ULU.16 – 4-bit comparator

This ULU

The *ULU.16 – 4-bit comparator* will compare two 4-bit data words and generate a signal when the left signal is less, equal or greater than the right signal.

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 3mm round LED ;
2x resistor to dim the LED;
2x LED holder;
3x 5mm rectangular LED;
3x resistor to dim the LED;
1x M3 standoff male/female 6mm;
1x black M3 bolt 8mm;
1x M3 lock nut;
4x 1-pole ON-OFF switch;
1x Arduino Nano;
11x 10K pull-down resistor;
1x micro (G6K-2F-Y-5VDC) relay;
1x fly back diode (1N4148);
1x 11 x 5-hole prototype board.

Construction

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

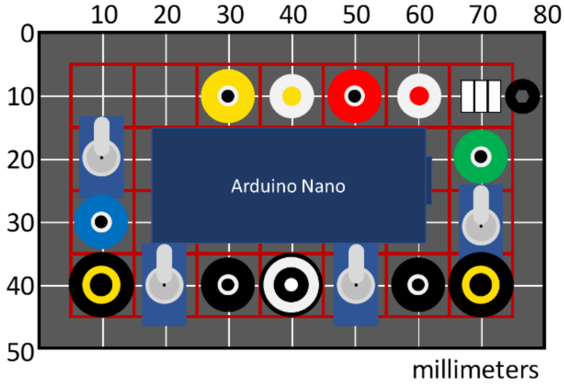


Figure 1 – Drill guide

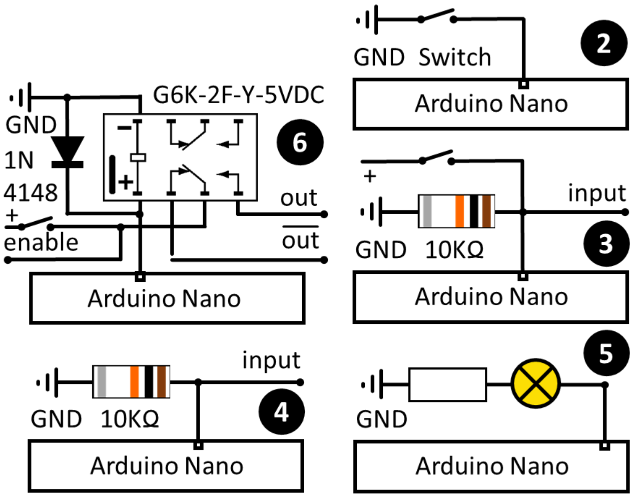


Figure 2 – Used Arduino interfaces

Construction is straight forward. A double-sided prototype PCB is cut to size. The components and wires are soldered on to the PCB. The wires of the PCB are soldered to the Arduino Nano, in a way that both PCB's can be folded like a book. When mounted in the enclosure, the wires are left as long as possible in order to be able to repair if necessary.

First the solder check program is loaded. It uses the output LEDs to test if the input is responding correctly. Don't forget to enable the output, otherwise the LED's won't burn.

	Port	Con.	Rest.	Func.	Interface	Signal
1.	D3	④		I	❹	left value 0
2.	D4	④		I	❹	left value 1
3.	D5	④		I	❹	left value 2
4.	D6	④		I	❹	left value 3
5.	D7	○		I	❹	Overflow
6.	D8	④		I	❹	right value 0
7.	D9	④		I	❹	right value 1
8.	D10	④		I	❹	right value 2
9.	D11	④		I	❹	right value 3
10.	D12	╿		I	❷	2-complement
11.	D13		O	L		Hartbeat
12.	A0	○		I	❸	Check 0
13.	A1	○	O	I	❸	Check 1
14.	A2	□		O	❻	Output
15.	A3	●		L	❺	< indicator
16.	A4	●		L	❺	= indicator
17.	A5	●		L	❺	> indicator
18.	+5V	⊙	I	I	⓿	+5V
19.	GND	⊙	I	I	⓿	GND

Input, Output, Led, SPI, Toggle switch, Rotary switch

Figure 3 – Pinout Arduino Nano

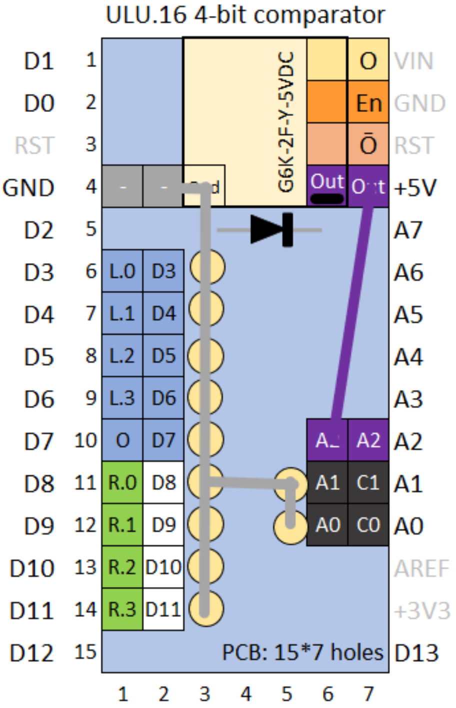


Figure 4 – Layout resistor & relay PCB

	Sinal	Output	< indicator	= indicator	> indicator
0	No signal	0	0	0	0
1	right value 0	0	0	0	1
2	right value 1	0	0	1	0
3	right value 2	0	0	1	1
4	right value 3	0	1	0	0
5	Check 0	0	1	0	1
6	Check 1	0	1	1	0
7	left value 0	0	1	1	1
8	left value 1 *	1	0	0	0
9	left value 2 *	1	0	0	1
10	left value 3 *	1	0	1	0
11	Overflow *	1	0	1	1
12	2-complement *	1	1	0	0

*** Don't forget to enable the output**

Figure 5 – Solder check signals

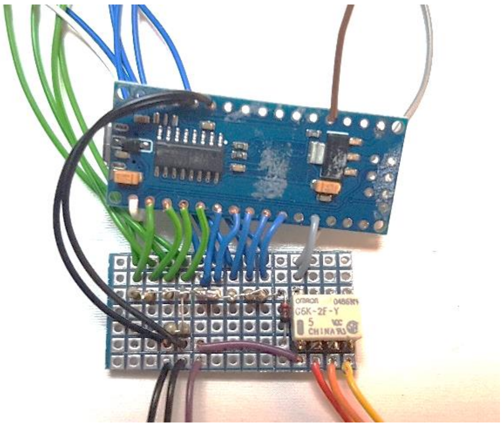
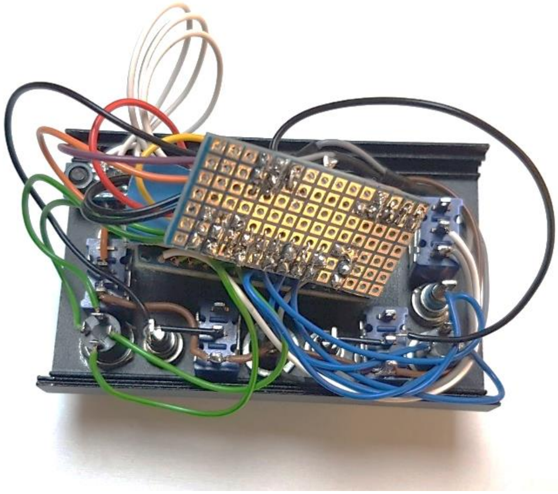


Figure 6 – The "book" PCB



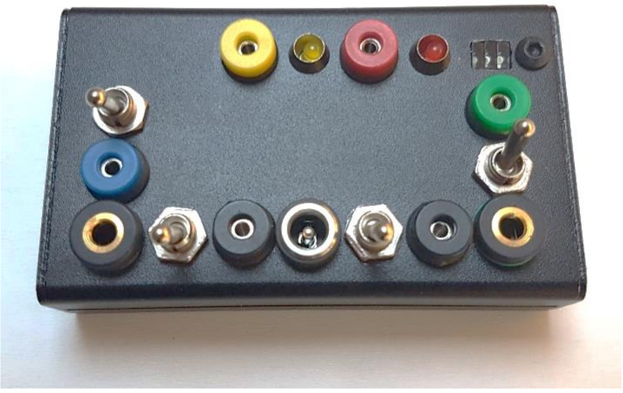


Figure 7 – ULU inside

Figure 8 – Finished ULU

Usage

The three LEDs used for the indication (<, =, >) of the relation between the two values are always pointing to the smallest value. The *ULU.14 4-bit comparato*r can be used to trigger an event, based on a predetermined value of a data-word. It can also be used to reset the *ULU.11 Up- & down counter* when a specific value is reached. Finally, it can be used to enable a jump instruction after an ALU calculation.

The required check can be selected by a two-bit signal input or two switches. Because there is also an inverted output, there are eight checks available (see Figure 9). If required, a relay (*ULU.03 Dual relay*) can be used to select one of the two (yellow or red) outputs.

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

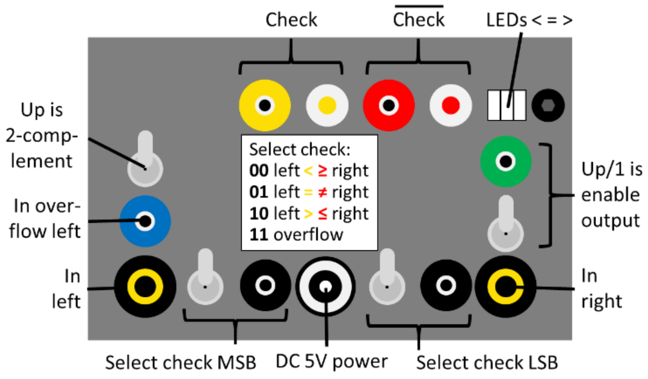


Figure 9 – Controls and connectors

Arduino Nano Solder check

```
/* ULU.16 4-bit comparator - solder check */
/* CC BY-NC-SA Jeroen Brinkman */

bool pressed;
#define LE0 3 // Adjust pin numbers when pins have changed
#define LE1 4
#define LE2 5
#define LE3 6
#define OVER 7
#define RI0 8
#define RI1 9
#define RI2 10
#define RI3 11
#define TWOCO 12
#define HEART 13
#define CHK0 A0
#define CHK1 A1
#define OUT A2
#define ISG A3
#define ISE A4
#define ISL A5

void setup() {
   pinMode(LE0, INPUT);
   pinMode(LE1, INPUT);
   pinMode(LE2, INPUT);
   pinMode(LE3, INPUT);
   pinMode(RI0, INPUT);
   pinMode(RI1, INPUT);
   pinMode(RI2, INPUT);
   pinMode(RI3, INPUT);
   pinMode(OVER, INPUT);
   pinMode(CHK0, INPUT);
   pinMode(CHK1, INPUT);
   pinMode(TWOCO, INPUT_PULLUP);
   pinMode(HEART,  OUTPUT); // blinking led showing the programma's hartbeat
   pinMode(OUT, OUTPUT);
   pinMode(ISL, OUTPUT);
   pinMode(ISE, OUTPUT);
   pinMode(ISG, OUTPUT);
   /* Testing output */
   digitalWrite(ISG, HIGH); delay(1000);
   digitalWrite(ISE, HIGH); delay(1000);
   digitalWrite(ISL, HIGH); delay(1000);
   digitalWrite(OUT, HIGH); delay(1000);
   digitalWrite(ISG, LOW);
   digitalWrite(ISE, LOW);
   digitalWrite(ISL, LOW);
   digitalWrite(OUT, LOW);
}

void loop(){
   digitalWrite(HEART, (millis() / 1000) % 2); //1s heartbeat for the onboard led
   pressed = false;
   /* Testing input */
   if (digitalRead(RI0) == HIGH) {  // 0001
      digitalWrite(OUT, LOW); //3
      digitalWrite(ISL, LOW);    //2
      digitalWrite(ISE, LOW);    //1
      digitalWrite(ISG, HIGH);   //0
      pressed = true;
   }
   if (digitalRead(RI1) == HIGH) {  // 0010
      digitalWrite(OUT, LOW); //3
      digitalWrite(ISL, LOW);    //2
      digitalWrite(ISE, HIGH);   //1
      digitalWrite(ISG, LOW);   //0
      pressed = true;
   }
   if (digitalRead(RI2) == HIGH) {  // 0011
      digitalWrite(OUT, LOW); //3
      digitalWrite(ISL, LOW);    //2
      digitalWrite(ISE, HIGH);   //1
      digitalWrite(ISG, HIGH);   //0
      pressed = true;
   }
   if (digitalRead(RI3) == HIGH) {  // 0100
      digitalWrite(OUT, LOW); //3
      digitalWrite(ISL, HIGH);   //2
      digitalWrite(ISE, LOW);    //1
      digitalWrite(ISG, LOW);   //0
      pressed = true;
   }
   if (digitalRead(CHK0) == HIGH) {  // 0101
      digitalWrite(OUT, LOW); //3
      digitalWrite(ISL, HIGH);   //2
      digitalWrite(ISE, LOW);    //1
      digitalWrite(ISG, HIGH);   //0
      pressed = true;
   }
   if (digitalRead(CHK1) == HIGH) {  // 0110
      digitalWrite(OUT, LOW); //3
      digitalWrite(ISL, HIGH);   //2
      digitalWrite(ISE, HIGH);   //1
      digitalWrite(ISG, LOW);   //0
      pressed = true;
   }
   if (digitalRead(LE0) == HIGH) {  // 0111
      digitalWrite(OUT, LOW); //3
      digitalWrite(ISL, HIGH);   //2
      digitalWrite(ISE, HIGH);   //1
      digitalWrite(ISG, HIGH);   //0
      pressed = true;
   }
   if (digitalRead(LE1) == HIGH) {  // 1000
      digitalWrite(OUT, HIGH);  //3
      digitalWrite(ISL, LOW);     //2
      digitalWrite(ISE, LOW);     //1
      digitalWrite(ISG, LOW);   //0
      pressed = true;
   }
   if (digitalRead(LE2) == HIGH) {  // 1001
      digitalWrite(OUT, HIGH);  //3
      digitalWrite(ISL, LOW);     //2
      digitalWrite(ISE, LOW);     //1
      digitalWrite(ISG, HIGH);    //0
      pressed = true;
   }
   if (digitalRead(LE3) == HIGH) {  // 1010
      digitalWrite(OUT, HIGH); //3
      digitalWrite(ISL, LOW);    //2
      digitalWrite(ISE, HIGH);   //1
      digitalWrite(ISG, LOW);   //0
      pressed = true;
   }
   if (digitalRead(OVER) == HIGH) {  // 1011
      digitalWrite(OUT, HIGH); //3
      digitalWrite(ISL, LOW);    //2
      digitalWrite(ISE, HIGH);   //1
      digitalWrite(ISG, HIGH);   //0
      pressed = true;
   }
   if (digitalRead(TWOCO) == LOW) {  // 1100
      digitalWrite(OUT, HIGH); //3
      digitalWrite(ISL, HIGH);   //2
      digitalWrite(ISE, LOW);    //1
      digitalWrite(ISG, LOW);   //0
      pressed = true;
   }
   if (!pressed) {  // 0000
      digitalWrite(OUT, LOW); //3
      digitalWrite(ISL, LOW);   //2
      digitalWrite(ISE, LOW);    //1
      digitalWrite(ISG, LOW);   //0
   }
}
```

Arduino Nano program

```
/* ULU.16 4-bit comparator - program code */
/* CC BY-NC-SA Jeroen Brinkman */

int left, right, check; // integers
int output, less, equal, greater; // signals
bool overflow; // booleans

#define LE0 3
#define LE1 4
#define LE2 5
#define LE3 6
#define OVER 7
#define RI0 8
#define RI1 9
#define RI2 10
#define RI3 11
#define TWOCO 12
#define HEART 13
#define CHK0 A0
#define CHK1 A1
#define OUT A2
#define ISG A3
#define ISE A4
#define ISL A5

void setup() {
   pinMode(LE0, INPUT);
   pinMode(LE1, INPUT);
   pinMode(LE2, INPUT);
   pinMode(LE3, INPUT);
   pinMode(RI0, INPUT);
   pinMode(RI1, INPUT);
   pinMode(RI2, INPUT);
   pinMode(RI3, INPUT);
   pinMode(OVER, INPUT);
   pinMode(CHK0, INPUT);
   pinMode(CHK1, INPUT);
   pinMode(TWOCO, INPUT_PULLUP);
   pinMode(HEART,  OUTPUT); // blinking led showing the programma's hartbeat
   pinMode(OUT, OUTPUT);
   pinMode(ISL, OUTPUT);
   pinMode(ISE, OUTPUT);
   pinMode(ISG, OUTPUT);
   /* Every ULU with an Arduino introduces itself. This one uses Blinking LEDs */
   digitalWrite(ISE, HIGH); digitalWrite(ISG, HIGH); delay(100); digitalWrite(ISE, LOW); digitalWrite(ISG, LOW); delay(100);
   digitalWrite(ISG, HIGH); delay(100); digitalWrite(ISG, LOW); delay(100);
   digitalWrite(ISG, HIGH); delay(100); digitalWrite(ISG, LOW); delay(100);
   digitalWrite(ISG, HIGH); delay(100); digitalWrite(ISG, LOW); delay(100);
   digitalWrite(ISG, HIGH); delay(100); digitalWrite(ISG, LOW); delay(100);
   digitalWrite(ISG, HIGH); delay(100); digitalWrite(ISG, LOW); delay(100);
}

void loop(){
    digitalWrite(HEART, (millis() / 1000) % 2); //1s heartbeat for the onboard led
   /* Read all input */
   check = (digitalRead(CHK1) * 2) + digitalRead(CHK0);
   overflow = (digitalRead(OVER) == HIGH);
   left =  digitalRead(LE0) + (digitalRead(LE1) * 2) + (digitalRead(LE2) * 4) + (digitalRead(LE3) * 8);
   right =  digitalRead(RI0) + (digitalRead(RI1) * 2) + (digitalRead(RI2) * 4) + (digitalRead(RI3) * 8);

   if (digitalRead(TWOCO) == LOW){
      if (left > 7) left -= 16;
      if (right > 7) right -= 16;
   }

   /* Perform checking */
   less = LOW; equal = LOW; greater = LOW; output=LOW;
   if (left < right) less = HIGH;
   if (left == right) equal = HIGH;
   if (left > right) greater = HIGH;
   switch (check) {
      case 0: // left < right
         output = less;
         break;
      case 1: // left = right
         output = equal;
         break;
      case 2: // left > right
         output = (((greater == HIGH) || overflow) ? HIGH : LOW);
         break;
      case 3: // overflow
           output = overflow ? HIGH : LOW;
      break;
      }

   /* Write output */
   digitalWrite(OUT, output);
   if (overflow == LOW) {
      digitalWrite(ISL, less);
      digitalWrite(ISE, equal);
      digitalWrite(ISG, greater);
   } else {
      digitalWrite(ISL, !less);
      digitalWrite(ISE, !equal);
      digitalWrite(ISG, !greater);
   }
}
```