UNIVERSAL
LOGIC
UNITS

ULU.12 – Up- & down counter

This ULU

The *ULU.12 – Up- & down counter* can be used to act as a visual counter or as a memory addressing unit.

Used parts

Only standard parts are used:

1x casing 80 x 50 x 20mm;
6x 2mm signal connector;
6x black O-ring 9 x 5 x 2mm;
2x 4-bit data connector;
2x colored O-ring 8 x 5 x 1.5mm;
1x power connector;
2x single digit 0.56" 7-segment display;
13x resistor for the display;
2x M3 standoff male 7mm;
2x black M3 bolt 8mm;
2x M3 lock nut;
1x ON-OFF switch;
1x Arduino Nano;
7x 10K pull-down resistor;
5x micro (G6K-2F-Y-5VDC) relay;
9x (fly back) diode (1N4148);
2x 16 x 8-hole prototype board.

Construction

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

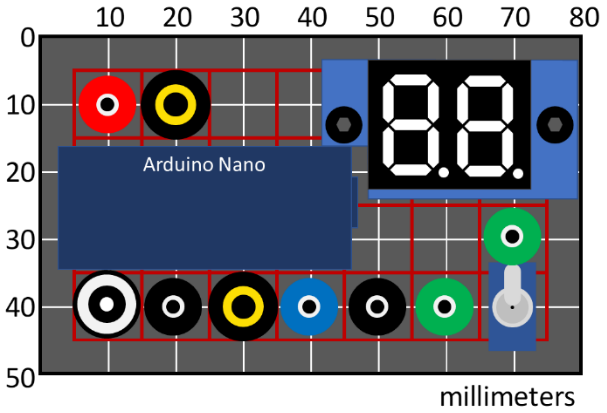

Figure 1 – Drill guide

The left digit will display the numbers 1, 2 and 3. In order to simplify the soldering, this is done with three segment-combinations as shown in Figure 3. Combination 1 will show a one, combinations 2 & 3 will show a two and combinations 1 & 3 will show a three.

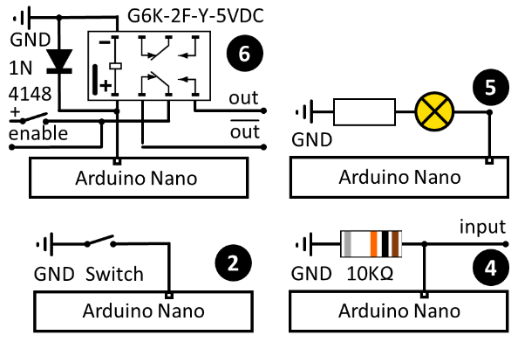

Figure 2 – Arduino interfaces

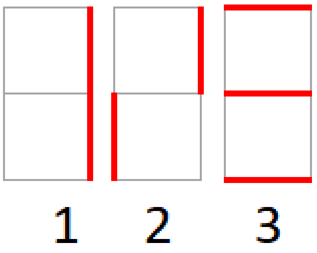

Figure 3 – Left digit encoding

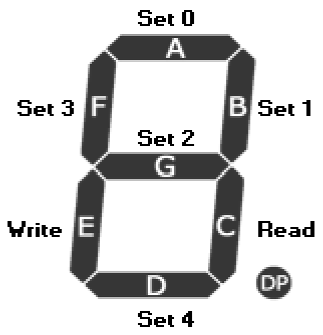

Figure 4 – Solder check

	Port	Con.	Rest.	Func.	Interface	Signal
1.	D1	8	O	O	❺	Left 1
2.	D0	8	O	O	❺	Left 2
3.	D2	8		O	❺	Left 3
4.	D3	④		I	❹	set value 0
5.	D4	④		I	❹	set value 1
6.	D5	④		I	❹	set value 2
7.	D6	④		I	❹	set value 3
8.	D7	④		I	❹	set value 4
9.	D8	8		O	❺	Right A
10.	D9	8		O	❺	Right B
11.	D10	8		O	❺	Right C
12.	D11	8		O	❺	Right D
13.	D12	8		O	❺	Right E
14.	D13	8	O	O	❺	Right F
15.	A0	8		O	❺	Right G
16.	A1	□		O	❻	Count 4
17.	A2	□		O	❻	Count 3
18.	A3	□		O	❻	Count 2
19.	A4	□		O	❻	Count 1
20.	A5	□		O	❻	Count 0
21.	A6	○	I	I	❹	cnt down/set
22.	A7	○	I	I	❹	cnt up/set
23.	+5V	⊙	I	I	⓿	+5V
24.	GND	⊙	I	I	⓿	GND

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

Figure 5 – Pinout Arduino Nano

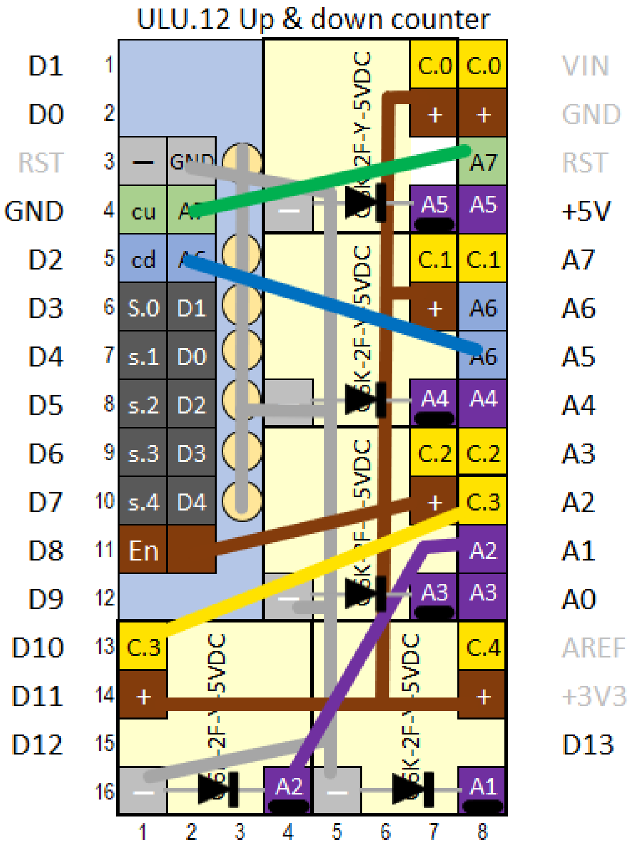

Figure 6 – Layout relay & resistor PCB

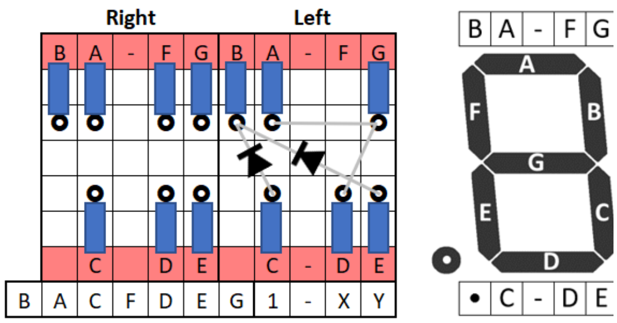

Figure 7 – 7-segment display layout

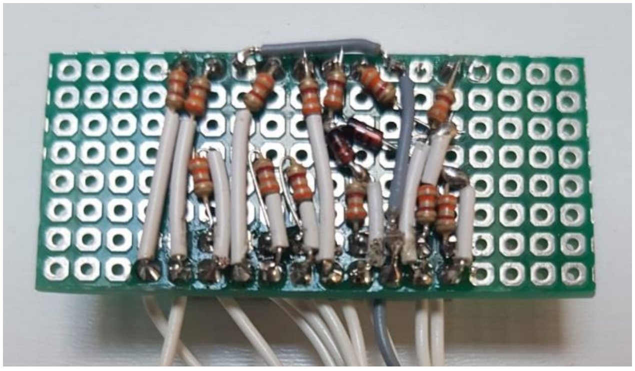

Figure 8 – 7-segment display soldered

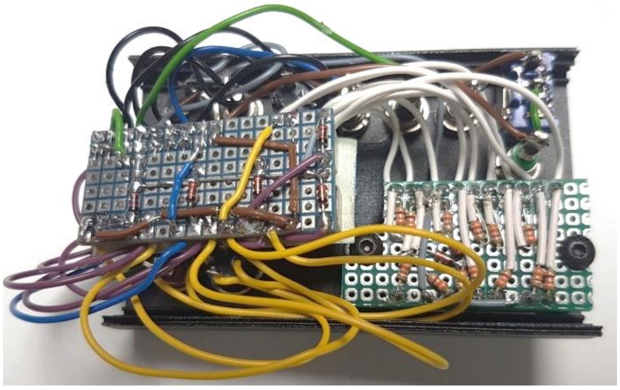

Figure 9 – ULU inside

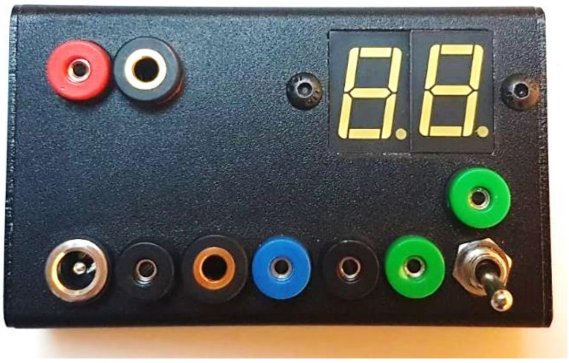

Figure 10 – Completed ULU

It is a very densely packed ULU, there is little space for the PCB's. The 7-segment display (See Figure 7) connections are soldered on the backside of the PCB. The insulator is removed from a wire and then put on an end of a resistor. Then the resistor is bend in a U-shape, making a connecting between the pin and the wire. The relay and resistor PCB is shown in Figure 6, the relay's and resistors are placed on top, the diodes on the bottom side of the PCB. Between the PCB and Arduino and on the metal casing, a piece of insulation tape is placed. The black *Set socket* is connected with two 1N4148 diodes to both the *Count up* and *Count down* sockets, triggering a signal on both sockets. The + end of the diodes is soldered to the *Set socket.*

It is strongly advised first load the solder check program. This program will:

1. first show the right segments a-g and the left segment-combinations 1-3;
2. than show the provided input on the 7-segment display as given in Figure 4;
3. If the output is enabled, the output will be equal to the set value that is on the input.

When connection and/or soldering errors are made, it is often less work to change the pin-numbers in the Arduino program than to resolder the connections.

Usage

The *ULU.12 Up- & down counter* can be used as an event counter, or in combination with the *ULU.11 Clock generator* as a countdown timer or a computer program counter. It can be used to address memory as well. The display counts from 0 to 39, the output – in combination with the overflow – from 0 to 31. The output is set to 10000 (only overflow signal) when the counter is counting from 32 to 39. The counter can be set by both providing a 1 on the *Set connector* in combination with the required value on the 5-bit set value. When nothing is connected to this set value, the set signal resets the clock to 0.

When counting upwards, the counter is a cyclic counter and will return to zero after 39 has been reached. If the carry output is connected to the set input, the counter acts as a 4-bit counter that resets to zero after 15 is reached. When counting downward the counter counts down to zero and will stay on zero. The *ULU.16 4-bit comparator* can be used to set or reset the counter when a predetermined value has been reached.

When the enable switch is set up, the corresponding socket can be used to obtain a logical 1 for use in the circuit.

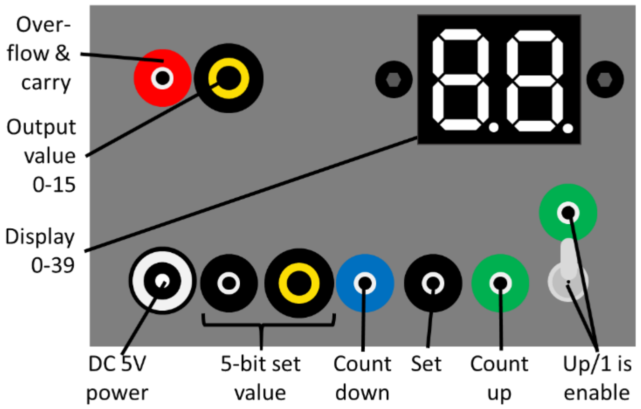

Figure 11 – Controls and connectors

Arduino Nano – solder check

```
/* ULU.12 Up- & down counter - solder check */
/* CC BY-NC-SA Jeroen Brinkman */

#define L1 1
#define L2 0
#define L3 2
#define SET0 3
#define SET1 4
#define SET2 5
#define SET3 6
#define SET4 7
#define RA 8
#define RB 9
#define RC 10
#define RD 11
#define RE 12
#define RF 13
#define RG A0
#define O4 A1
#define O3 A2
#define O2 A3
#define O1 A4
#define O0 A5
#define CD A6
#define CU A7

void setup() {
   pinMode(SET0, INPUT);
   pinMode(SET1, INPUT);
   pinMode(SET2, INPUT);
   pinMode(SET3, INPUT);
   pinMode(SET4, INPUT);
   pinMode(RA, OUTPUT);
   pinMode(RB, OUTPUT);
   pinMode(RC, OUTPUT);
   pinMode(RD, OUTPUT);
   pinMode(RE, OUTPUT);
   pinMode(RF, OUTPUT);
   pinMode(RG, OUTPUT);
   pinMode(L1, OUTPUT);
   pinMode(L2, OUTPUT);
   pinMode(L3, OUTPUT);
   pinMode(O4, OUTPUT);
   pinMode(O3, OUTPUT);
   pinMode(O2, OUTPUT);
   pinMode(O1, OUTPUT);
   pinMode(O0, OUTPUT);
   pinMode(CD, INPUT);
   pinMode(CU, INPUT);
   digitalWrite(RA, LOW);
   digitalWrite(RB, LOW);
   digitalWrite(RC, LOW);
   digitalWrite(RD, LOW);
   digitalWrite(RE, LOW);
   digitalWrite(RF, LOW);
   digitalWrite(RG, HIGH);
   digitalWrite(L1, LOW);
   digitalWrite(L2, LOW);
   analogWrite(L3, 0);
   delay(1000);
   digitalWrite(RA, HIGH); delay(1000);
   digitalWrite(RB, HIGH); delay(1000);
   digitalWrite(RC, HIGH); delay(1000);
   digitalWrite(RD, HIGH); delay(1000);
   digitalWrite(RE, HIGH); delay(1000);
   digitalWrite(RF, HIGH); delay(1000);
   digitalWrite(RG, HIGH); delay(1000);
   digitalWrite(L1, HIGH); delay(1000);
   digitalWrite(L1, LOW); delay(1000);
   digitalWrite(L2, HIGH); delay(1000);
   digitalWrite(L3, HIGH); delay(1000);
   digitalWrite(RA, LOW);
   digitalWrite(RB, LOW);
   digitalWrite(RC, LOW);
   digitalWrite(RD, LOW);
   digitalWrite(RE, LOW);
   digitalWrite(RF, LOW);
   digitalWrite(RG, HIGH);
   digitalWrite(L1, LOW);
   digitalWrite(L2, LOW);
   digitalWrite(L3, LOW); delay(1000);
}

void loop(){
   digitalWrite(RA, digitalRead(SET0));
   digitalWrite(RB, digitalRead(SET1));
   digitalWrite(RG, digitalRead(SET2));
   digitalWrite(RF, digitalRead(SET3));
   digitalWrite(RC, (analogRead(CU) > 500 ? HIGH : LOW));
   digitalWrite(RE, (analogRead(CD) > 500 ? HIGH : LOW));
   digitalWrite(RD, digitalRead(SET4));
   digitalWrite(O0, digitalRead(SET0));
   digitalWrite(O1, digitalRead(SET1));
   digitalWrite(O2, digitalRead(SET2));
   digitalWrite(O3, digitalRead(SET3));
   digitalWrite(O4, digitalRead(SET4));
}
```

Arduino Nano program

```
/* ULU.12 Up- & down counter - program code */
/* CC BY-NC-SA Jeroen Brinkman */

int count, lastcount, i, sumu, sumd; // integers
int up, down, lastup, lastdown; // signals
const bool binair[40][5] = {{0,0,0,0,0}, {0,0,0,0,1}, {0,0,0,1,0}, {0,0,0,1,1}, {0,0,1,0,0}, {0,0,1,0,1}, {0,0,1,1,0},
{0,0,1,1,1}, {0,1,0,0,0}, {0,1,0,0,1}, {0,1,0,1,0}, {0,1,0,1,1}, {0,1,1,0,0}, {0,1,1,0,1}, {0,1,1,1,0}, {0,1,1,1,1},
{1,0,0,0,0}, {1,0,0,0,1}, {1,0,0,1,0}, {1,0,0,1,1}, {1,0,1,0,0}, {1,0,1,0,1}, {1,0,1,1,0}, {1,0,1,1,1}, {1,1,0,0,0},
{1,1,0,0,1}, {1,1,0,1,0}, {1,1,0,1,1}, {1,1,1,0,0}, {1,1,1,0,1}, {1,1,1,1,0}, {1,1,1,1,1}, {1,0,0,0,0}, {1,0,0,0,0},
{1,0,0,0,0}, {1,0,0,0,0}, {1,0,0,0,0}, {1,0,0,0,0}, {1,0,0,0,0}, {1,0,0,0,0}};
const bool left[40][3] = {{0,0,0}, {0,0,0}, {0,0,0}, {0,0,0}, {0,0,0}, {0,0,0}, {0,0,0}, {0,0,0}, {0,0,0}, {0,0,0}, {1,0,0},
{1,0,0}, {1,0,0}, {1,0,0}, {1,0,0}, {1,0,0}, {1,0,0}, {1,0,0}, {1,0,0}, {1,0,0}, {0,1,1}, {0,1,1}, {0,1,1}, {0,1,1}, {0,1,1},
{0,1,1}, {0,1,1}, {0,1,1}, {0,1,1}, {0,1,1}, {1,0,1}, {1,0,1}, {1,0,1}, {1,0,1}, {1,0,1}, {1,0,1}, {1,0,1}, {1,0,1}, {1,0,1},
{1,0,1}};
const bool right[40][7] = {{1,1,1,1,1,1,0}, {0,1,1,0,0,0,0}, {1,1,0,1,1,0,1}, {1,1,1,1,0,0,1}, {0,1,1,0,0,1,1},
{1,0,1,1,0,1,1}, {1,0,1,1,1,1,1}, {1,1,1,0,0,0,0}, {1,1,1,1,1,1,1}, {1,1,1,1,0,1,1}, {1,1,1,1,1,1,0}, {0,1,1,0,0,0,0},
{1,1,0,1,1,0,1}, {1,1,1,1,0,0,1}, {0,1,1,0,0,1,1}, {1,0,1,1,0,1,1}, {1,0,1,1,1,1,1}, {1,1,1,0,0,0,0}, {1,1,1,1,1,1,1},
{1,1,1,1,0,1,1}, {1,1,1,1,1,1,0}, {0,1,1,0,0,0,0}, {1,1,0,1,1,0,1}, {1,1,1,1,0,0,1}, {0,1,1,0,0,1,1}, {1,0,1,1,0,1,1},
{1,0,1,1,1,1,1}, {1,1,1,0,0,0,0}, {1,1,1,1,1,1,1}, {1,1,1,1,0,1,1}, {1,1,1,1,1,1,0}, {0,1,1,0,0,0,0}, {1,1,0,1,1,0,1},
{1,1,1,1,0,0,1}, {0,1,1,0,0,1,1}, {1,0,1,1,0,1,1}, {1,0,1,1,1,1,1}, {1,1,1,0,0,0,0}, {1,1,1,1,1,1,1}, {1,1,1,1,0,1,1}};

#define BOUNCE 8
#define MAX 39
#define MIN 0
#define L1 1
#define L2 0
#define L3 2
#define SET0 3
#define SET1 4
#define SET2 5
#define SET3 6
#define SET4 7
#define RA 8
#define RB 9
#define RC 10
#define RD 11
#define RE 12
#define RF 13
#define RG A0
#define O4 A1
#define O3 A2
#define O2 A3
#define O1 A4
#define O0 A5
#define CD A6
#define CU A7

void setup() {
   pinMode(SET0, INPUT);
   pinMode(SET1, INPUT);
   pinMode(SET2, INPUT);
   pinMode(SET3, INPUT);
   pinMode(SET4, INPUT);
   pinMode(RA, OUTPUT);
   pinMode(RB, OUTPUT);
   pinMode(RC, OUTPUT);
   pinMode(RD, OUTPUT);
   pinMode(RE, OUTPUT);
   pinMode(RF, OUTPUT);
   pinMode(RG, OUTPUT);
   pinMode(L1, OUTPUT);
   pinMode(L2, OUTPUT);
   pinMode(L3, OUTPUT);
   pinMode(O4, OUTPUT);
   pinMode(O3, OUTPUT);
   pinMode(O2, OUTPUT);
   pinMode(O1, OUTPUT);
   pinMode(O0, OUTPUT);
   pinMode(CD, INPUT);
   pinMode(CU, INPUT);

   /* Every ULU with an Arduino introduces itself. This one uses the 7-segment display */
   digitalWrite(RA, HIGH);digitalWrite(RB, HIGH);digitalWrite(RD, HIGH);digitalWrite(RE, HIGH);
   digitalWrite(RG, HIGH);delay(1000); digitalWrite(L1, HIGH); delay(1000);
   digitalWrite(RA, LOW); digitalWrite(RB, LOW);  digitalWrite(RD, LOW); digitalWrite(RE, LOW); digitalWrite(RG, LOW);
   digitalWrite(L1, LOW); delay(1000);   count = 0; lastcount = 40; lastup = LOW; lastdown = LOW;
}

void loop(){
   /* read & process input */
   sumu = 0; sumd = 0;
   for (i = 0; i < BOUNCE; i++) {sumu += analogRead(CU); sumd += analogRead(CD); } // eliminate bouncing
   up = (sumu > (i * 512)) ? HIGH : LOW; down = (sumd > (i * 512)) ? HIGH : LOW;
   if ((down == HIGH) && (up == HIGH) && ((lastdown == LOW) || (lastup == LOW))) {
      count =  digitalRead(SET0) + (digitalRead(SET1) * 2) + (digitalRead(SET2) * 4) + (digitalRead(SET3) * 8) +
(digitalRead(SET4) * 16);}
   if ((up == HIGH) && (lastup == LOW)) count +=1;
   if ((down == HIGH) && (lastdown == LOW)) count -=1;
   lastup = up; lastdown = down;
   if (count >  MAX) count = MIN; if (count <  MIN) count = MIN;

   /* write output left[40][123] right[40][abcdefg] */
   if (count != lastcount) {
      digitalWrite(L1, left[count][0]);
      digitalWrite(L2, left[count][1]);
      digitalWrite(L3, left[count][2]);
      digitalWrite(RA, right[count][0]);
      digitalWrite(RB, right[count][1]);
      digitalWrite(RC, right[count][2]);
      digitalWrite(RD, right[count][3]);
      digitalWrite(RE, right[count][4]);
      digitalWrite(RF, right[count][5]);
      digitalWrite(RG, right[count][6]);
      digitalWrite(O4, binair[count][0]);
      digitalWrite(O3, binair[count][1]);
      digitalWrite(O2, binair[count][2]);
      digitalWrite(O1, binair[count][3]);
      digitalWrite(O0, binair[count][4]);
   }
   lastcount = count;
}
```