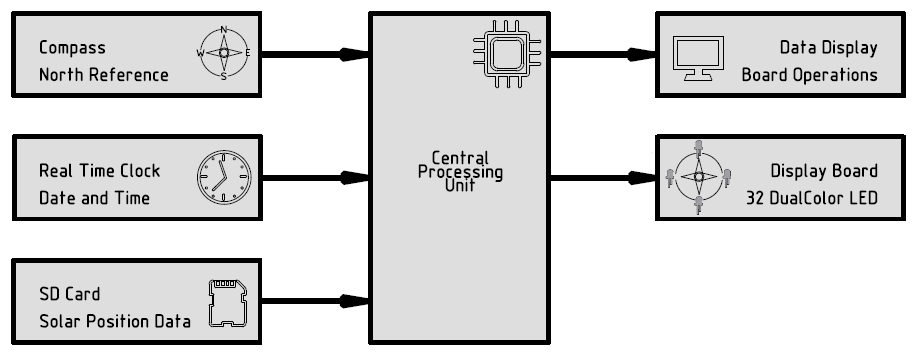
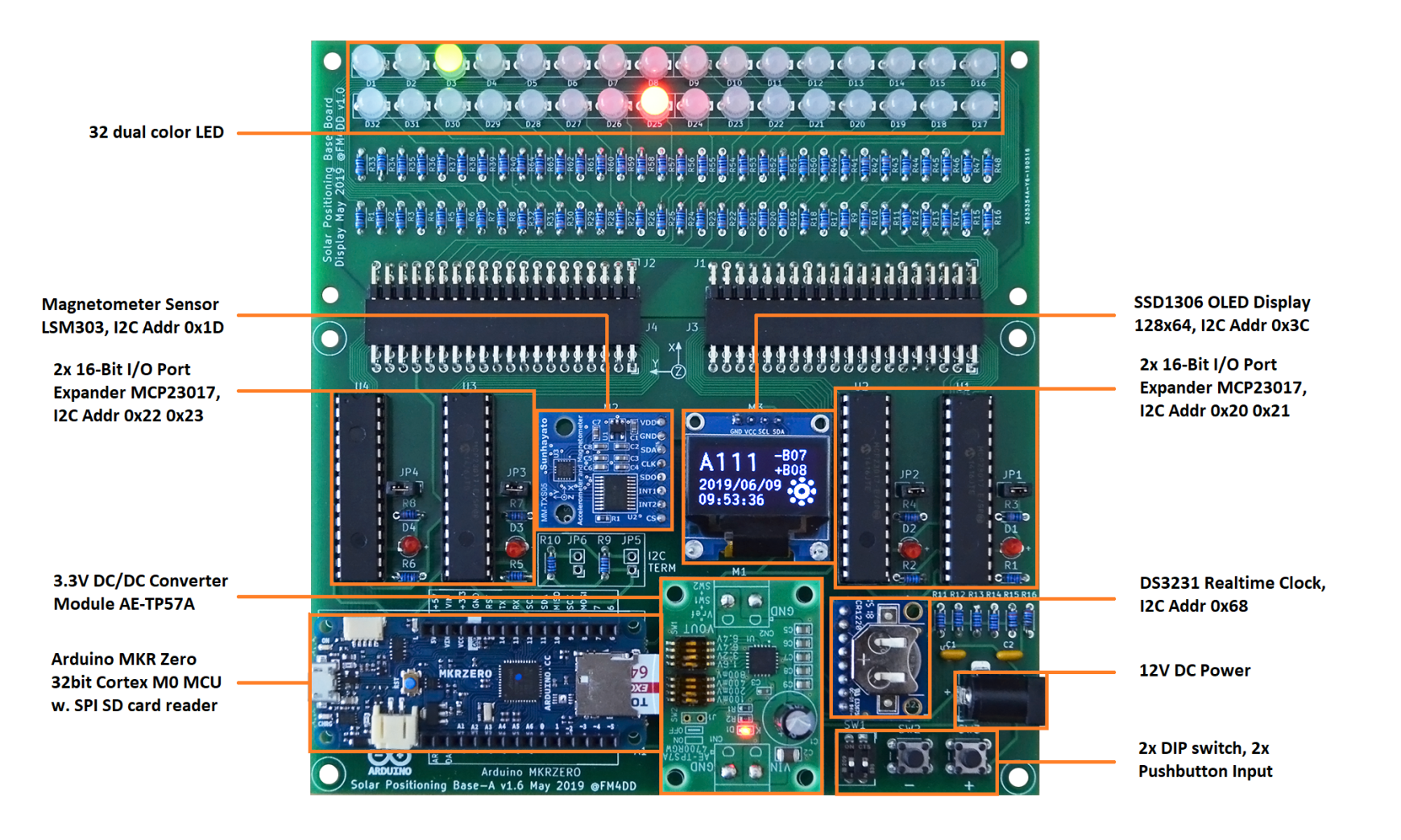
Das Suntracker2R2 ist eine 32-bit Arduino MKR Zero basierte Mikroprozessor Entwicklungsumgebung zur Ansteuerung von 64 digitalen I/O, deren Pegel durch ein Displayboard mit 32x Zweifarben-LED angezeigt werden. Als Eingangsdaten steht eine Echtzeituhr (RTC), ein Magnetometer Sensor, und der MicroSD Kartenleser des Arduino MKR Zero zur Verfuegung. Ein 128x64 OLED Display dient zur Zustands-, Daten und Messwertausgabe.

## **Board Design**



## **Komponenten Uebersicht**



Below tables show some of the DE0-Nano-SOC’s basic built-in HW components: LED, Pushbutton, DIP, and Clock.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **LED** | **FPGA PIN** |  | **Push Button** | **FPGA PIN** |  | **DIP Switch** | **FPGA PIN** |
| LED[0] | PIN\_W15 |  | KEY[0] | PIN\_AH17 |  | DIP Switch[0] | PIN\_L10 |
| LED[1] | PIN\_AA24 |  | KEY[1] | PIN\_AH16 |  | DIP Switch[1] | PIN\_L9 |
| LED[2] | PIN\_V16 |  |  |  |  | DIP Switch[2] | PIN\_H6 |
| LED[3] | PIN\_V15 |  | **Clock** | **FPGA PIN** |  | DIP Switch[3] | PIN\_H5 |
| LED[4] | PIN\_AF26 |  | FPGA\_CLK1\_50 | PIN\_V11 |  |  |  |
| LED[5] | PIN\_AE26 |  | FPGA\_CLK2\_50 | PIN\_Y13 |  |  |  |
| LED[6] | PIN\_Y16 |  |  | | | | |
| LED[7] | PIN\_AA23 |  | Pushbutton keys are level “high”, and go “low” when pressed. | | | | |
|  |  |  | DIP switches are level “high” in down position, and “low” when up. | | | | |

## **Arduino MKR Zero**

**I/O Header Pin Belegungsplan:**

Am Arduino sind 8 Pins belegt,

und 20 von 28 Pins ungenutzt.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Board Pin** | **Arduino Pin** | **#** |  | **#** | **Arduino Pin** | **Board Pin** |
| N/A | AREF | 1 |  | 28 | +5V | N/A |
| N/A | DAC0 | 2 | 27 | VIN | N/A |
| N/A | A1 | 3 | 26 | +3.3V | DC Converter + |
| N/A | A2 | 4 | 25 | GND | DC Converter - |
| N/A | A3 | 5 | 24 | RESET | N/A |
| N/A | A4 | 6 | 23 | 14 | N/A |
| N/A | A5 | 7 | 22 | 13 | N/A |
| N/A | A6 | 8 | 21 | SCL | I2C Bus Clock |
| DIP Switch 0 | 0 | 9 | 20 | SDA | I2C Bus Data |
| DIP Switch 1 | 1 | 10 | 19 | 10 | N/A |
| Taster 1 (SW2) | 2 | 11 | 18 | 9 | N/A |
| Taster 2 (SW3) | 3 | 12 | 17 | 8 | N/A |
| N/A | 4 | 13 | 16 | 7 | N/A |
| N/A | 5 | 14 | 15 | 6 | N/A |

## **DIE DIp Schalter und Taster** (Default OFF = 1, ON = 0)

Der Demo Sketch **s2r2-dipkey-test.ino** zeigt die Verwendung der Schalter und Taster:

/\* ------------------------------------------------- \*/

/\* Suntracker2R2 DIP and Pushbutton Test Sketch \*/

/\* ------------------------------------------------- \*/

#include "Arduino.h" // default library

#define DIP1 0 // dip switch 1 <-> IO-pin 0

#define DIP2 1 // dip switch 2 <-> IO-pin 1

#define KEY1 2 // pushbutton 1 <-> IO-pin 2

#define KEY2 3 // pushbutton 2 <-> IO-pin 3

uint8\_t dippos1 = 1; // dip switch 1 position

uint8\_t dippos2 = 1; // dip switch 2 position

uint8\_t push1 = 1; // pushbutton 1 position

uint8\_t push2 = 1; // pushbutton 2 position

void setup() {

Serial.begin(115200);

while (!Serial); // wait for serial port to connect.

Serial.println("Serial OK");

/\* ------------------------------------------------- \*/

/\* Set dip switch and push button IO pins for input \*/

/\* ------------------------------------------------- \*/

pinMode(DIP1, INPUT);

pinMode(DIP2, INPUT);

pinMode(KEY1, INPUT);

pinMode(KEY2, INPUT);

pinMode(push1, INPUT);

pinMode(push2, INPUT);

Serial.println("DIP and KEY pins set for input.");

}

void loop() {

/\* ------------------------------------------------- \*/

/\* Get and display dip switch and pushbutton status \*/

/\* ------------------------------------------------- \*/

Serial.print(" DIP1: "); dippos1 = digitalRead(DIP1); Serial.print(dippos1);

Serial.print(" DIP2: "); dippos2 = digitalRead(DIP2); Serial.print(dippos2);

Serial.print(" KEY1: "); push1 = digitalRead(KEY1); Serial.print(push1);

Serial.print(" KEY1: "); push2 = digitalRead(KEY2); Serial.print(push2);

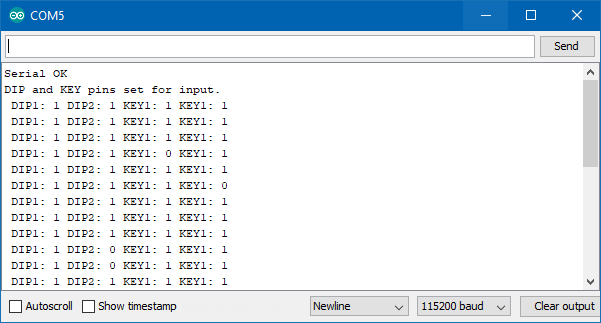
Serial.println();

delay(500);

}

\*Nicht Vergessen den Seriellen Monitor in der Arduino IDE zu oeffnen, sonst beginnt das Program nicht.

Serielle Ausgabe:



* Beachte die invertierte Logik: Ein Tastendruck oder Switch ON generiert ein ‘0’ Signal.

## **Der I2C BUS**

Auf dem Mainboard verbindet der I2C Bus die acht folgenden Geraete mit dem Arduino:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Geraet** | **Typ** | **Hersteller** | **I2C Bus Addresse (Hex)** |
| 1 | Accelerometer | LSM303DLHC | Adafruit | 0x19 |
| 2 | Magnetometer | See above | See above | 0x1E |
| 3 | IO Expander 1 | MCP23017 E/SP | Microchip | 0x20 |
| 4 | IO Expander 2 | MCP23017 E/SP | Microchip | 0x21 |
| 5 | IO Expander 3 | MCP23017 E/SP | Microchip | 0x22 |
| 6 | IO Expander 4 | MCP23017 E/SP | Microchip | 0x23 |
| 7 | OLED Display | SSD1306 | Unbranded | 0x3C |
| 8 | RTC Clock | DS3231 | Adafruit | 0x68 |

Der Demo Sketch **s2r2-i2cscan-test.ino** ueberprueft ob die Gerate am I2C Bus gefunden werden:

/\* ------------------------------------------------- \*/

/\* Suntracker2R2 12C Bus Scanner Test Sketch \*/

/\* ------------------------------------------------- \*/

#include <Wire.h> // Arduino built-in I2C bus function library

uint8\_t error = 0; // I2C commuications error

uint8\_t address = 0; // I2C device address

uint8\_t counter = 0; // I2C device counter

void setup(){

Serial.begin(115200);

while (!Serial); // wait for serial port to connect.

Serial.println("Serial OK. Scanning I2C Bus:");

Wire.begin(); // enable I2C bus, default speed

for(address = 0; address < 127; address++ ) {

Serial.print(".");

Wire.beginTransmission(address);

error = Wire.endTransmission();

if (error == 0){

Serial.print("\nDevice found: 0x");

Serial.println(address,HEX);

counter++;

continue;

}

if (error == 4) {

Serial.print("\nDevice error: 0x");

Serial.println(address,HEX);

continue;

}

delay(100);

}

Serial.print("\n I2C bus scan result: ");

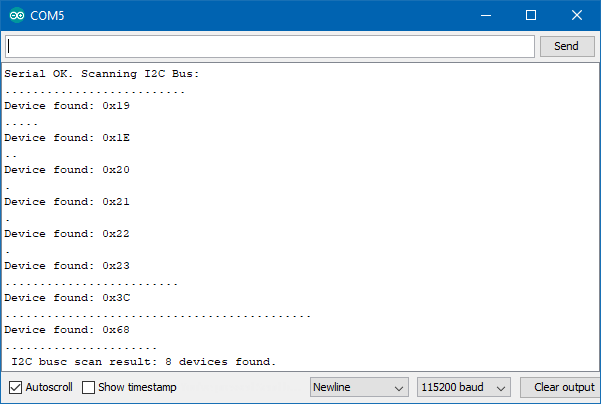
Serial.print(counter);

Serial.print(" devices found.");

}

void loop() {}

Serielle Ausgabe:



## **Der IO Expander MCP23017**

Der IO Expander Chip MCP23017 von Microchip stellt 16 digitale Pins via I2C Bus zur Verfuegung. Die IO Pins koennen als Input oder Output gesetzt werden. Durch drei Addressleitungen am Expander Chip kann man bis zu acht Expander an einem I2C Bus betreiben, und bekommt damit maximal 128 IO. Das Suntracker2 R2 Mainboard hat vier Expander (U1-U4), deren 64 IO Pins auf zwei 40-Pin Expansion Header (J3, J4) gelegt sind. Zur optischen Statuskontrolle und zum schnellen Debug hat das Mainboard vier rote Status LEDs (D1-D4), die jeweils auf den ersten IO Pin des jeweiligen IO Expanders geschalten sind. Diese Status LED kann man bei Bedarf durch Entfernen von Jumper JP1-JP4 einzeln Abschalten, um zum Beispiel diesen Pin auch als Input Pin zu betreiben zu koennen.

Durch Anstecken der optionalen Displayboard Erweiterung werden die 64 IO Pins vom Development Mainboard mit den 32 Zweifarben-LED (rot/gruen) auf der Displayboard Erweiterung verbunden. Fuer die Displayboard Erweiterung werden dazu alle IO Pins der vier Expander im Output Mode betrieben.

Der Demo Sketch **s2r2-mcp23017-test1.ino** blinkt die rote Status LED (D1) des ersten IO Expanders (U1). Ist die Displayboard Erweiterung angeschlossen, blinkt auch die LED D1 des Displayboards in rot mit der Status LED mit.

/\* ------------------------------------------------- \*/

/\* Suntracker2R2 1. IO Expander MCP23017 Test Sketch \*/

/\* ------------------------------------------------- \*/

#include <Wire.h> // Arduino built-in I2C bus function library

#include <Adafruit\_MCP23017.h> // https://github.com/adafruit/Adafruit-MCP23017-Arduino-Library

Adafruit\_MCP23017 mcp1; // create object for 1st MCP23017

void setup() {

Serial.begin(115200);

while (!Serial); // wait for serial port to connect.

Serial.println("Serial OK");

Wire.begin(); // enable I2c bus

mcp1.begin(); // enable 1st MCP23017 expander chip

mcp1.pinMode(0, OUTPUT); // set 1st expander pin-0 as output

mcp1.digitalWrite(0, LOW); // set 1st expander pin-0 to '0'

}

void loop() {

mcp1.digitalWrite(0, HIGH); // set 1st expander pin-0 to '1'

Serial.print("1st MCP23017 pin-0 state: ");

Serial.print(mcp1.digitalRead(0));

delay(500);

mcp1.digitalWrite(0, LOW); // set 1st expander pin-0 to '0'

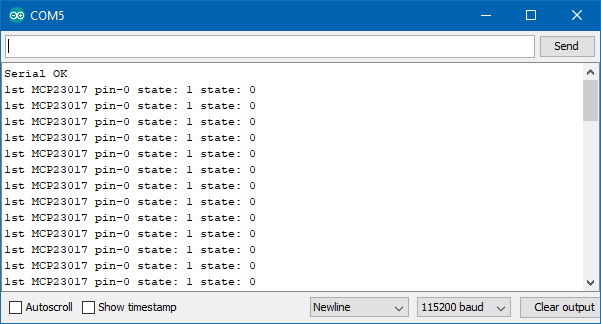
Serial.print(" state: ");

Serial.println(mcp1.digitalRead(0));

delay(500);

}

Serielle Ausgabe:



**Achtung:** Zur Ansteuerung des MCP23017 benutzt dieser Demo Sketch die externe Adafruit Funktionsbibliothek fuer den MCP23017 Chip. Externe Bibliotheken werden in der Ardunio IDE durch den Menuepunkt “Tools” 🡪 “Manage Libraries” hinzugefuegt. Ist man Internet-verbunden, gibt man dort im Suchfenster den Begriff “MCP23017” ein. Daraufhin sollten mehrere Bibliotheken vorgeschlagen werden, eine davon ist die hier verwendete Bibliothek, erkennbar am Namen “Adafruit”. Diese soll vor dem Kompilieren des Sketches installiert werden.

**Aufgabe-1:**

Erstelle einen Sketch **s2r2-key1led1-test.ino**, worin das Druecken eines Pushbuttons die Status LED des ersten IO Expanders aufleuchten laesst. Kombiniere dazu Code aus dem Demo program **s2r2-dipkey-test.ino** mit dem Code aus dem Demo Program **s2r2-mcp23017-test.ino.**

Wenn man die weiteren 15 Pins des ersten IO Expanders als Output, und auf ‘HIGH’ setzt, steuert man damit die 16 LED der oberen Reihe auf der Displayboard Erweiterung mit der Farbe rot an.

**Displayboard LED Belegungsplan:**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **IO Expander** | **LED Gruppe** | **Farbe** |
| 1 | U1 pin 0-15 | D1-D16 (oben) | rot |
| 2 | U2 pin 0-15 | D17-D32 (unten) | rot |
| 3 | U3 pin 0-15 | D1-D16 (oben) | gruen |
| 4 | U4 pin 0-15 | D17-32 (unten) | gruen |

**Ansteuerung von mehreren IO Expandern:**

Das naechste Beispiel **s2r2-mcp23017-test4.ino** zeigt die Ansteuerung des vierten IO Expanders (U4). Wir schalten die untere LED Reihe (D17-32) gruen:

/\* ------------------------------------------------- \*/

/\* Suntracker2R2 4. IO Expander MCP23017 Test Sketch \*/

/\* ------------------------------------------------- \*/

#include <Wire.h> // Arduino built-in I2C bus function library

#include <Adafruit\_MCP23017.h> // https://github.com/adafruit/Adafruit-MCP23017-Arduino-Library

Adafruit\_MCP23017 mcp4; // create object for 4th MCP23017

uint8\_t pin = 0; // pin counter to enable LED

void setup() {

Wire.begin(); // enable I2c bus

mcp4.begin(3); // enable 4th MCP23017 expander chip

for (pin = 0; pin<16; pin++) { // run through all 16 pins on the expander

mcp4.pinMode(pin, OUTPUT); // set expander pin as output

mcp4.digitalWrite(pin, HIGH); // set expander pin to '1' (LED ON)

}

}

void loop() {}