

## Übung 1-4 – Übermitteln von CAN-Daten

Über einen **Thermistor** (Heißleiter, NTC (MF52-103 3435)) soll die **Temperatur gemessen** werden. Dabei wird der Spannungsabfall über den temperaturabhängigen Widerstand mithilfe eines Spannungsteilers gemessen.

Die gemessene Temperatur soll auf dem **CAN übertragen** werden. Dafür wird das **Standalone-CAN-Board** mit dem **CAN-Controller MCP2515** über den **SPI-Bus** an den Arduino angeschlossen.

### Schaltungsaufbau

- Spannungsteiler: Vorwiderstand ( $10\text{ k}\Omega$ ) an 5V und NTC ( $10\text{ k}\Omega$  bei  $25^\circ\text{C}$ ) an Masse
- Spannungsabgriff an Eingang A5 des Arduino mit einer Auflösung des ADC von 10 Bit bei Spannungsbereich von  $0 \dots 5\text{V}$
- Anschluss des CAN-Boards (MCP2515) via SPI (SCK, MISO, MOSI, CS, GND, VCC, (INT))

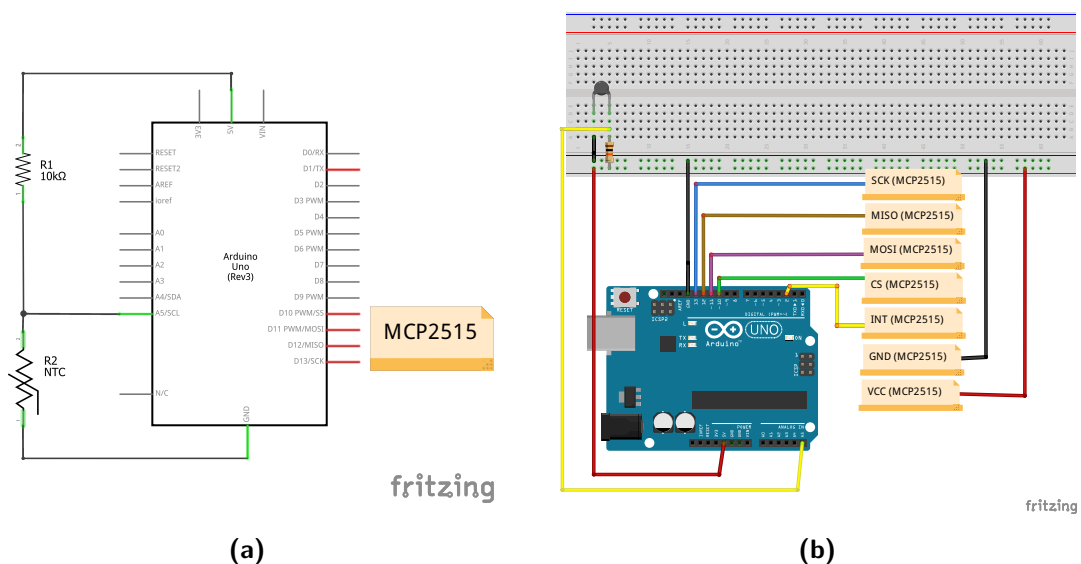


Abb. 1: Aufbau: (a) Schaltplan, (b) Steckplatine

## Softwareerstellung

Verwenden Sie als Basis das Arduino-Template **Can.ino**. Erstellen Sie einen **Ordner** mit dem Name **Can** und legen Sie in diesem Ordner die aus OPAL heruntergeladenen **vier Dateien** (**can.ino**, **can.h**, **mcp2515.h**, **mcp2515.cpp**) ab.

Für weitere Hilfe bei der Programmierung mittel Arduino-Bibliothek siehe <https://www.arduino.cc/reference/de/> sowie zum MCP2515-Treiber im Anhang dieser Anleitung.

**Initialisierung:** ( **void setup()** )

- Initialisierung der CAN-Nachricht (Länge, CAN-ID, ...)  
**Beachte:** Der CAN-ID lautet 100h + <Testplatznummer> – Somit sendet bspw. der Testplatz 4 seine Temperatur unter der CAN-ID **104h** und der Testplatz 10 mit der ID **10Ah**
- Initialisierung des CAN-Controllers mittels **mcp2515.reset()**
- Setzen der Bitrate auf 500 kBit/s und Taktfrequenz 8 MHz mittels **mcp2515.setBtrate(...)**
- Aktivieren des Normal-Modus mittels **mcp2515.setNormalMode()**
- Serielle Konsole für Debug-Zwecke aktivieren mittels **Serial.begin(9600)**

**Endlosschleife:** ( **void loop()** )

1. Spannungswert am Pin A5 mittels **analogRead()** einlesen und in Spannung umwandeln
2. Widerstand des Thermistors anhand Spannungsteilerregel bestimmen
3. Temperatur aus Widerstandswert bestimmen gemäß Formel

$$T(R) = \frac{1}{\frac{1}{B} \ln\left(\frac{R}{R_{25}}\right) + \frac{1}{T_{25}}}$$

wobei  $B$  – Sensorkonstante,  $R_{25}$  – Widerstand des Thermistors bei 25°C,  $T_{25}$  – Temperatur in Kelvin (25°C)

Ermitteln Sie die Sensorkonstante (B-Wert) aus dem Datenblatt des Thermistors (Siehe OPAL).

4. Ausgeben des Temperaturwert auf der seriellen Konsole und Kopieren in eine CAN-Botschaft  
**Beachte:** Auf dem AVR µC werden float-Variablen im Little-Endian-Format abgelegt – auch die erstellte CAN-Datenbasis erwartet die Temperatur in der CAN-Nachricht im Little-Endian-Format
5. Die Übertragung der Temperatur soll zyklisch alle 1000 ms erfolgen

## Anhang – Infos zum MCP2515-Treiber

CAN-Nachricht-Format: (siehe [can.h](#))

```
struct can_frame {
    canid_t can_id; /* 32 bit CAN_ID + EFF/RTR/ERR flags */
    __u8 can_dlc; /* frame payload length in byte */
    __u8 data[CAN_MAX_DLEN] __attribute__((aligned(8)));
};
```

Notwendige Methoden des MCP2515-Treibers: (siehe [mcp2515.h](#))

```
// Konstruktor
MCP2515(const uint8_t _CS,
        const uint32_t _SPI_CLOCK = DEFAULT_SPI_CLOCK);

// Reset
ERROR reset(void);

// Set to Normal Mode
ERROR setNormalMode();

// Set BittRate and Clock Frequency
ERROR setBittRate(const CAN_SPEED canSpeed, const CAN_CLOCK
                  canClock);

// Send a message
ERROR sendMessage(const struct can_frame *frame);
```

Mögliche Taktfrequenzen: (siehe [mcp2515.h](#))

```
enum CAN_CLOCK {
    MCP_20MHZ,
    MCP_16MHZ,
    MCP_8MHZ
};
```

Mögliche CAN-Bitraten: (siehe [mcp2515.h](#))

```
enum CAN_SPEED {
    CAN_5KBPS,
    CAN_10KBPS,
    CAN_20KBPS,
    ...
    CAN_100KBPS,
    CAN_125KBPS,
    CAN_200KBPS,
    CAN_250KBPS,
    CAN_500KBPS,
    CAN_1000KBPS
};
```

**Fehlercodes:** (siehe `mcp2515.h`)

```
class MCP2515
{
public:
    enum ERROR {
        ERROR_OK          = 0,
        ERROR_FAIL        = 1,
        ERROR_ALLTXBUSY    = 2,
        ERROR_FAILINIT     = 3,
        ERROR_FAILTX       = 4,
        ERROR_NOMSG        = 5
    };
    ...
};
```