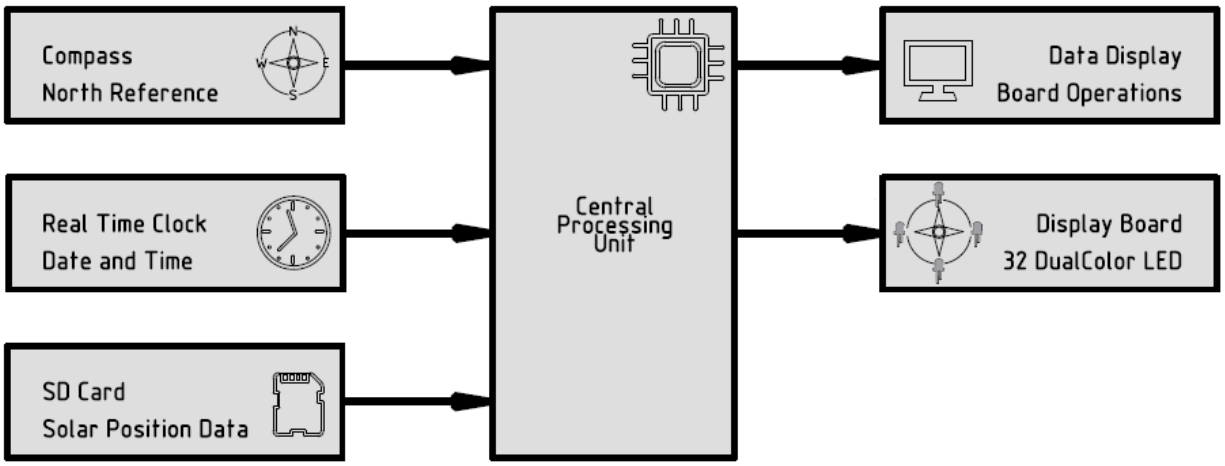
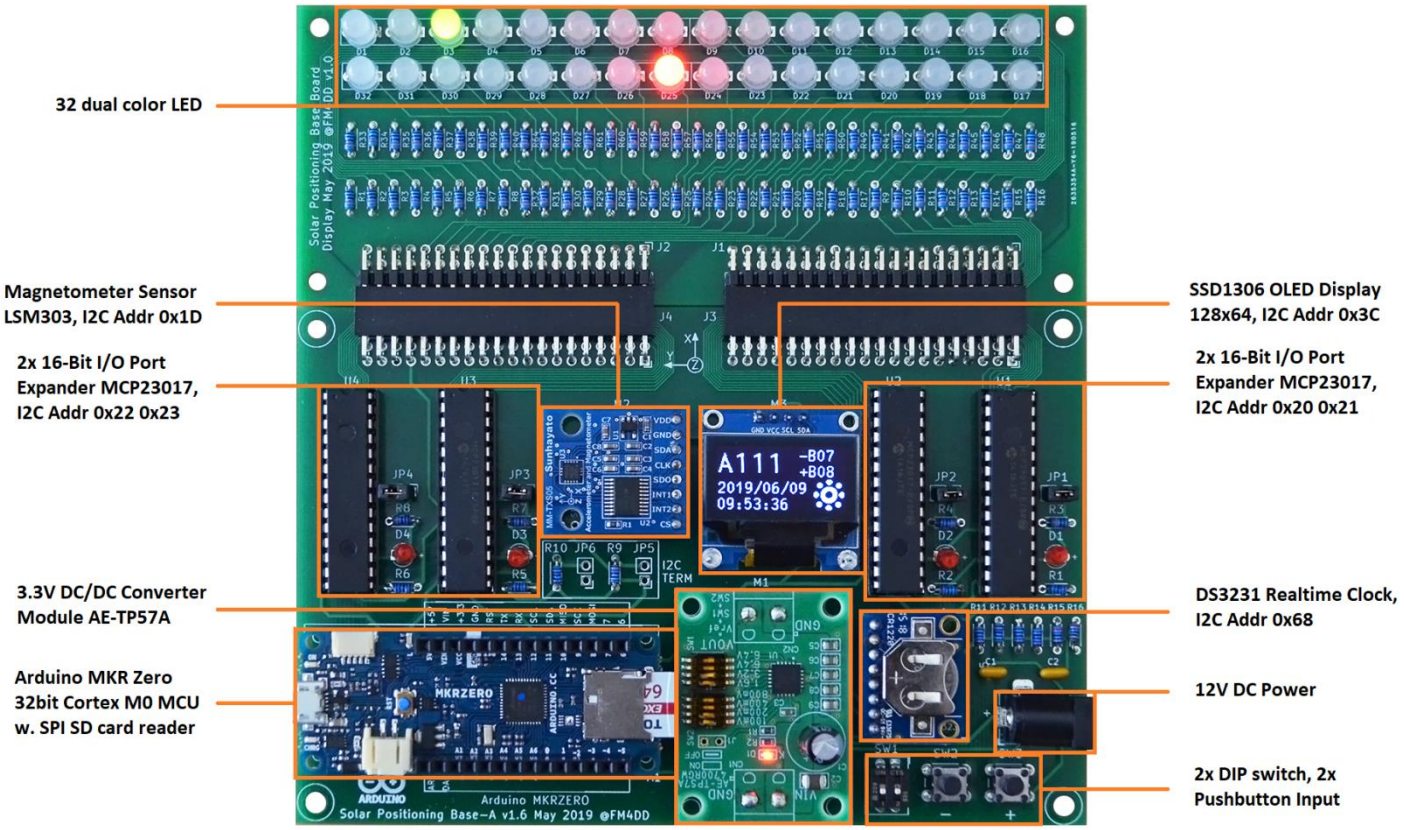
Suntracker2R2 Development Board Manual

Das Suntracker2R2 ist eine 32-bit Arduino MKR Zero basierte Mikroprozessor Entwicklungsumgebung zur Ansteuerung von 64 digitalen I/O. Damit kann man zum Beispiel eine Displayboard Erweiterung mit 32x Zweifarben-LED zur Datenausgabe ansteuern. 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

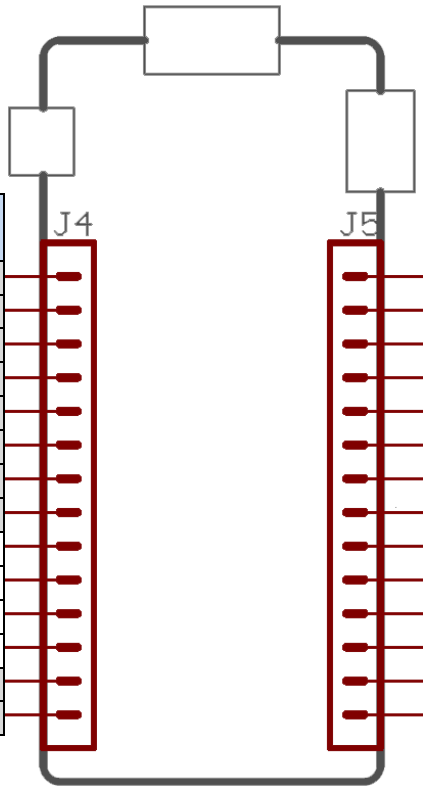


ARDUINO MKR ZERO

Herzstück des Boards ist der Arduino MKR Zero, eine Cortex M0 basierte 32bit CPU mit 32K RAM und 256K Flash Speicher.

I/O Header Pin Belegungsplan:

Am Arduino sind 8 Pins belegt, und 20 von 28 Pins ungenutzt.



Board Pin	Arduino Pin	#
N/A	AREF	1
N/A	DAC0	2
N/A	A1	3
N/A	A2	4
N/A	A3	5
N/A	A4	6
N/A	A5	7
N/A	A6	8
DIP Switch 0	0	9
DIP Switch 1	1	10
Taster 1 (SW2)	2	11
Taster 2 (SW3)	3	12
N/A	4	13
N/A	5	14

#	Arduino Pin	Board Pin
28	+5V	N/A
27	VIN	N/A
26	+3.3V	DC Converter +
25	GND	DC Converter -
24	RESET	N/A
23	14	N/A
22	13	N/A
21	SCL	I2C Bus Clock
20	SDA	I2C Bus Data
19	10	N/A
18	9	N/A
17	8	N/A
16	7	N/A
15	6	N/A

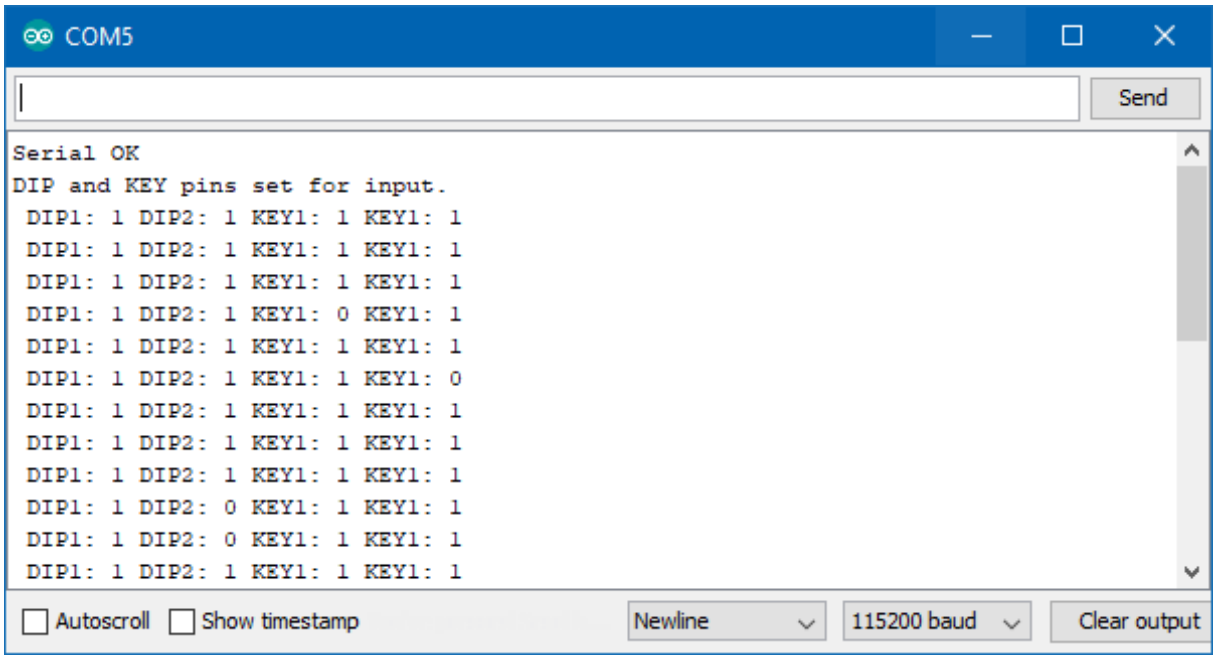
DIE DIP SCHALTER UND TASTER (DEFAULT OFF = 1, ON = 0)

Das Board ist mit zwei DIP Schaltern und zwei Tastern ausgestattet. Diese können zur Steuerung von Programmen und Ausgaben frei belegt werden. Der folgende Demo Sketch **s2r2-dipkey-test.ino** zeigt die Ansteuerung:

```
/* ------------------------------------------------ */
/* 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);
}
```

Die Ausgabe erfolgt hier über den seriellen Monitor der Arduino IDE. Dieser sollte in der Arduino IDE gestartet werden, sonst beginnt das Program nicht. Hier ist ein Beispiel-Screenshot der seriellen 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 der Arduino MCU:

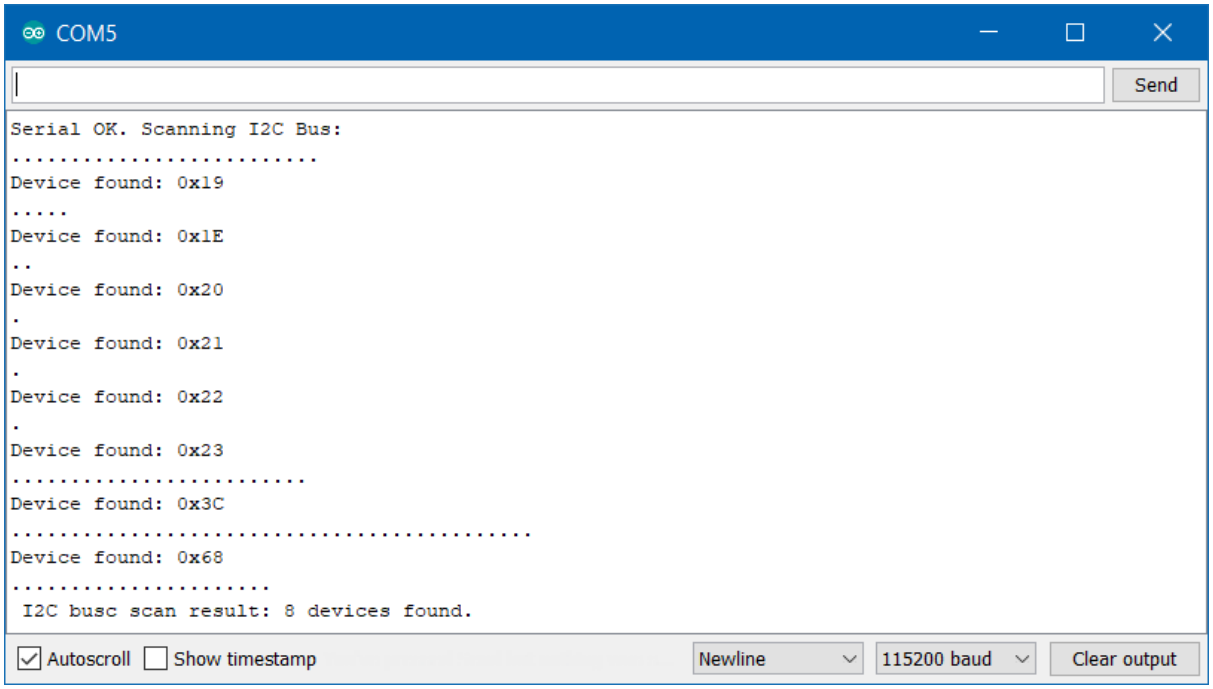
#	Geraet	Typ	Hersteller	I2C Bus Addresse (Hex)
1	Accelerometer	LSM303DLHC	Adafruit	0x19
2	Magnetometer	See above	See above	0x1E
3	IO Expander 1	MCP23017 E/SP	Microchip	0x20
4	IO Expander 2	MCP23017 E/SP	Microchip	0x21
5	IO Expander 3	MCP23017 E/SP	Microchip	0x22
6	IO Expander 4	MCP23017 E/SP	Microchip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

Der Demo Sketch **s2r2-i2cscan-test.ino** ueberprüft, ob alle 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

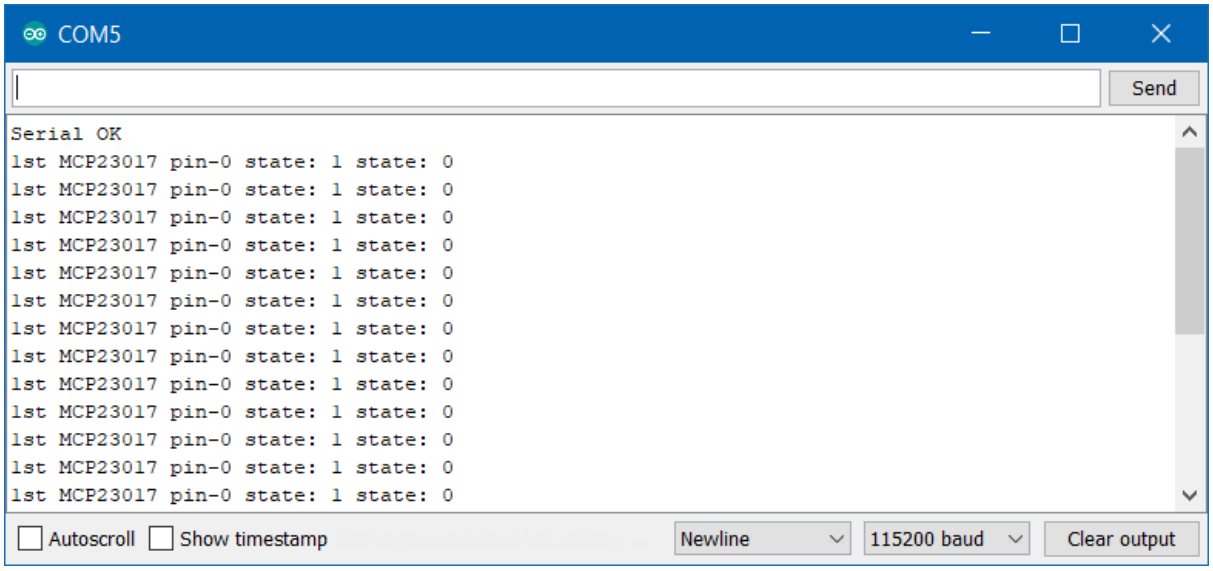
Der IO Expander Chip MCP23017 von Microchip stellt 16 digitale Pins via I2C Bus zur Verfügung. Diese IO Pins koennen als Input oder Output gesetzt werden. Durch drei Addressleitungen am Expander Chip kann man bis zu acht Expander an einem I2C Bus betreiben, und bekommt damit maximal 128 IO. Das Suntracker2 R2 Mainboard hat vier Expander (U1-U4), deren 64 IO Pins auf zwei 40-Pin Expansion Steckerleisten (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 betreiben zu können.

Durch Anstecken der optionalen Displayboard Erweiterung werden die 64 IO Pins vom Development Mainboard mit den 32 Zweifarben-LED (rot/grün) auf der Displayboard Erweiterung verbunden. Fuer die Displayboard Erweiterung werden dazu alle Pins der vier IO 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 eine externe Funktionsbibliothek für den MCP23017 Chip von Adafruit. Externe Bibliotheken werden in der Ardunio IDE durch den Menüpunkt “Tools” → “Manage Libraries” hinzugefuegt. Ist man mit dem Internet verbunden, gibt man unter diesem Menüpunkt im Suchfenster den Begriff “MCP23017” ein. Daraufhin sollten durch die Suche mehrere Bibliotheken vorgeschlagen werden. Eine davon ist die hier verwendete Bibliothek, erkennbar am Namen “Adafruit”. Diese installiert man vor dem Kompilieren des Sketches.

Aufgabe: Erstelle einen Sketch **s2r2-key1led1-test.ino**, worin das Drücken 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.**

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

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

Ansteuerung von mehreren IO Expandern:

Das nächste Beispiel **s2r2-mcp23017-test2.ino** zeigt die Ansteuerung des vierten IO Expanders (U4). Durch Ihn schalten wir hier die untere LED Reihe (D17-32) der Displayboard Erweiterung mit grüner Farbe an.

```
/* ------------------------------------------------ */
/* 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() {}
```

Das letzte Beispiel **s2r2-mcp23017-test3.ino** lässt die rote LED auf dem Displayboard im Kreis wandern:

```
/* ------------------------------------------------ */
/* Suntracker2R2 Wandering LED 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
Adafruit_MCP23017 mcp2;        // create object for 2nd MCP23017
uint8_t pin  = 0;              // pin counter to enable LED
uint8_t led  = 0;              // led counter

void setup() {
   Wire.begin();                    // enable I2c bus
   mcp1.begin(0);                   // enable 1st MCP23017 expander chip
   mcp2.begin(1);                   // enable 2nd MCP23017 expander chip

   for (pin=0; pin<16; pin++) {     // run through all 16 pins on the expander
      mcp1.pinMode(pin, OUTPUT);    // set expander pin as output
      mcp1.digitalWrite(pin, LOW);  // set expander pin to '0' (LED OFF)
      mcp2.pinMode(pin, OUTPUT);    // set expander pin as output
      mcp2.digitalWrite(pin, LOW);  // set expander pin to '0' (LED OFF)
   }
}

void loop() {
   for (led=0; led<32; led++) {         // run through all 32 led
      if(led < 16) {                    // for led 0-15, work on mcp1
      pin = led;                        // pin number equals led
      mcp1.digitalWrite(pin, HIGH);     // set expander pin to '1' (LED ON)
      if(pin == 0)                      // if we are on pin 0
         mcp2.digitalWrite(15, LOW);    // turn off the last LED on mcp2
      else
         mcp1.digitalWrite(pin-1, LOW); // turn off the previous LED
```

```
      }
      else {                            // we are on led 16 or above
         pin = led-16;                  // pin number equals led minus 16
         mcp2.digitalWrite(pin, HIGH);  // set expander pin to '1' (LED ON)
      if(pin == 0)                      // if we are on pin 0
         mcp1.digitalWrite(15, LOW);    // turn off the last LED on mcp1
      else
         mcp2.digitalWrite(pin-1, LOW); // turn off the previous LED
      }
      delay(60);
   }
}
```

DAS OLED DISPLAY MIT TEXTAUSGABE

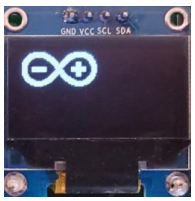
Das monochrome OLED Display hat eine Aufloesung von 128x64 Pixel. Es hat eine prima Leuchtstärke und ist mit einer passenden Bibliothek relativ einfach anzusteuern. Der Grafik-Chip ist ein SSD1306 von SOLOMON SYSTECH. Das Display wird ebenfalls ueber den I2C Bus angesprochen, unter der I2C Addresse 0x3C.
Zur Programmierung verwende ich die U8G2 Bibliothek von Oli Kraus. Die Dokumentation gibt es unter https://github.com/olikraus/u8g2.

Beim Installieren der Bibliothek (Ardunio IDE, “Tools” → “Manage Libraries”) wird auch eine Reihe von Demo Sketches bereitgestellt. Hier ist eine verkürzte Variante für den schnellen Erfolg zum Anzeigen von “Hello World”.

Der Beispielsketch **s2r2-display-test.ino** schreibt “Hello World” auf das Display:

```
/* ------------------------------------------------ */
/* Suntracker2R2 4SSD1306 OLED Display Test Sketch  */
/* ------------------------------------------------ */
#include <Arduino.h>           // Arduino default library
#include <U8g2lib.h>           // https://github.com/olikraus/u8g2

U8G2_SSD1306_128X64_NONAME_F_HW_I2C oled1(U8G2_R0);

void setup() {
   oled1.begin();
   oled1.setFont(u8g2_font_t0_17_mr);
   oled1.drawStr(0, 14, "Hello World");
   oled1.sendBuffer();
}
void loop() {}
```

DAS OLED DISPLAY MIT GRAFIKAUSGABE



Das Display kann auch wunderbar Pixelgrafiken anzeigen. Dazu müssen wir aber erstmal eine geignete Bilddatei in das unterstützte XBPM (.xbm) Format umwandeln. Es gibt verschiedene Online Konvertierseiten und Tools dafür. Hier zum Beispiel unter der URL https://onlineconvertfree.com/convert-format/bmp-to-xbm/

Der Beispielsketch **s2r2-graphic-test.ino** zeigt das Arduino Logo:

```
/* ------------------------------------------------ */
/* Suntracker2R2 4SSD1306 OLED Graphics Test Sketch */
/* ------------------------------------------------ */
#include <Wire.h>
#include <U8g2lib.h> // https://github.com/olikraus/u8g2

U8G2_SSD1306_128X64_NONAME_F_HW_I2C oled1(U8G2_R0);

// 'arduino logo', 64x32px
static const unsigned char myLogo[] U8X8_PROGMEM = {
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0xFC, 0x0F, 0x00, 0x00, 0xF0, 0x3F, 0x00,
  0x80, 0xFF, 0x3F, 0x00, 0x00, 0xFE, 0xFF, 0x00, 0xC0, 0xFF, 0xFF, 0x00,
  0x80, 0xFF, 0xFF, 0x03, 0xE0, 0xFF, 0xFF, 0x01, 0xC0, 0xFF, 0xFF, 0x07,
  0xF0, 0x07, 0xF8, 0x07, 0xE0, 0x1F, 0xE0, 0x0F, 0xF8, 0x01, 0xE0, 0x0F,
```

```
  0xF0, 0x07, 0x80, 0x0F, 0xFC, 0x00, 0xC0, 0x1F, 0xF8, 0x01, 0x00, 0x1F,
  0x7C, 0x00, 0x00, 0x3F, 0xFC, 0x00, 0x00, 0x3E, 0x3E, 0x00, 0x00, 0x7E,
  0x7E, 0xC0, 0x03, 0x3E, 0x3E, 0x00, 0x00, 0x7C, 0x3F, 0xC0, 0x03, 0x3C,
  0x1E, 0x00, 0x00, 0xF8, 0x1F, 0xC0, 0x03, 0x7C, 0x1E, 0x00, 0x00, 0xF0,
  0x0F, 0xC0, 0x03, 0x78, 0x1E, 0xFE, 0x1F, 0xF0, 0x07, 0xFC, 0x3F, 0x78,
  0x1E, 0xFE, 0x1F, 0xE0, 0x03, 0xFC, 0x3F, 0x78, 0x1E, 0xFE, 0x1F, 0xE0,
  0x07, 0xFC, 0x3F, 0x78, 0x1E, 0xFE, 0x1F, 0xF0, 0x0F, 0xFC, 0x3F, 0x78,
  0x1E, 0x00, 0x00, 0xF8, 0x0F, 0xC0, 0x03, 0x78, 0x1E, 0x00, 0x00, 0xFC,
  0x1F, 0xC0, 0x03, 0x3C, 0x3E, 0x00, 0x00, 0x7E, 0x3E, 0xC0, 0x03, 0x3C,
  0x3C, 0x00, 0x00, 0x3F, 0x7E, 0xC0, 0x03, 0x3E, 0x7C, 0x00, 0x80, 0x1F,
  0xFC, 0x01, 0x00, 0x1F, 0xF8, 0x00, 0xC0, 0x0F, 0xF8, 0x03, 0x80, 0x1F,
  0xF8, 0x03, 0xF0, 0x07, 0xF0, 0x0F, 0xC0, 0x0F, 0xF0, 0x0F, 0xFC, 0x03,
  0xE0, 0x3F, 0xF0, 0x07, 0xE0, 0xFF, 0xFF, 0x01, 0x80, 0xFF, 0xFF, 0x03,
  0x80, 0xFF, 0x7F, 0x00, 0x00, 0xFF, 0xFF, 0x01, 0x00, 0xFF, 0x1F, 0x00,
  0x00, 0xFC, 0x7F, 0x00, 0x00, 0xF8, 0x03, 0x00, 0x00, 0xE0, 0x0F, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, };

void setup() {
  Wire.begin();          /* Enable I2C bus library     */
  oled1.begin();         /* Enable the OLED module     */
  oled1.clearDisplay();
  oled1.drawXBMP(0,0, 64, 32, myLogo);
  oled1.sendBuffer();
}

void loop() {}
```

DIE ECHTZEITUHR DS3231

Die Echtzeituhr bekommt die Zeit einmal eingestellt, und mittels der 1220 Knopfbatterie wird die Zeit ständig aktualisiert. Der Arduino holt sich dann diese Zeit mittels Abfragen ueber den I2C Bus zur Geräteadresse der Uhr. Diese wird unter 0x68 angesprochen. Für das Uhrmodul gibt es mehrere Bibliotheken. Ich benutze die uRTCLib, deren Urpsrung sich hier befindet: https://github.com/Naguissa/uRTCLib

Der Beispielsketch **s2r2-ds3231-test.ino** zeigt wie man die Zeit setzt, abruft und auf das OLED displaz schreibt:

```
/* ------------------------------------------------ */
/* Suntracker2R2 DS3231 Realtime Clock Test Sketch   */
/* ------------------------------------------------ */
#include <Arduino.h>
#include "uRTCLib.h"              // https://github.com/Naguissa/uRTCLib
#include <U8g2lib.h>              // https://github.com/olikraus/u8g2

U8G2_SSD1306_128X64_NONAME_F_HW_I2C oled1(U8G2_R0);
uRTCLib rtc;                      // realtime clock object
char lineStr[15];                 // display output buffer, 14-chars + \0

void setup() {
   Wire.begin();                  // enable I2C bus
   rtc.set_rtc_address(0x68);     // Enable realtime clock
   rtc.set_model(URTCLIB_MODEL_DS3231);
   // rtc.set sec,min,hour,dayOfWeek,dayOfMonth,month,year
   // to set the time, enable the line below
   //rtc.set(0, 51, 0, 0, 16, 6, 19);
   oled1.begin();                 // enable oled display
   oled1.setFont(u8g2_font_t0_17_mr);
}

void loop() {
   rtc.refresh();                 // get new time from clock
   snprintf(lineStr, sizeof(lineStr), "%d/%02d/%02d", rtc.year(),rtc.month(),rtc.day());
   oled1.drawStr(0, 15, lineStr);
   snprintf(lineStr, sizeof(lineStr), "%02d:%02d:%02d", rtc.hour(),rtc.minute(),rtc.second());
   oled1.drawStr(0, 30, lineStr);
   oled1.sendBuffer();           // update oled display
}
```

DER MAGNETSENSOR

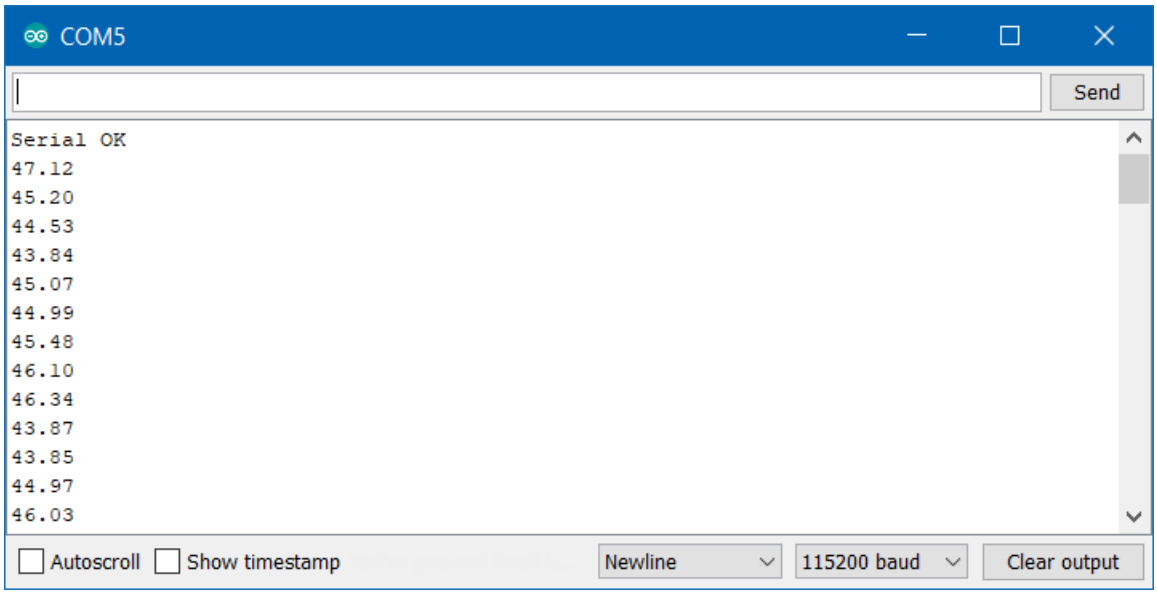
Der Magnetsensor (Magnetometer) ist ein LSM303DLHC Modul von Adafruit. Er misst das Magnetfeld der Erde über drei Achsen. Das Erdmagnetfeld ist einerseits sehr schwach, und an jedem Ort unterschiedlich ausgeprägt. Deswegen erreicht man gute Ergebnisse nur mit Kalibrierung des Sensors, wobei er speziell auf die örtlichen Gegebenheiten eingestellt wird. Ich benutze hier nicht die Adafruit Bibliothek, sondern die Version von Pololu.

https://github.com/pololu/lsm303-arduino

Der Beispielsketch **s2r2-lsm303-test.ino** zeigt wie man eine Richtungsanzeige (Heading) abruft:

```
/* ------------------------------------------------ */
/* Suntracker2R2 LSM303 Magnetometer Test Sketch    */
/* ------------------------------------------------ */
#include <Wire.h>
#include <LSM303.h>

LSM303 compass;

void setup() {
   Serial.begin(115200);
   while (!Serial);   // wait for serial port to connect.
   Serial.println("Serial OK");

   Wire.begin();
   compass.init();
   compass.enableDefault();
   /* Default Calibration values of +/-32767 for each axis */
   compass.m_min = (LSM303::vector<int16_t>){-32767, -32767, -32767};
   compass.m_max = (LSM303::vector<int16_t>){+32767, +32767, +32767};
}

void loop() {
   compass.read();
   float heading = compass.heading();
   Serial.println(heading);
   delay(100);
}
```

Serielle Ausgabe:



Zur Sensor Kalibrierung verwende ich einfach den mitgelieferten Beispielsketch unter “File” → “Examples” →”LSM303” → “Calibrate”. Mit den dabei ermittelten Werten ersetzt man dann die default Werte von 32767.

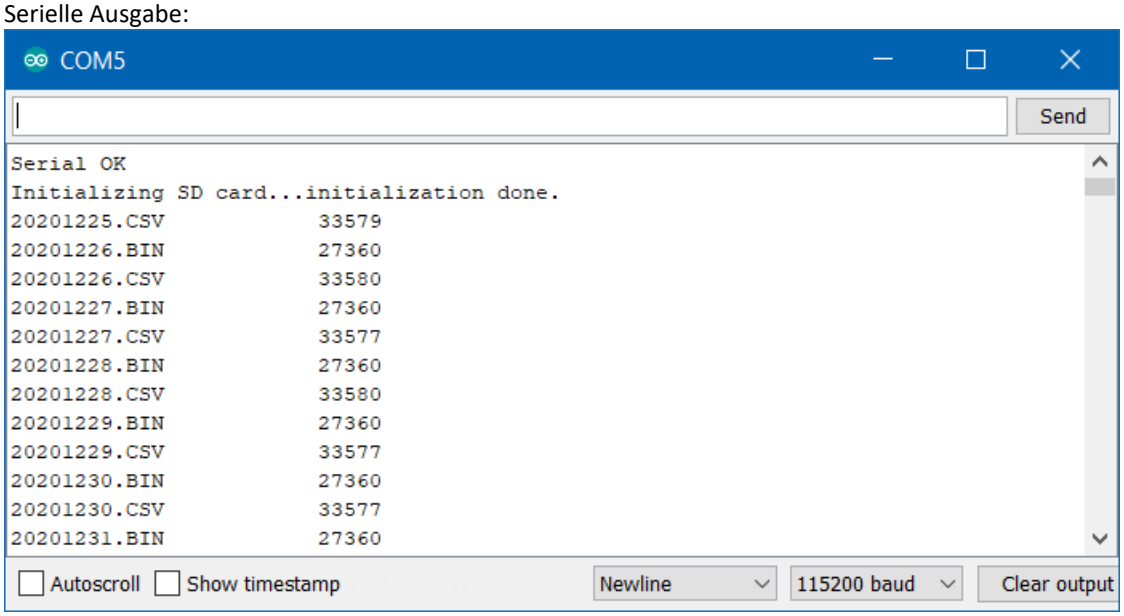
DER MICRO-SD KARTENLESER

Der in das Arduino Board eingebaute Micro-SD Karteneinschub ist das einzige Gerät was nicht ueber den I2C Bus angesprochen wird. Es benutzt stattdessen den SPI Bus. Mit dem Kartenleser kann man normale FAT16 und FAT32 formatierte Mico SD Karten mit kleinen Einschränkungen Lesen und Schreiben. Die Bibliothek ist diesmal schon mit der Arduino IDE dabei, man muss sie nur im Programm einbinden.

Der Beispielsketch **s2r2-sdcard-test.ino** zeigt die gespeicherten Files auf der Karte an:

```
/* ------------------------------------------------ */
/* Suntracker2R2 MKR Zero SD Kartenleser Test Sketch */
/* ------------------------------------------------ */
#include <SPI.h>
#include <SD.h>

File root;

void setup() {
   Serial.begin(115200);
   while (!Serial);   // wait for serial port to connect.
   Serial.println("Serial OK");

   Serial.print("Initializing SD card...");
   if (!SD.begin(SDCARD_SS_PIN)) {
      Serial.println("initialization failed!");
      while(1);
   }
   Serial.println("initialization done.");
   root = SD.open("/");
   printDirectory(root, 0);
   Serial.println("done!");
}

void loop() {}

void printDirectory(File dir, int numTabs) {
   while (true) {
      File entry =  dir.openNextFile();
      if (! entry) break;     // no more files
      for (uint8_t i = 0; i < numTabs; i++) {
         Serial.print('\t');
      }
      Serial.print(entry.name());
      if (entry.isDirectory()) {
         Serial.println("/");
         printDirectory(entry, numTabs + 1);
      } else {              // show file size
         Serial.print("\t\t");
         Serial.println(entry.size(), DEC);
      }
      entry.close();
   }
}
```

Serielle Ausgabe:



WEITERE INFORMATIONEN

Mit den Komponenten auf dem Mainboard und dem Displayboard kann man zum Beispiel einen Kompass bauen, LED Muster erzeugen, oder die Uhrzeit anzeigen. Mit dem 2x40 Pin Erweiterungskartensteckplatz kann man andere Erweiterungskarten entwickeln, und so andere digitale Module einzubinden. Diese können sowohl Dateninput liefern, oder den Output vom Arduino verarbeiten. Auf dem Mainboard sind die meisten Module gesockelt, dadurch kann man Sie einfach ersetzen oder woanders weiterverwenden.

Die Leiterplatten und Schaltplaene des Mainboards und des Displayboards sind frei auf Github verfügbar unter: https://github.com/fm4dd/suntracker2-r2

Viel Spass!