button_pressed(buttonPins[i]) != expectedState[i]) {
90             allCorrect = false; // Hvis en knapp ikke er i riktig tilstand
91             break; // Avslutt sjekk tidlig
92         }
93     }
94
95     if (allCorrect) {
96         riktig_bra_jobba();
97         delay(1800);
98         Serial.println("Riktige_knapper_trykket!_gar_videre_til_C");
99         letter = 3;
100     } else {
101         feil_prov_igjen();
102         delay(1800);
103         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     // Definerer hvilke knapper som skal vere trykket (1) og ikke trykket
112     // (0)
113     const bool expectedState[NUM_BUTTONS] = {true, true, false, false,
114         true, true}; // Tilpass til kravene for C
115
116     bool allCorrect = true;
117
118     for (int i = 0; i < NUM_BUTTONS; i++) {
119         // Sjekk om hver knapp er i forventet tilstand
120         if (is_button_pressed(buttonPins[i]) != expectedState[i]) {
121             allCorrect = false; // Hvis en knapp ikke er i riktig tilstand
122             break; // Avslutt sjekk tidlig
123         }
124     }
125
126     if (allCorrect) {
127         riktig_bra_jobba();
128         delay(1800);
129         Serial.println("Riktige_knapper_trykket!_gar_videre_til_D");
130         letter = 4;
131     } else {
132         feil_prov_igjen();
133         delay(1800);
134         Serial.println("Feil,_prov_igjen!");
135         letter = 3;
136     }
137
138     delay(1000); // Unnga raske gjentakelser
139 }

```

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
44 }
45
46 //Avspillingsfunksjoner start
47 void spill_av_a(){
48     blind_keys_audio.play(11);
49 }
50
51 void spill_av_b(){
52     blind_keys_audio.play(12);
53 }
54
55 void spill_av_c(){
56     blind_keys_audio.play(10);
57 }
58
59 void feil_prov_igjen(){
60     blind_keys_audio.play(8);
61 }

```



```

62
63 void riktig_bra_jobba(){
64     blind_keys_audio.play(9);
65 }
66
67 void startknapp_trykket(){
68     blind_keys_audio.play(13);
69 }
70
71 void blind_keys_er_klar(){
72     blind_keys_audio.play(14);
73 }
74 //Avspillingsfunksjoner slutt
75
76
77
78 //Funksjon for debugging, legges i loop
79 void debug_blind_keys_audio(){
80
81     if (blind_keys_audio.available()) {
82         printDetail(blind_keys_audio.readType(), blind_keys_audio.read());
83         //Print the detail message from DFPlayer to handle different
84         errors and states.
85     }
86 }
87 void printDetail(uint8_t type, int value){
88     switch (type) {
89         case TimeOut:
90             Serial.println(F("Time_Out!"));
91             break;
92         case WrongStack:
93             Serial.println(F("Stack_Wrong!"));
94             break;
95         case DFPlayerCardInserted:
96             Serial.println(F("Card_Inserted!"));
97             break;
98         case DFPlayerCardRemoved:
99             Serial.println(F("Card_Removed!"));
100             break;
101         case DFPlayerCardOnline:
102             Serial.println(F("Card_Online!"));
103             break;
104         case DFPlayerUSBInserted:
105             Serial.println(F("USB_Inserted!"));
106             break;
107         case DFPlayerUSBRemoved:
108             Serial.println(F("USB_Removed!"));
109             break;
110         case DFPlayerPlayFinished:
111             Serial.print(F("Number:"));
112             Serial.print(value);
113             Serial.println(F("_Play_Finished!"));
114             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 }

```