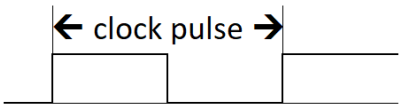
This ULU

The *ULU.11 – Clock generator* is a symmetric pulse generator with variable frequency from 0.1Hz to 20Hz. A clock pulse consists out of a 1 (high) and 0 (low) part, both 50% of the pulse.



Used parts

The following standard parts are used:

1x casing 80 x 50 x 20mm;
5x 2mm signal connector;
4x black O-ring 9 x 5 x 2mm;
1x power connector;
2x 3mm round LED;
2x resistor to dim the LED;
1x prototype board 15x7 holes;
3x 10K pull up resistor
2x LED holder;
1x push switch;
1x micro (G6K-2F-Y-5VDC) relay;
1x fly back diode (1N4148);
1x Arduino Nano;
3x 10K pull-down resistor.

In addition to that a rotary switch with push button and a 14mm x 16mm black aluminum knob are used.

Construction

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

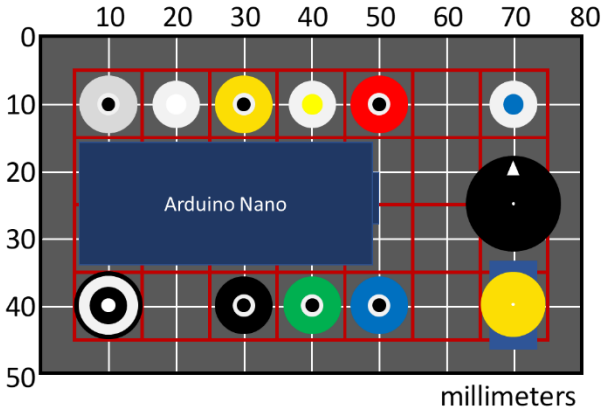


Figure 1 – Drill guide

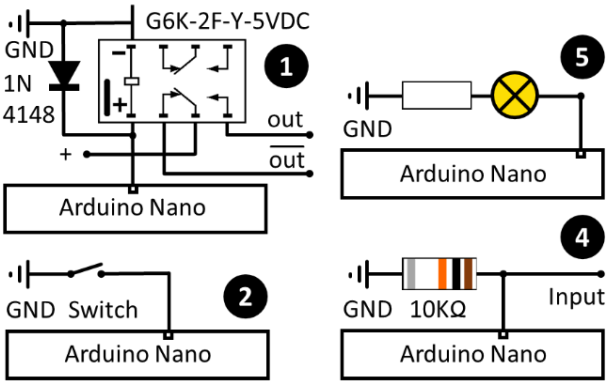


Figure 2 – Arduino interfaces

	sec	Hz	tot Ms		sec	Hz	tot Ms
1	20	0,05Hz	20000	12	1	1,0Hz	1000
2	15	0,07Hz	15000	13	0,9	1,11Hz	900
3	10	0,1Hz	10000	14	0,8	1,25Hz	800
4	9	0,11Hz	9000	15	0,7	1,43Hz	700
5	8	0,13Hz	8000	16	0,6	1,67Hz	600
6	7	0,14Hz	7000	17	0,5	2,0Hz	500
7	6	0,17Hz	6000	18	0,4	2,5Hz	400
8	5	0,2Hz	5000	19	0,3	3,33Hz	300
9	4	0,25Hz	4000	20	0,2	5,0Hz	200
10	3	0,33Hz	3000	21	0,1	10,0Hz	100
11	2	0,5Hz	2000	22	0,05	20,0Hz	50

Figure 3 – Available clock frequencies

	Port	Con.	Rest.	Func.	Interface	Signal
1.	D1		O	L		Hartbeat
2.	D2			I	4	Toggle count
3.	D3			I	4	Count on
4.	D4			I	4	Single pulse
5.	D5			P	2	Manual pulse
6.	D6			R	2	Rotary right
7.	D7			R	2	Rotary left
8.	D8			P	2	Rotary push
9.	D9			L	5	Boundary
9.	D10			O	1	Clock on
10.	D11			O	1	Clock pulse
11.	D13		O	L		Clock
12.	+5V		I	I	0	+5V DC
13.	GND		I	I	0	GND

Input, Output, Led, Push switch, Toggle switch, Rotary switch

Figure 4 – Pinout Arduino Nano

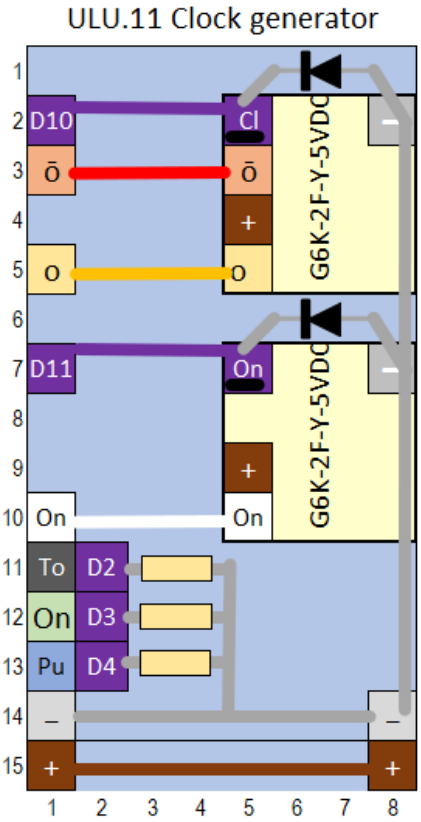


Figure 5 – Layout relay & resistor PCB

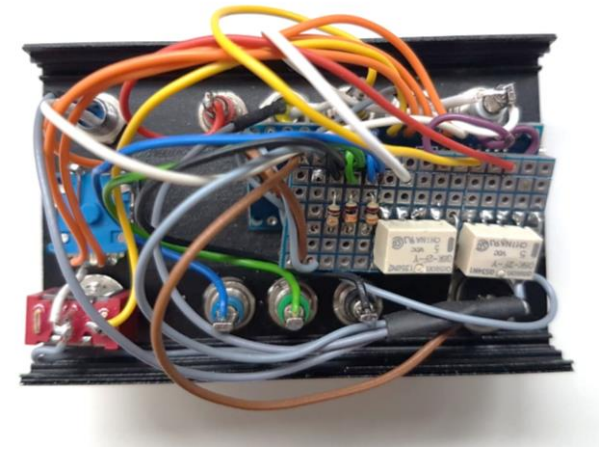


Figure 6 – ULU inside

Figure 7 – Finished ULU

Usage

This ULU can be used to generate a clock signal for repetitive processes. It can be activated in five different ways:

1. Initiating a manual pulse by pressing the yellow button. The output is high as long as the button is pressed.
2. Start/stop the clock manually, by pressing the black rotary knob.
3. Start/stop the clock with an on/off toggle signal, by feeding a signal to the green socket.
4. Run the clock while the signal is 1, by feeding a 1 to the black socket.

5. Generate a single pulse, by feeding a 1 to the blue socket. Only after the signal returned to 0, a new pulse will trigger a new clock pulse.

The frequency of the clock can be regulated between 0,1 Hz (10 seconds) and 20Hz (50ms) by manual turning the frequency select knob. Turn right will increase, turn left will decrease the frequency. The white LED above the knob will flash two times when the frequency increases and one time when it decreases. The LED will be permanent on when the frequency is set to 0.1 Hz, 1Hz and 20Hz.

When a single pulse (blue socket) is given and the inverted output (red socket) is used, the clock generator can be used as a signal delay line. The delay is regulated by turning the frequency select knob.

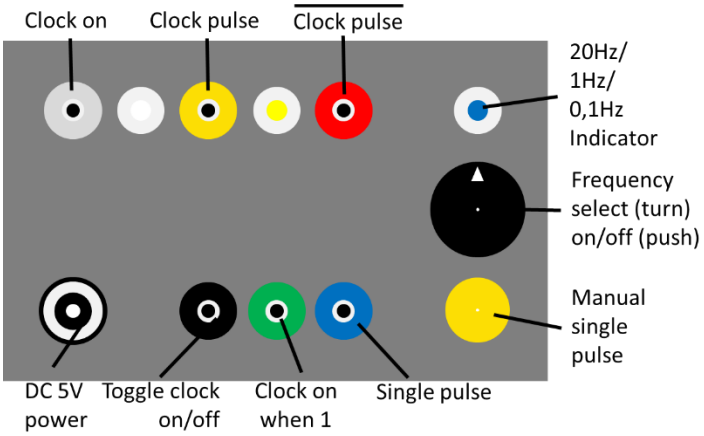


Figure 8 – Controls and connectors

Arduino Nano – solder check

```
/* ULU.11 Solder check */
/* CC BY-NC-SA Jeroen Brinkman */

int temp1, temp2;

#define MAX 21
#define CONT 2
#define TOGGLE 3
#define PULSE 4
#define MANUAL 5
#define ROTA 6 // Use ROTA 7 if right turn does decrease frequency
#define ROTB 7 // Use ROTB 6 if right turn does decrease frequency
#define ROTP 8
#define INDICATE 9
#define CLCKON 10
#define CLOCK 11
#define HEART  13

void setup() {
   pinMode(CONT, INPUT);
   pinMode(TOGGLE, INPUT);
   pinMode(PULSE, INPUT);
   pinMode(MANUAL, INPUT_PULLUP);
   pinMode(ROTA, INPUT_PULLUP);
   pinMode(ROTB, INPUT_PULLUP);
   pinMode(ROTP, INPUT_PULLUP);
   pinMode(INDICATE, OUTPUT);
   pinMode(CLOCK, CLCKON);
   pinMode(CLOCK, OUTPUT);
   pinMode(HEART,  OUTPUT); // blinking led showing the programma's hartbeat
}

void loop() {
   temp = LOW;
   temp = temp || digitalRead(CONT);
   temp = temp || digitalRead(TOGGLE);
   temp = temp || digitalRead(PULSE);
   temp = temp || !digitalRead(MANUAL);
   temp = temp || !digitalRead(ROTA);
   temp = temp || !digitalRead(ROTB);
   temp = temp || !digitalRead(ROTP);
   digitalWrite(13, (millis() / 1000) % 2); //1s heartbeat for the onboard led
  digitalWrite(INDICATE, temp);
  digitalWrite(CLOCK, temp);
  digitalWrite(HEART, temp);
}
```

Arduino Nano program

```
/* ULU.11 Clock generator - program code */
/* CC BY-NC-SA Jeroen Brinkman */

int i, freq, status, last_status, done, input; // integers
int signal1, signal2, last_rot, last_cont, last_pulse,  last_toggle, last_push, out, last_out; // signals
bool inp_toggle, inp_cont, inp_push,  inp_pulse, inp_manual; // booleans
unsigned long check, lastcheck, flip, lastflip; // milliseconds
int ms [22] = {10000, 7500, 5000, 4500, 4000, 3500, 3000, 2500, 2000, 1500, 1000, 500, 450, 400, 350, 300, 250, 200, 150, 100,
50, 25};
int state [16][4] = {{0, 1, 2, 0}, {1, 1, 2, 1}, {2, 2, 0, 2}, {2, 2, 0, 2}, {2, 2, 2, 2}, {2, 2, 2, 2}, {2, 2, 2, 2}, {2, 2,
2, 2}, {3, 3, 3, 3}, {3, 3, 3, 3}, {3, 3, 3, 3}, {3, 3, 3, 3}, {3, 3, 3, 3}, {3, 3, 3, 3}, {3, 3, 3, 3}, {3, 3, 3, 3}};

#define BOUNCE 8
#define MAX 21
#define EEN 1
#define TOGGLE 2
#define CONT 3
#define PULSE 4
#define MANUAL 5
#define ROTA 6 // Use ROTA 7 if right turn does decrease frequency
#define ROTB 7 // Use ROTB 6 if right turn does decrease frequency
#define ROTP 8
#define INDICATE 9
#define CLOCK 11
#define HEART  13
#define CHECK  180

void setup() {
   pinMode(EEN, OUTPUT);
   pinMode(CONT, INPUT);
   pinMode(TOGGLE, INPUT);
   pinMode(PULSE, INPUT);
   pinMode(MANUAL, INPUT_PULLUP);
   pinMode(ROTA, INPUT_PULLUP);
   pinMode(ROTB, INPUT_PULLUP);
   pinMode(ROTP, INPUT_PULLUP);
   pinMode(INDICATE, OUTPUT);
   pinMode(CLOCK, OUTPUT);
   pinMode(HEART,  OUTPUT); // blinking led showing the programma's hartbeat
   freq = 10; status = 0;
   last_toggle = LOW;  last_pulse = LOW; last_rot = LOW; last_push = LOW, out = LOW, last_out=LOW;
   lastcheck = millis(); flip = millis();
}

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

   /* read manual push */
   inp_manual  = (digitalRead(MANUAL) == LOW); // Manual pulse

   /* read on when logical 1 */
   signal1 = 0;
   for (int i = 0; i < BOUNCE; i++) {signal1 += digitalRead(CONT); } // eliminate bouncing
   signal1 = (signal1 > i / 2) ? HIGH : LOW;
   inp_cont = (signal1 == HIGH);
   inp_toggle = ((last_cont == HIGH) && (signal1 == LOW));
   last_cont = signal1;

   /* read toggle */
   signal1 = 0;
   for (int i = 0; i < BOUNCE; i++) {signal1 += !digitalRead(ROTP); } // eliminate bouncing
   signal1 = (signal1 > i / 2) ? HIGH : LOW;
   inp_toggle = inp_toggle || ((last_push == LOW) && (signal1 == HIGH));
   last_push = signal1;
   signal2 = 0;
   for (int i = 0; i < BOUNCE; i++) {signal2 += digitalRead(TOGGLE); } // eliminate bouncing
   signal2 = (signal2 > i / 2) ? HIGH : LOW;
   inp_toggle =  inp_toggle || ((last_toggle == LOW) && (signal2 == HIGH));
   last_toggle = signal2;

   /* single pulse */
   signal1 = digitalRead(PULSE); // Single pulse when 1
   inp_pulse = ((last_pulse == LOW) && (signal1 == HIGH) );
   last_pulse = signal1;

   /* read rotary function */
   signal1 = 0; signal2 = 0;
   for (int i = 0; i < BOUNCE; i++) {signal1 += !digitalRead(ROTA); } // eliminate bouncing
   signal1 = (signal1 > i / 2) ? HIGH : LOW;
   for (int i = 0; i < BOUNCE; i++) {signal2 += !digitalRead(ROTB); }  // eliminate bouncing
   signal2 = (signal2 > i / 2) ? HIGH : LOW;
   if ((last_rot == LOW) && (signal1 == HIGH)) {
      if (signal2 == LOW) {
         freq = freq - 1; if (freq < 0) freq = 0;
         digitalWrite(INDICATE, HIGH); delay(30);
         digitalWrite(INDICATE, LOW);
      } else {
         freq = freq + 1; if (freq > MAX) freq = MAX;
         digitalWrite(INDICATE, HIGH); delay(30);
         digitalWrite(INDICATE, LOW); delay(30);
         digitalWrite(INDICATE, HIGH); delay(30);
         digitalWrite(INDICATE, LOW);
      }
   }
   last_rot = signal1;
   digitalWrite(INDICATE, ((freq == MAX) || (freq == 0) || (freq == 10) ) ? HIGH : LOW);

   /* determine new clock status */
```

```
   // 0 - clock off
   // 1 - single pulse
   // 2 - clock on (pulse train)
   // 3 - out high (when manual button is pressed)
   input = 0;
   input += inp_manual ? 8 : 0;
   input += inp_cont ? 4 : 0;
   input += inp_toggle ? 2 : 0;
   input += inp_pulse ? 1 : 0;
   last_status = status;
   status = state[input][status];
   if (last_status != status) flip = millis();

   /* enable/disable clock according to status */
   check = millis() - lastcheck; // time needed for one check
   lastcheck = millis() ;
   switch (status) {
   case 0: // clock off
      out = LOW; done = 0;
      break;
   case 1: // single pulse
      if ((millis() < flip) && (millis() + check > flip)) delay(flip - millis());
      if (millis() >= flip) { // flip the output
         out = (out == HIGH) ? LOW : HIGH;
         flip = flip + (ms[freq]);
      }
      if ((done == 0) && (out == HIGH)) done = 1;
      if ((done == 1) && (out == LOW)) done = 2;
      if ((done == 2) && (out == HIGH)) {
        done = 3; out = LOW;
      }
      if ((done == 3) && !inp_pulse) status = 0;

      break;
   case 2: // clock on
      if ((millis() < flip) && (millis() + check > flip)) delay(flip - millis());
      if (millis() >= flip) { // flip the output
         out = (out == HIGH) ? LOW : HIGH;
         flip = flip + (ms[freq]);
      }
      break;
   case 3: // out high
      out = HIGH;
      break;
   }
   digitalWrite(HEART, out);
   digitalWrite(CLOCK, out);
}
```