ULU.22 – Buzzer

This ULU

The *ULU.22 – Buzzer* can be used as a beeper or simple electronic organ.

Used parts

Only standard parts are used:

1x casing 50 x 25 x 25mm;
2x 2mm signal connector;
2x black O-ring 9 x 5 x 2mm;
1x 4-bit data connector;
1x colored O-ring 8 x 5 x 1.5mm;
1x power connector;
6x 10K pull-up resistor;
1x Arduino Nano.

Construction

The standard ULU specifications are applicable as specified in the datasheet *ULU.00 – Common specifications*.

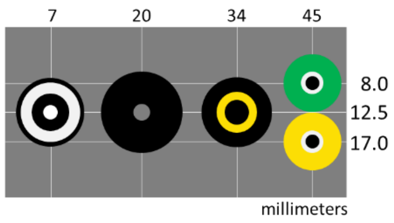

Figure 1 – Drill guide

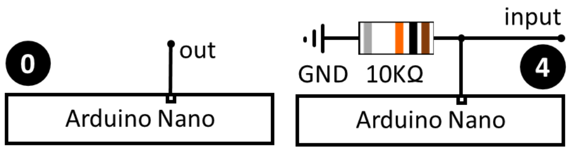

Figure 2 – Used arduino interfaces

	Port	Con.	Rest.	Func.	Interface	Signal
1.	D1		O	L		Heartbeat
2.	D2	🟢		I	❹	Enable (play)
3.	D3	🟡		I	❹	Beep
4.	D4	④		I	❹	Tone 0
5.	D5	④		I	❹	Tone 1
6.	D6	④		I	❹	Tone 2
7.	D7	④		I	❹	Tone 3
8.	D8			O	⓿	Buzzer
9.	D13		O	L		Buzzer
10.	+5V	⊙	I	I	⓿	+5V
11.	GND	⊙	I	I	⓿	GND

Input, **O**utput, **L**ed, **S**PI, **T**oggle switch, **R**otary switch

Figure 3 – Pinout Arduino Nano

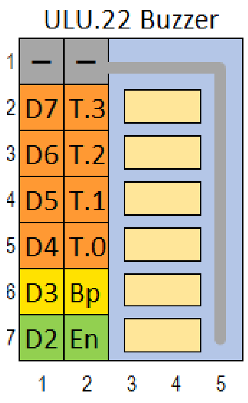

Figure 4 – Layout resistor PCB

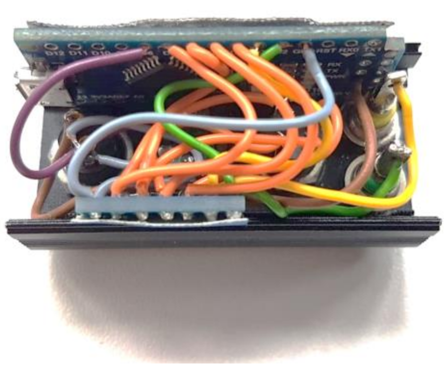

Figure 5 – ULU inside

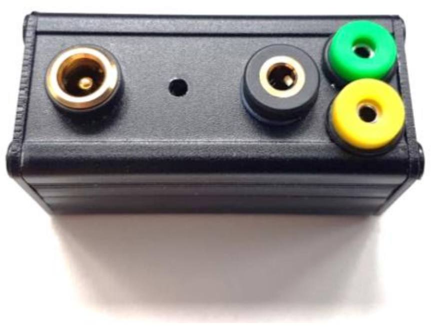

Figure 6 – Finished ULU

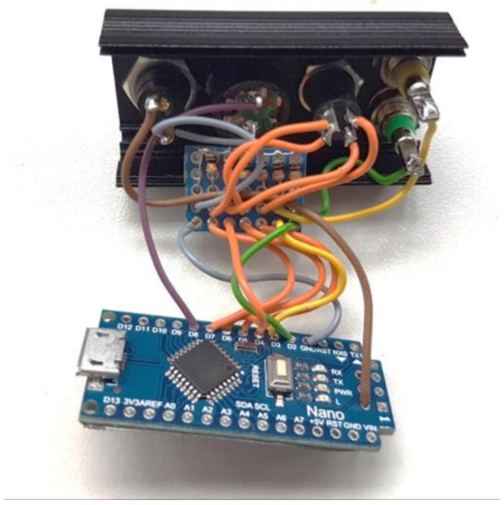
Figure 7 – PCB

Both PCB's are placed at the sides of the casing, facing each other. Tape is used to protect the back sides of both PCB's against unwanted connections. The Arduino Nano will fit into the casing, when a small portion of the right corner is cut off. The +5V DC power line cannot be connected to the outside of the PCB, instead one of the inner solder points is used. Also see Figure 7.

For the organ a total of 96 different notes are available, but only 15 can be played. (See Figure 8). The selected default tone frequencies are marked. If required, other frequencies can be chosen by altering the tone array in the Arduino source code.

	Octave 1	Octave 2	Octave 3	Octave 4	Octave 5	Octave 6	Octave 7	Octave 8
C	33	65	131	262	**523**	**1047**	2093	4186
C#	35	69	139	277	554	1109	2217	4435
D	37	73	147	294	**587**	**1175**	2349	4699
D#	39	78	156	311	622	1245	2489	4978
E	41	82	165	330	**659**	**1319**	2637	5274
F	44	87	175	**349**	**698**	**1397**	2794	5588
F#	46	93	185	370	740	1480	2960	5920
G	49	98	196	**392**	**784**	1568	3136	6272
G#	52	104	208	415	831	1661	3322	6645
A	55	110	220	**440**	**880**	1760	3520	7040
A#	58	117	233	466	932	1865	3729	7459
B	62	123	247	**494**	**988**	1976	3951	7902

Default frequencies (Hz)

Figure 8 – Available tones with the default frequencies

Usage

The buzzer socket will give a tone of 750 Hz. The Organ can be played by:

1. Putting a 4-bit value from 1-15 on the 4-bit data bus;
2. Putting a 1 to the enable socket.

If the 4-bit value is 0 or the enable is 0, no sound will be played.

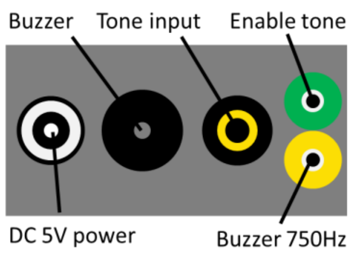

Figure 9 – Controls and connectors

Arduino Nano Solder check

```
/* ULU.22 Buzzer - solder check */
/* CC BY-NC-SA Jeroen Brinkman */

int note, freq, lastfreq;
const int tones[16] = {0, 349, 392, 440, 494, 523, 587, 659, 698, 784, 880, 988, 1046, 1175, 1319, 1397};

#define MORSE 750
#define HEART  1
#define PLAY  2
#define BEEP  3
#define TONE0  4
#define TONE1  5
#define TONE2  6
#define TONE3  7
#define BUZZER  8
#define LED  13

void setup() {
   pinMode(TONE0, INPUT);
   pinMode(TONE1, INPUT);
   pinMode(TONE2, INPUT);
   pinMode(TONE3, INPUT);
   pinMode(BEEP, INPUT);
   pinMode(PLAY, INPUT);
   pinMode(BUZZER, OUTPUT);
   pinMode(LED, OUTPUT);
   pinMode(HEART,  OUTPUT); // blinking led showing the programma's hartbeat
   tone(BUZZER, MORSE); delay(500); noTone(BUZZER);
   lastfreq = 0;
}

void loop(){
   digitalWrite(HEART, (millis() / 1000) % 2); //1s heartbeat for the onboard led

   /* Read input */
   freq = 0;
   if (digitalRead(BEEP) == HIGH) freq = MORSE;
   if (digitalRead(PLAY) == HIGH) freq = tones[3];
   if (digitalRead(TONE0) == HIGH) freq = tones[6];
   if (digitalRead(TONE1) == HIGH) freq = tones[9];
   if (digitalRead(TONE2) == HIGH) freq = tones[12];
   if (digitalRead(TONE3) == HIGH) freq = tones[15];

   /* Play tone */
   if ((freq > 0) && (lastfreq !=freq)) {
      tone(BUZZER, freq);
      digitalWrite(LED, HIGH);
   }
   if ((freq == 0) && (lastfreq > 0)) {
      noTone(BUZZER);
      digitalWrite(LED, LOW);
   }
   lastfreq = freq;
}
```

Arduino Nano program

```
/* ULU.22 Buzzer - program code */
/* CC BY-NC-SA Jeroen Brinkman */

#define BOUNCE 8
int freq, lastfreq; // Frequencies
int note, index, count, maxcount; // Integers
const int tones[16] = {0, 349, 392, 440, 494, 523, 587, 659, 698, 784, 880, 988, 1046, 1175, 1319, 1397};
int input[BOUNCE];

#define ONE  50
#define TWO  100
#define THREE  150
#define SIX  300
#define MORSE 750
#define HEART  1
#define PLAY  2
#define BEEP  3
#define TONE0  4
#define TONE1  5
```

```
#define TONE2  6
#define TONE3  7
#define BUZZER  8
#define LED  13

void setup() {
   pinMode(TONE0, INPUT);
   pinMode(TONE1, INPUT);
   pinMode(TONE2, INPUT);
   pinMode(TONE3, INPUT);
   pinMode(BEEP, INPUT);
   pinMode(PLAY, INPUT);
   pinMode(BUZZER, OUTPUT);
   pinMode(LED, OUTPUT);
   pinMode(HEART,  OUTPUT); // blinking led showing the programma's hartbeat

  /* Every ULU with an Arduino introduces itself. This one uses Morse code */
   tone(BUZZER, MORSE); delay(ONE); noTone(BUZZER); delay(ONE);
   tone(BUZZER, MORSE); delay(ONE); noTone(BUZZER); delay(ONE);
   tone(BUZZER, MORSE); delay(THREE); noTone(BUZZER); delay(ONE);
   tone(BUZZER, MORSE); delay(THREE); noTone(BUZZER); delay(ONE);
   tone(BUZZER, MORSE); delay(THREE); noTone(BUZZER); delay(ONE);
   delay(TWO);
   tone(BUZZER, MORSE); delay(ONE); noTone(BUZZER); delay(ONE);
   tone(BUZZER, MORSE); delay(ONE); noTone(BUZZER); delay(ONE);
   tone(BUZZER, MORSE); delay(THREE); noTone(BUZZER); delay(ONE);
   tone(BUZZER, MORSE); delay(THREE); noTone(BUZZER); delay(ONE);
   tone(BUZZER, MORSE); delay(THREE); noTone(BUZZER); delay(ONE);
   lastfreq = 0;
}

void loop(){
   digitalWrite(HEART, (millis() / 1000) % 2); //1s heartbeat for the onboard led

   /* Read note */
   for (int i = 0; i < BOUNCE; i++) {
      input[i] =  digitalRead(TONE0) + (digitalRead(TONE1) * 2) + (digitalRead(TONE2) * 4) + (digitalRead(TONE3) * 8);
   }
   /* Determine majority note, in order to avoid problems when a note played longer */
   maxcount = 0;
   index = -1;
   for (int i = 0; i < BOUNCE; i++) {
      count = 0;
      for (int j = 0; j < BOUNCE; j++) {
         if (input[i] == input[j]) count++;
      }
      if (count > maxcount) {
         maxcount = count;
         index = i;
      }
   }
   note = input[index];

   /* Determine action */
   freq = 0;
   if ((digitalRead(PLAY) == HIGH) && note > 0) freq = tones[note];
   if (digitalRead(BEEP) == HIGH) freq = MORSE;

   /* Play tone */
   if ((freq > 0) && (lastfreq !=freq)) {
      tone(BUZZER, freq);
      digitalWrite(LED, HIGH);
   }
   if ((freq == 0) && (lastfreq > 0)) {
      noTone(BUZZER);
      digitalWrite(LED, LOW);
   }
   lastfreq = freq;
}
```