

This ULU

The *ULU.31 Optical tape reader* offers a data input using optical paper tape. In combination with the serial input feature of the *ULU.17 64x8-bit RAM & display*, *ULU.27 32x4-bit RAM* or *ULU.32 16x4 message display*, those ULUs can be filled with data. It is a modern version of the punch paper tape.

Used parts

The following standard parts are used:

- 1x casing 80 x 50 x 20mm;
- 1x 2mm signal connector;
- 1x 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;
- 5x 10K pull-up resistor;
- 1x 5 LED bar;
- 5x micro (G6K-2F-Y-5VDC) relay;
- 5x fly back diode (1N4148);
- 2x 5mm M3 female/male standoff;
- 2x 5mm M3 bolt;
- 4x M3 nut;
- 1x Arduino Nano.

The following additional parts are used:

- 5x LDR 10K;
- 5x 3mm transparent white LED;
- 5x 200Ω resistor;
- 2x M3x20mm bolt;
- 1x aluminum bar 10x10050mm;
- 1x aluminum strip 10x1x50mm;
- 1x aluminum strip 20x2x50mm.

Construction

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

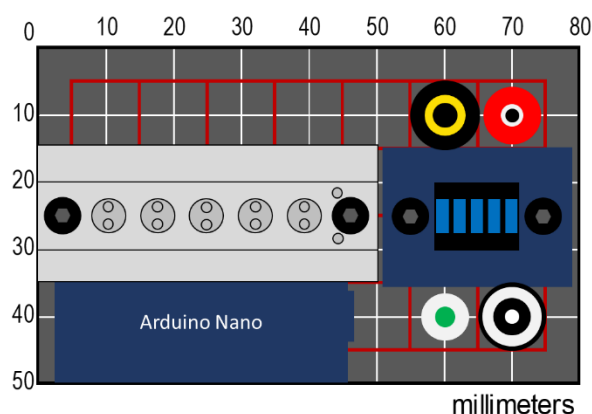


Figure 1 – Drill guide

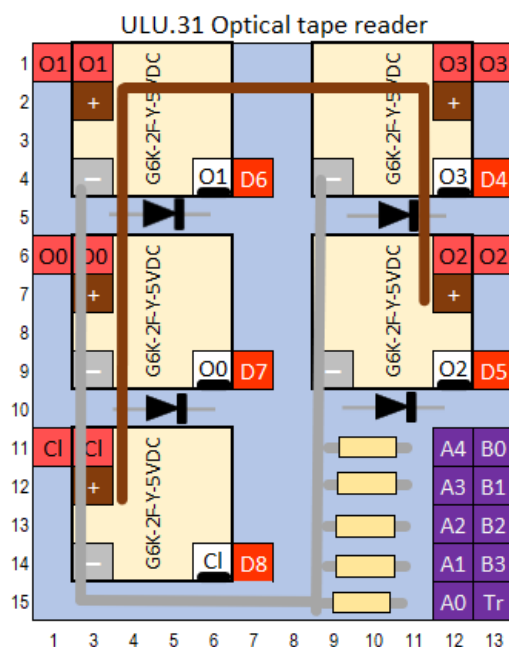


Figure 2 – PCB layout

This ULU is mechanically a more complicated one to make. The construction details can be found in Figure 5. A drill stand is needed to drill/mill the aluminum LED bridge. Another option is to 3d print this bridge. In this bridge LEDs are placed on top and LDRs at the bottom. The paper tape is fed between LEDs and LDRs.

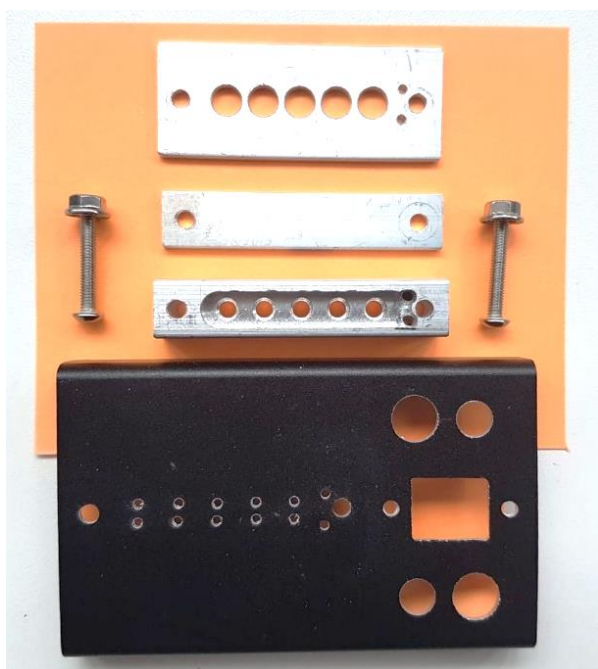


Figure 3 – Used mechanical parts

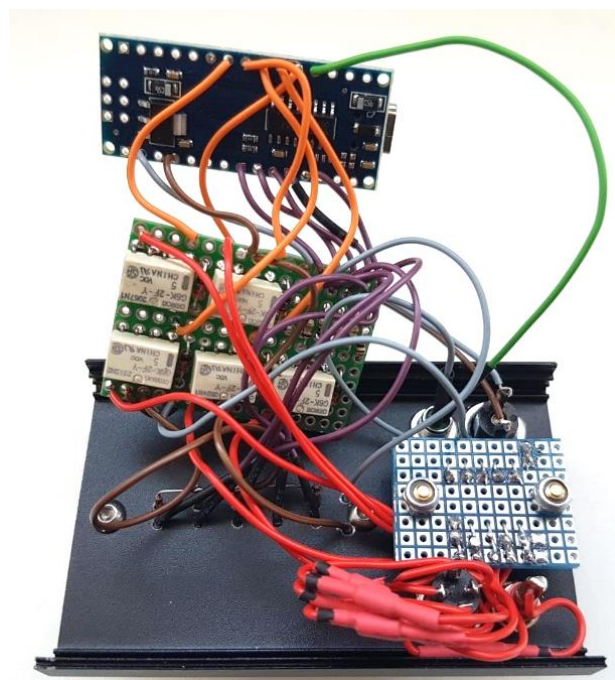


Figure 4 – ULU inside

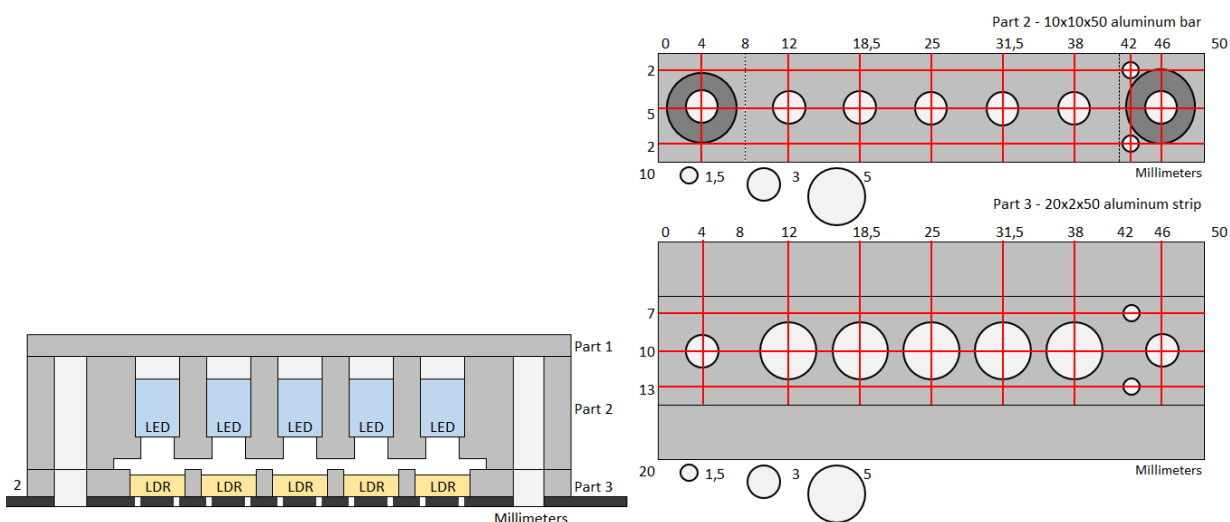


Figure 5 – Mechanical construction LED bridge

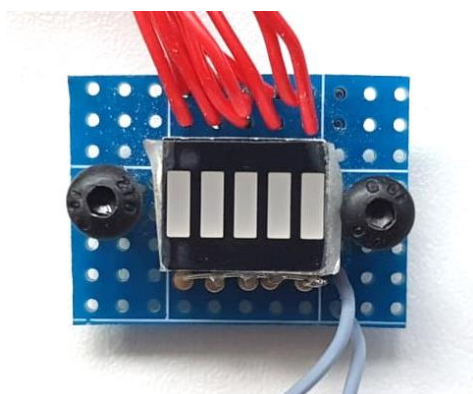


Figure 6 – LED bar graph

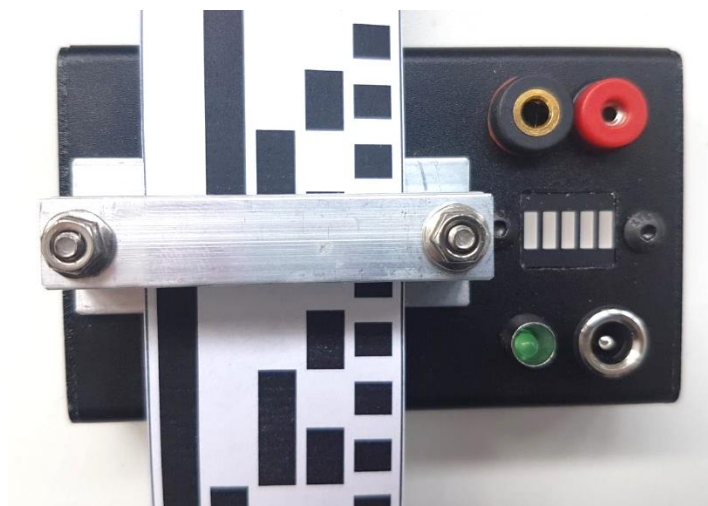























Figure 7 – Finished ULU

	Port	Con.	Rest.	Func.	Interface	Signal
1.	D4			O		Out 0
2.	D5			O		Out 1
3.	D6			O		Out 2
4.	D7			O		Out 3
5.	D8			O		Clock out
6.	D9			L		Paper
7.	D13		O	L		Heartbeat
8.	A4			I		Bit0
9.	A3			I		Bit1
10.	A2			I		Bit2
11.	A1			I		Bit3
12.	A0			I		Trigger
13.	+5V		I	I		+5V
14.	GND		I	I		GND

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

Figure 8 – Arduino pinout

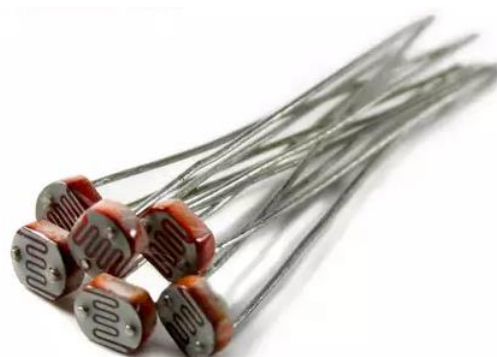


Figure 9 – Used LDRs

Before the tape reader can be used, it must be calibrated for the five specific LDRs used. Therefore, the solder check program can be used. This program prints the read values of these LDRs on the serial monitor of the Arduino IDE. When a sample paper tape is fed in the ULU, the threshold between black and white must be chosen in the middle of the corresponding readings. Then the values in the code line:

```
int thres[5] = {975,975,975,975,500};
```

can be adjusted accordingly for the left LDR – right LDR. The line:

```
#define THRESP 700
```

Can be adjusted to contain the proper threshold value for the detection of the paper.

Usage

For this ULU an Excel sheet, *ULU.31 Optical tape maker*, is available to make the necessary tapes. To avoid damaging the formulas in the sheet, the tabs in the sheet are protected without using a password. This tool can be used in two ways. The first option is to enter a hexadecimal string leaded with a start character. This start character is needed to preserve the leading zeros. For instance, the string '>48656C6C6F20776F726C64' will give 'Hello world' in ASCII characters. This option is selected by setting the string variable to 'Y'.

The second option is to define 16 or more mnemonics in the corresponding tab. Then in the input tab select the appropriate mnemonic on each line on the paper tape. To use this option the string variable must be set to 'N'. After that, the tape tab needs to be printed where the printed tape has a width of 32-33mm to fit the reader well (see Figure 10).



Figure 10 – Example paper tape

When the data out socket is connected to the data input socket and the clock out socket to the serial programming socket of the ULU that has to be filled with data, the data is easily entered.
When necessary, several paper tapes can be used sequentially.

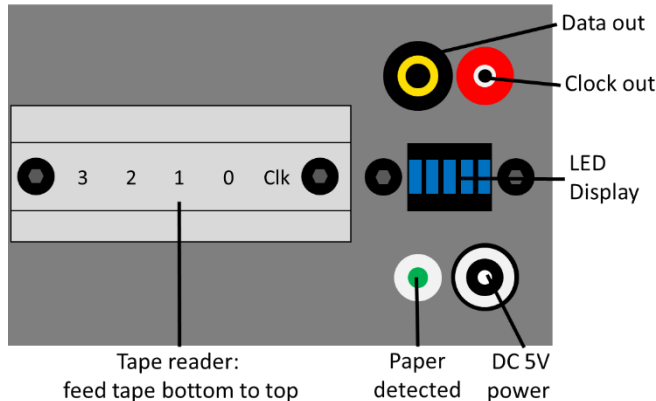


Figure 11 – Controls and connectors

Arduino Nano solder check

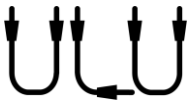
```
/* ULU.31 Optical tape reader - test program */
/* CC BY-NC-SA Jeroen Brinkman */

int channel; // integers
int thres[5] = {975,975,975,975,500}; // threshold for the four bits 0,1,2,3 and trigger (4) difference between black (1) and white (0)

#define THRESP 700 // threshold when the trigger has detected paper
#define BOUNCE 8
#define BIT3 A4
#define BIT2 A3
#define BIT1 A2
#define BIT0 A1
#define TRIGGER A0
#define PAPER 9
#define OUT3 4
#define OUT2 5
#define OUT1 6
#define OUT0 7
#define CLOCK 8
#define HEART 13

void setup() {
  Serial.begin(9600); // enables the serial monitor
  pinMode(PAPER, OUTPUT);
  pinMode(CLOCK, OUTPUT);
  pinMode(OUT0, OUTPUT);
  pinMode(OUT1, OUTPUT);
  pinMode(OUT2, OUTPUT);
  pinMode(OUT3, OUTPUT);
  pinMode(HEART, OUTPUT);
  channel = 0;
};

void loop(){
  digitalWrite(HEART, (millis() / 1000) % 2); //1s heartbeat for the onboard led
  Serial.print(analogRead(TRIGGER) < THRESP ? "PAPER: " : "WAIT: ");
  digitalWrite(PAPER, (analogRead(TRIGGER) < THRESP ? HIGH : LOW));
  Serial.print(analogRead(BIT3)); Serial.print("-> "); Serial.print(analogRead(BIT3) > thres[3] ? 0 : 1); Serial.print(", ");
  Serial.print(analogRead(BIT2)); Serial.print("-> "); Serial.print(analogRead(BIT2) > thres[2] ? 0 : 1); Serial.print(", ");
  Serial.print(analogRead(BIT1)); Serial.print("-> "); Serial.print(analogRead(BIT1) > thres[1] ? 0 : 1); Serial.print(", ");
  Serial.print(analogRead(BIT0)); Serial.print("-> "); Serial.print(analogRead(BIT0) > thres[0] ? 0 : 1); Serial.print(", ");
  Serial.print(analogRead(TRIGGER)); Serial.print("-> "); Serial.print(analogRead(TRIGGER) > thres[4] ? 0 : 1);
  Serial.println(", ");
  switch (channel) {
    case 0:
      digitalWrite(OUT3, LOW); digitalWrite(CLOCK, HIGH);
      break;
    case 1:
      digitalWrite(CLOCK, LOW); digitalWrite(OUT0, HIGH);
      break;
    case 2:
      digitalWrite(OUT0, LOW); digitalWrite(OUT1, HIGH);
      break;
    case 3:
      digitalWrite(OUT1, LOW); digitalWrite(OUT2, HIGH);
      break;
    case 4:
      digitalWrite(OUT2, LOW); digitalWrite(OUT3, HIGH);
      break;
  }
  channel += 1; if (channel == 5) channel = 0;
  delay(500);
}
```



```
}
```

Arduino Nano program

```
/* ULU.31 Optical tape reader - program code */
/* CC BY-NC-SA Jeroen Brinkman */

int i, direct, trigger, trigger_1, trigger_2, trigger_3, last, actual, sumt; // integers
int thres[5] = {975,975,975,975,500}; // threshold for the four bits 0,1,2,3 and trigger (4) difference between black (1) and white (0)

#define THRESP 700 // threshold when the trigger has detected paper
#define BOUNCE 3
#define THRES 2
#define BIT0 A4
#define BIT1 A3
#define BIT2 A2
#define BIT3 A1
#define TRIGGER A0
#define PAPER 9
#define OUT3 7
#define OUT2 6
#define OUT1 5
#define OUT0 4
#define CLOCK 8
#define HEART 13

void setup() {
  pinMode(PAPER, OUTPUT);
  pinMode(CLOCK, OUTPUT);
  pinMode(OUT0, OUTPUT);
  pinMode(OUT1, OUTPUT);
  pinMode(OUT2, OUTPUT);
  pinMode(OUT3, OUTPUT);
  pinMode(HEART, OUTPUT);

  /* Every ULU with an Arduino introduces itself. This uses a flashing LED*/
  digitalWrite(PAPER, HIGH); delay(100); digitalWrite(PAPER, LOW); delay(100);
  digitalWrite(PAPER, HIGH); delay(100); digitalWrite(PAPER, LOW); delay(100);
  digitalWrite(PAPER, HIGH); delay(100); digitalWrite(PAPER, LOW); delay(400);
  digitalWrite(PAPER, HIGH); delay(100); digitalWrite(PAPER, LOW);

  trigger_3 = 0; trigger_2 = 0; trigger_1 = 0; trigger = 0;
  direct = 0;
};

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

  /* direct meaning: */
  // 0. No paper
  // 1. downwards
  // 2. bottom (black)
  // 3. upwards
  // 4. top (white)
  // Black is less light is lower number

  /* Read & process trigger */
  sumt = 0;
  for (i = 0; i < BOUNCE; i++) {sumt += analogRead(TRIGGER); } // eliminate variances
  trigger = sumt / BOUNCE;
  if (trigger <= THRESP) { // paper inserted
    if (direct == 0) direct = 1;
    digitalWrite(PAPER, HIGH);
  } else {
    direct = 0;
    digitalWrite(PAPER, LOW);
    digitalWrite(OUT0, LOW);
    digitalWrite(OUT1, LOW);
    digitalWrite(OUT2, LOW);
    digitalWrite(OUT3, LOW);
    digitalWrite(CLOCK, LOW);
  }
  actual = trigger + trigger_1 + trigger_2;
  last = trigger_1 + trigger_2 + trigger_3;

  if ((last > (actual + THRES)) && (direct == 3)) direct = 4;
  if (((last + THRES) < actual) && (direct == 1)) direct = 2;
  trigger_3 = trigger_2; trigger_2 = trigger_1; trigger_1 = trigger;

  /* Write output */
  if (direct == 2) {
    direct = 3;
    digitalWrite(OUT0, (analogRead(BIT0) < thres[0] ? HIGH : LOW));
    digitalWrite(OUT1, (analogRead(BIT1) < thres[1] ? HIGH : LOW));
    digitalWrite(OUT2, (analogRead(BIT2) < thres[2] ? HIGH : LOW));
    digitalWrite(OUT3, (analogRead(BIT3) < thres[3] ? HIGH : LOW));
    digitalWrite(CLOCK, HIGH);
  }
  if (direct == 4) {
    direct = 1;
    digitalWrite(OUT0, LOW);
    digitalWrite(OUT1, LOW);
    digitalWrite(OUT2, LOW);
    digitalWrite(OUT3, LOW);
    digitalWrite(CLOCK, LOW);
  }
}
```

<pre>} }</pre>
