## This ULU

The *ULU.27 – 32x4-bit RAM* can be used as a RAM memory or as several registers connected to the same data bus. The content of the RAM is shown on two 8 x 8 dot-matrices.

## Used parts

Mostly standard parts are used:

1x casing 80 x 50 x 20mm;
4x 2mm signal connector;
4x 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;
12x 10K pull-up resistor;
4x micro (G6K-2F-Y-5VDC) relay;
4x fly back diode (1N4148);
1x Arduino Nano;
1x 20 x 7-holes prototype board.

In addition to that a 2x4cm, 8x16-dot matrix is used. I used the Open-smart 0.8 inch LED matrix. Also a 20 x 20mm aluminum L-profile and a wooden paint stirrer are used to mount the dot-matrix.

## Construction

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

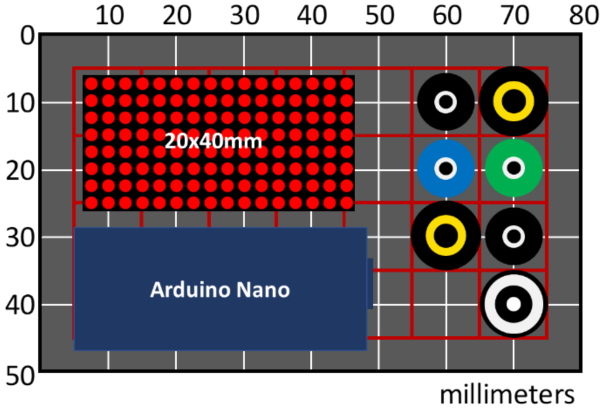

*Figure 1 – Drill guide*

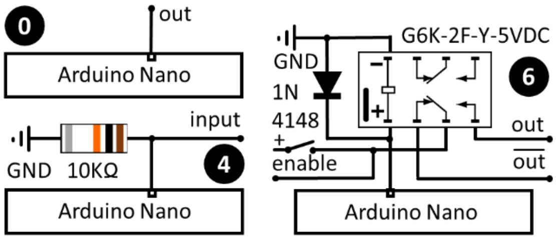

*Figure 2 – Used Arduino interfaces*

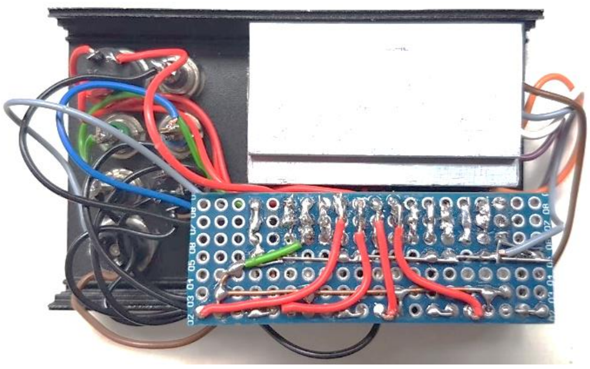

*Figure 3 – ULU inside*

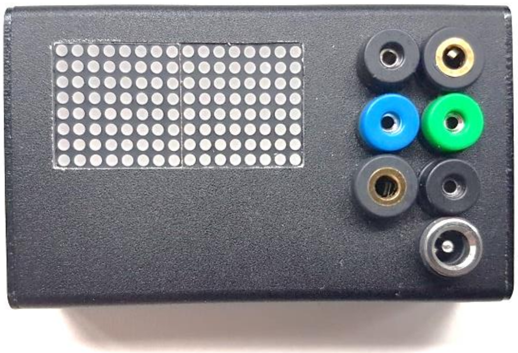

*Figure 4 – Finished ULU*

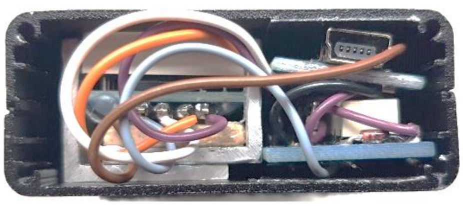

*Figure 5 – Clamping construction*

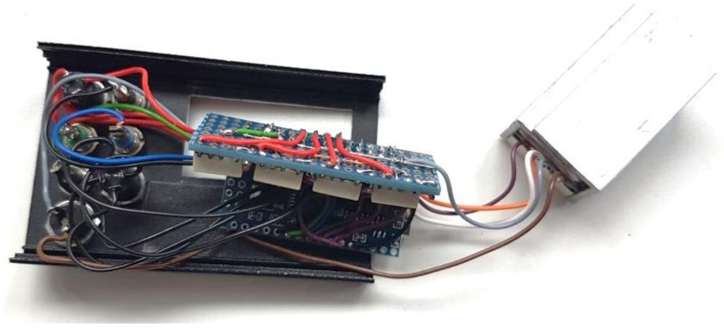

*Figure 6 – PCBs and display interconnected*

| | Port | Con. | Rest. | Func. | Interface | Signal |
|---|---|---|---|---|---|---|
| 1. | D1 | | O | L | | Heartbeat |
| 2. | D0 | | O | - | | - |
| 3. | D2 | ④ | | I | ❹ | DatIn 0 |
| 4. | D3 | ④ | | I | ❹ | DatIn 1 |
| 5. | D4 | ④ | | I | ❹ | DatIn 2 |
| 6. | D5 | ④ | | I | ❹ | DatIn 3 |
| 7. | D6 | ④ | | I | ❹ | Address 0 |
| 8. | D7 | ④ | | I | ❹ | Address 1 |
| 9. | D8 | ④ | | I | ❹ | Address 2 |
| 10. | D9 | ④ | | I | ❹ | Address 3 |
| 11. | D10 | 🟥 | | SPI | ⓿ | cs |
| 12. | D11 | 🟥 | | SPI | ⓿ | clk |
| 13. | D12 | 🟥 | | SPI | ⓿ | din |
| 14. | D13 | 🟥 | O | SPI | | - |
| 15. | A0 | ◉ | | I | ❹ | Bank |
| 16. | A1 | □ | | O | ❻ | DatOut 0 |
| 17. | A2 | □ | | O | ❻ | DatOut 1 |
| 18. | A3 | □ | | O | ❻ | DatOut 2 |
| 19. | A4 | □ | | O | ❻ | DatOut 3 |
| 20. | A5 | 🔵 | | I | ❹ | Write |
| 21. | A6 | 🟢 | | I | ❹ | Read |
| 22. | A7 | ◉ | | I | ❹ | Serial |
| 23. | +5V | ⊙ | I | I | ⓿ | +5V |
| 24. | GND | ⊙ | I | I | ⓿ | GND |

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

*Figure 7 – Pinout Arduino Nano*

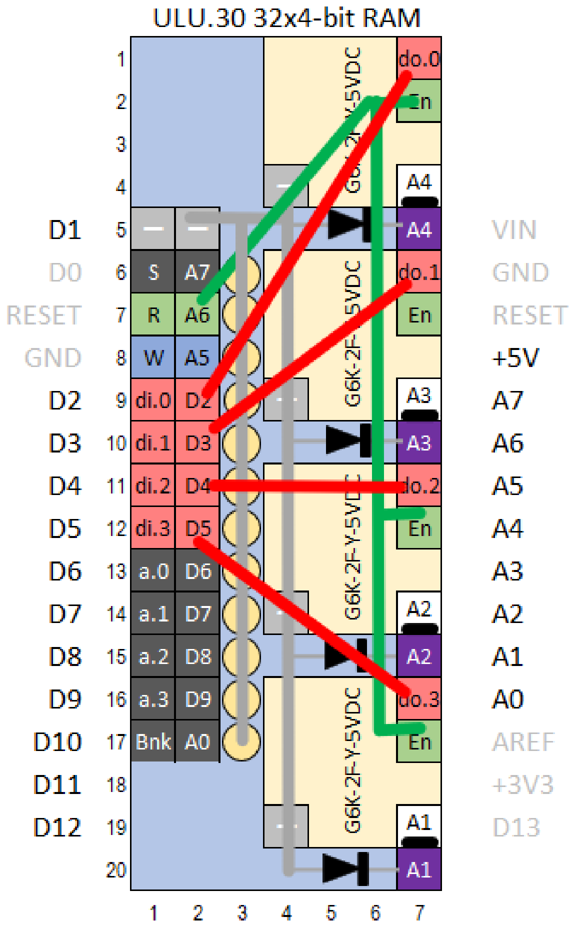

*Figure 8 – Layout resistor PCB*

Unfortunately, the only small dot matrix I could find, has no attachment points. Therefore, a construction needs to be made that clamps the matrix in the casing. I used two aluminum L-profiles, a paint stirrer and double-sided adhesive tape to fix this (See Figure 5). For insulation, duct tape is placed on the inside of the casing.

A library (LedControl.h) is needed and can be found in GitHub to control the LED display.

## Usage

The 32x4-bit RAM memory is a memory module that can be used for random access memory or as 32 registers attached to the same data bus. It has two memory banks of 16 4-bit words and has two modus operandi:

1. The serial programming-mode.
2. The write/read-mode.

The first mode can be used to prefill the memory. When a 1 is set on the serial input, this ULU will store the 4-bit input data on the current memory address. This memory address is initially the same as the actual memory address. After that, the subsequent memory addresses are determined by adding 1 to the last used address. First bank 0 will be written and then bank 1. The serial programming mode is entered by putting a 1 on the read or write connector. The *ULU.20 Hex keyboard,* the *ULU.05 Octal switch* and the *ULU.31 Optical tape reader* can be used as input devices for the serial programming.

In the write/read-mode the actual address will be written with the value of the actual data input when a 1 is entered on the write-socket. When a 1 is entered on the read-socket the content of the actual memory address is put on to the data output.

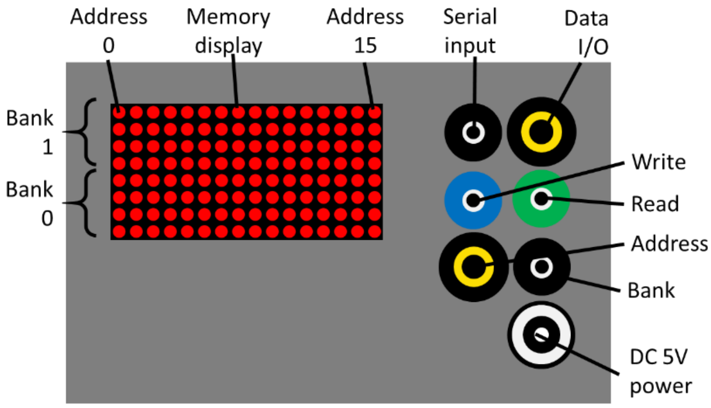

*Figure 9 – Controls and connectors*

| Signal | Row | Dot |
|---|---|---|
| Address 0 | 1 | 1 |
| Address 1 | 1 | 2 |
| Address 2 | 1 | 3 |
| Address 3 | 1 | 4 |
| Bank | 1 | 5 |
| Data 0 | 2 | 1 |
| Data 1 | 2 | 2 |
| Data 2 | 2 | 3 |
| Data 3 | 2 | 4 |
| Write | 3 | 1 |
| Read | 3 | 2 |
| Serial | 3 | 3 |

*Figure 10 – Solder check*

## Arduino Nano Solder check

```
/* ULU.27 32x4-bit RAM memory - solder check */
/* CC BY-NC-SA Jeroen Brinkman */

#include "LedControl.h"
LedControl lc=LedControl(12,11,10,2);

int address, woord, varia; // 4-bit words

#define HEART 1
#define DATI0 2
#define DATI1 3
#define DATI2 4
#define DATI3 5
#define AD0 6
#define AD1 7
#define AD2 8
#define AD3 9
#define BANK A0
#define DATO0 A1
#define DATO1 A2
#define DATO2 A3
#define DATO3 A4
#define WRITE A5
#define READ A6
#define SERI A7

void setup() {
   pinMode(HEART, OUTPUT);
   pinMode(DATO0, OUTPUT);
   pinMode(DATO1, OUTPUT);
   pinMode(DATO2, OUTPUT);
   pinMode(DATO3, OUTPUT);
   pinMode(DATI0, INPUT);
   pinMode(DATI1, INPUT);
   pinMode(DATI2, INPUT);
   pinMode(DATI3, INPUT);
   pinMode(BANK, INPUT);
   pinMode(AD0, INPUT);
   pinMode(AD1, INPUT);
   pinMode(AD2, INPUT);
   pinMode(AD3, INPUT);
   pinMode(WRITE, INPUT);
   pinMode(READ, INPUT);
   pinMode(SERI, INPUT);

   /* Initiate dot matrix */
   lc.shutdown(0,false); //The MAX72XX is in power-saving mode on startup, wakeup the display
   lc.shutdown(1,false); //The MAX72XX is in power-saving mode on startup, wakeup the display
   lc.setIntensity(0, 1); // Set the brightness to a low value
   lc.setIntensity(1, 1); // Set the brightness to a low value
   lc.clearDisplay(0); //clear the display
   lc.clearDisplay(1); //clear the display

   /* Every ULU with an Arduino introduces itself. This one uses dot text */
   lc.setColumn(0,7,62);
   lc.setColumn(0,6,32);
   lc.setColumn(0,5,62);
   lc.setColumn(0,3,62);
   lc.setColumn(0,2,32);
   lc.setColumn(0,1,32);
   lc.setColumn(1,7,62);
   lc.setColumn(1,6,32);
   lc.setColumn(1,5,62);
   lc.setColumn(1,3,32);
   delay(800);
   lc.clearDisplay(0); //clear the display
   lc.clearDisplay(1); //clear the display
   lc.setColumn(0,7,58);
   lc.setColumn(0,6,42);
   lc.setColumn(0,5,46);
   lc.setColumn(0,3,2);
   lc.setColumn(0,2,2);
   lc.setColumn(0,1,62);
   lc.setColumn(1,5,50);
   lc.setColumn(1,4,34);
   lc.setColumn(1,3,62);
   lc.setColumn(1,2,42);
   lc.setColumn(1,1,28);
   delay(800);
   lc.clearDisplay(0); //clear the display
   lc.clearDisplay(1); //clear the display
   /* Write output */
   lc.setColumn(1,1,1); digitalWrite(DATO0, HIGH);  delay(1000);
   lc.setColumn(1,1,2); digitalWrite(DATO0, LOW);  digitalWrite(DATO1, HIGH); delay(1000);
   lc.setColumn(1,1,4); digitalWrite(DATO1, LOW);  digitalWrite(DATO2, HIGH); delay(1000);
   lc.setColumn(1,1,8); digitalWrite(DATO2, LOW);  digitalWrite(DATO3, HIGH); delay(1000);
   lc.setColumn(1,1,0); digitalWrite(DATO3, LOW);
   lc.setColumn(1,0,15);
   lc.setColumn(0,7,240);
};

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

   /* Read input */
   address =  digitalRead(AD0) + (digitalRead(AD1) * 2) + (digitalRead(AD2) * 4) + (digitalRead(AD3) * 8) + (digitalRead(BANK)
* 16);
   woord =  digitalRead(DATI0) + (digitalRead(DATI1) * 2) + (digitalRead(DATI2) * 4) + (digitalRead(DATI3) * 8);

   /* Write output */
   lc.setColumn(1,1,address);
   lc.setColumn(1,2,woord);
   varia = (digitalRead(WRITE) == HIGH ? 1 : 0) + (analogRead(READ) > 500 ? 2 : 0) + (analogRead(SERI) > 500 ? 4 : 0);
   lc.setColumn(1,3,varia);
}
```

## Arduino Nano program

```
/* ULU.27 32x4-bit RAM memory - program code */
/* CC BY-NC-SA Jeroen Brinkman */

int memory[4][16]; // memory
int serad, address, lastaddress, woord; // 4-bit words
int bank, serba, d0, d1, d2, d3; // bits
bool read, write, serial, lastserial; // booleans
int i, display, column, signa, state; // integers

#include "LedControl.h"
LedControl lc=LedControl(12,11,10,2);

#define BOUNCE 8
#define HEART 1
#define DATI0 2
#define DATI1 3
#define DATI2 4
#define DATI3 5
#define AD0 6
#define AD1 7
#define AD2 8
#define AD3 9
```

```
#define BANK A0
#define DATO0 A1
#define DATO1 A2
#define DATO2 A3
#define DATO3 A4
#define WRITE  A5
#define READ  A6
#define SERI  A7

void setup() {
   pinMode(HEART, OUTPUT);
   pinMode(DATO0, OUTPUT);
   pinMode(DATO1, OUTPUT);
   pinMode(DATO2, OUTPUT);
   pinMode(DATO3, OUTPUT);
   pinMode(DATI0, INPUT);
   pinMode(DATI1, INPUT);
   pinMode(DATI2, INPUT);
   pinMode(DATI3, INPUT);
   pinMode(BANK, INPUT);
   pinMode(AD0, INPUT);
   pinMode(AD1, INPUT);
   pinMode(AD2, INPUT);
   pinMode(AD3, INPUT);
   pinMode(WRITE, INPUT);
   pinMode(READ, INPUT);
   pinMode(SERI, INPUT);

   /* Initiate dot matrix */
   lc.shutdown(0,false); //The MAX72XX is in power-saving mode on startup, wakeup the display
   lc.shutdown(1,false); //The MAX72XX is in power-saving mode on startup, wakeup the display
   lc.setIntensity(0, 1); // Set the brightness to a low value
   lc.setIntensity(1, 1); // Set the brightness to a low value
   lc.clearDisplay(0); //clear the display
   lc.clearDisplay(1); //clear the display

   /* Every ULU with an Arduino introduces itself. This one uses dot text */
   lc.setColumn(0,7,62);
   lc.setColumn(0,6,32);
   lc.setColumn(0,5,62);
   lc.setColumn(0,3,62);
   lc.setColumn(0,2,32);
   lc.setColumn(0,1,32);
   lc.setColumn(1,7,62);
   lc.setColumn(1,6,32);
   lc.setColumn(1,5,62);
   lc.setColumn(1,3,32);
   delay(800);
   lc.clearDisplay(0); //clear the display
   lc.clearDisplay(1); //clear the display
   lc.setColumn(0,7,58);
   lc.setColumn(0,6,42);
   lc.setColumn(0,5,46);
   lc.setColumn(0,3,2);
   lc.setColumn(0,2,2);
   lc.setColumn(0,1,62);
   lc.setColumn(1,5,50);
   lc.setColumn(1,4,34);
   lc.setColumn(1,3,62);
   lc.setColumn(1,2,42);
   lc.setColumn(1,1,28);
   delay(800);
   lc.clearDisplay(0); //clear the display
   lc.clearDisplay(1); //clear the display
   for (int i = 0; i < 17; i++) {memory[0][i] = 0; memory[1][i] = 0; memory[2][i] = 0; memory[3][i] = 0;} // clear array
   lastserial = false; bank = 0; address = 0; lastaddress = 0; state = 0; column = 0;
};

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

  /* Read all input */
   signa = 0;
   for (int i = 0; i < BOUNCE; i++) {signa += (analogRead(SERI) > 500 ? 1 : 0); delay(2);} // eliminate bouncing
   serial = (signa > i / 2);
   if ((state == 0) && !serial) {
      address =  digitalRead(AD0) + (digitalRead(AD1) * 2) + (digitalRead(AD2) * 4) + (digitalRead(AD3) * 8);
      bank = (digitalRead(BANK) == HIGH ? 1 : 0);
   }
   d0 =  (digitalRead(DATI0) == HIGH ? 1 : 0); d1 = (digitalRead(DATI1) == HIGH ? 1 : 0); d2 = (digitalRead(DATI2) == HIGH ? 1
: 0); d3 = (digitalRead(DATI3) == HIGH ? 1 : 0);
   write = (digitalRead(WRITE) == HIGH);
   read =  (analogRead(READ) > 500);

   /* Clear output */
   if (!write) {
      digitalWrite(DATO0, LOW);
      digitalWrite(DATO1, LOW);
      digitalWrite(DATO2, LOW);
      digitalWrite(DATO3, LOW);
   }

   /* Manage serial programming */
   // state == 0 -> regular use
   // state == 1 -> Serial programming initiated
   // state == 2 -> Serial programming in progress

   if (serial && state == 0) { // first time serial is 1, serial programming starts
      state = 1; serad = address; serba = bank; // take the actual address as a starting position
   }
   if (state > 0 && (write || read)) state = 0; // when read or write is 1, serial programming ended
   if (serial && !lastserial && (state == 1)) { // only when this condition is met, the memory may be written
      state = 2;
      address = serad; bank = serba;
      write = true;
   }
   lastserial = serial;

   if (address != lastaddress) lc.setColumn(display, column, memory[3][lastaddress]);
   lastaddress = address;

   /* Write memory */
   // memory[0] - bank 0: - - - - 3 2 1 0
   // memory[1] - bank 1: - - - - 3 2 1 0
   // memory[2] - bank 1: 7 6 5 4 - - - -
   // memory[3] - bank 1 + bank 0: 7 6 5 4 3 2 1 0
   if (write && !read) {
      if (bank == 0) {
         memory[0][address] = d0 + (d1 * 2) + (d2 * 4) + (d3 * 8);
      } else {
         memory[1][address] = d0 + (d1 * 2) + (d2 * 4) + (d3 * 8);
         memory[2][address] = (d0 * 16) + (d1 * 32) + (d2 * 64) + (d3 * 128);
      }
      memory[3][address] =  memory[2][address] + memory[0][address];
   }

   /* Post process serial programming */
   if (state == 2) { // serial programming and the memory has been written
      serad += 1;
      if (serad == 16) {
          serad = 0; serba = (serba == 1 ? 0 : 1);
      }
      state = 1;
   }

   /* Read memory */
   if (read && !write) {
      woord = memory[bank][address];
      d3 = 0; d2 = 0; d1 = 0; d0 = 0;
      if (woord > 7) {d3 = 1; woord -= 8;}
      if (woord > 3) {d2 = 1; woord -= 4;}
      if (woord > 1) {d1 = 1; woord -= 2;}
      d0 = woord;
      digitalWrite(DATO0, (d0 == 1 ) ? HIGH : LOW);
      digitalWrite(DATO1, (d1 == 1 ) ? HIGH : LOW);
      digitalWrite(DATO2, (d2 == 1 ) ? HIGH : LOW);
      digitalWrite(DATO3, (d3 == 1 ) ? HIGH : LOW);
   }

   /* Write blinking display */
   // blink bank 0 -> memory[2] and memory[3]
   // blink bank 1 -> memory[0] and memory[3]
   column = address; display = 0;
   if (address < 8) display = 1; else column -= 8;
   if (((millis() / 1000) % 2) == 1) {
      lc.setColumn(display, column, memory[3][address]); //
   } else {
      if (bank == 0) {
         lc.setColumn(display, column, memory[2][address]); //
      } else {
         lc.setColumn(display, column, memory[0][address]); //
      }
   }
}
```