ID S4

The CAN BUS and transmission of sensor signals in a CAN network

Required Knowledge

- Lab ID S3 (function and characterization of a temperature and humidity sensor with 1-wire interface and a gas sensor with I2C interface)
- Differential measurements with a two-channel oscilloscope
- CAN bus
- Physical Layer: CANHigh, CANLow, data rate
- CAN telegrams
- Arbitration

Material

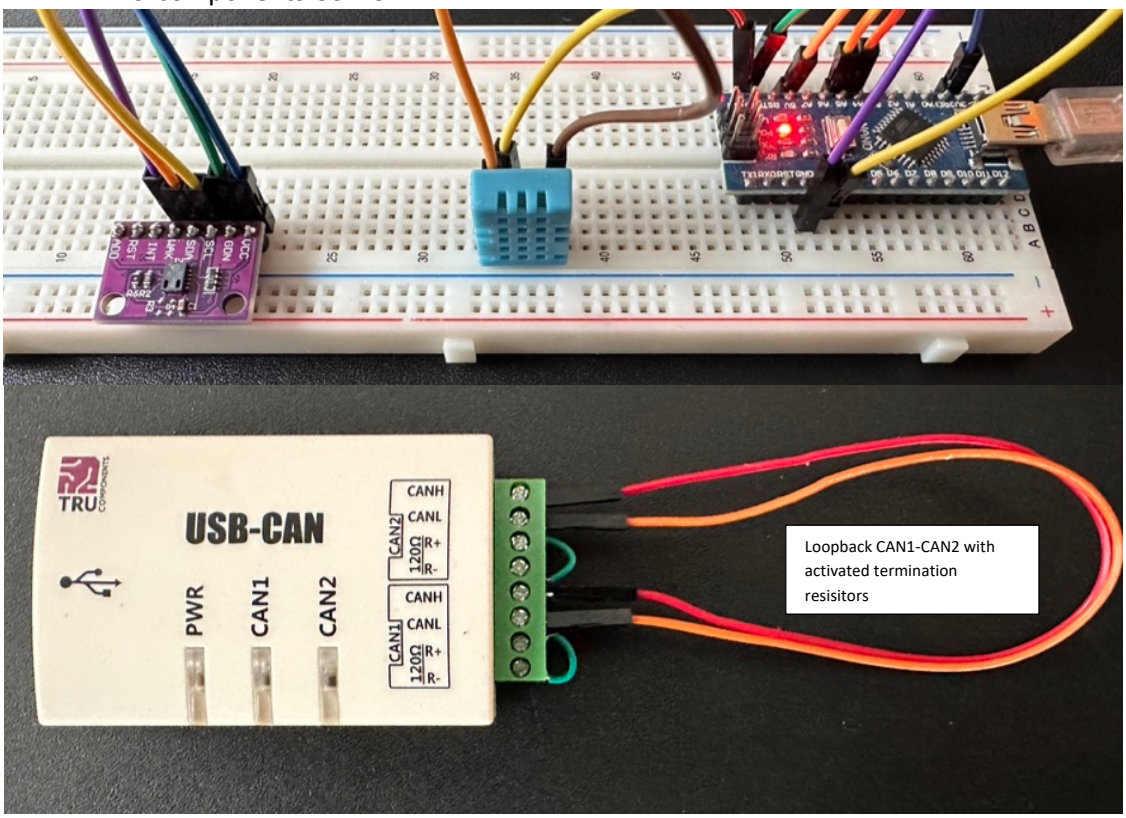
- Material from experiment ID S3 (Arduino, DHT11, CCS811)
- Program code from ID S3 (DHT11, CCS811)
- USB-CAN interface TRU-Components USB-CAN (https://asset.conrad.com/media10/add/160267/c1/-/en/002368701DS00/datenblatt-2368701-tru-components-tc-9474804-can- converter-usb-can-bus-sub-d9-not-galvanically-isolated-5-vdc-1-st.pdf)
- Male/male cables

Duration approx. 2h

Setup

Hardware:

- DHT11 und CCS811 from Lab ID S3
- TRU-components USB-CAN



Software:

USB-Driver:

- Windows: Install USB-Drivers with ZADIG (https://zadig.akeo.ie); map USB-CAN-Interface to libusb32; Driver will be automatically downloaded and installed

- OSX: Install Homebrew (OSX-Packet-Manager) with these commands:
    - /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
    - echo 'eval "$(/opt/homebrew/bin/brew shellenv)"' >> ~/.zprofile
    - brew install libusb

For emergencies (not recommended):

- https://asset.conrad.com/media10/add/160267/c1/-/gl/002368701DL00/download-2368701-tru-components-tc-9474804-can-umsetzer-usb-can-bus-sub-d9-nicht-galvanisch-getrennt-5-vdc-1-st.zip

Python:

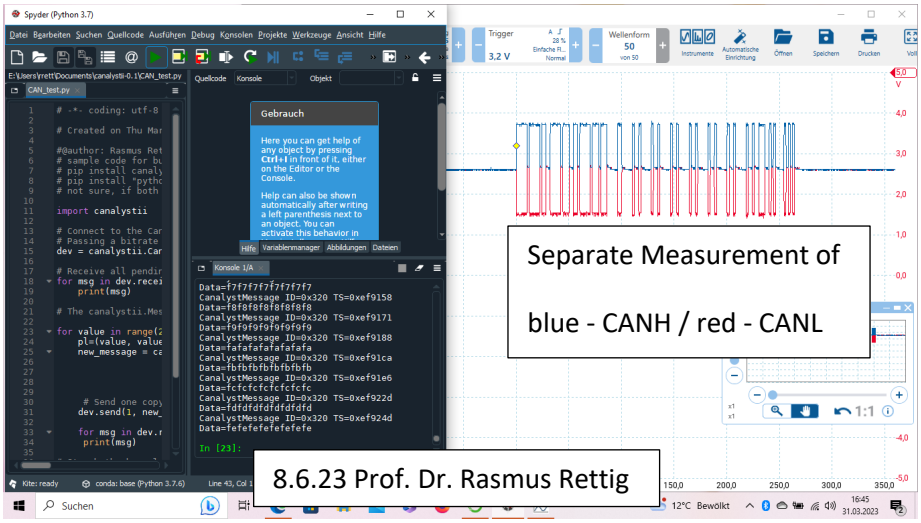
- Anaconda/Spyder with the library „catalystii“ („pip install canalystii“)
- Vorbereitete Programme von der Teams-Site (Python)
- Your programs from ID S3 (DHT11, CCS811)

Exercise 1 – CAN-Interface Setup und Test

- Install the USB CAN interface according to the instructions above
- Connect the two CAN interfaces (CANH-CANH and CANL-CANL) and activate the terminating resistors by wire bridges between R+ and R- for both interfaces
- Load the "CAN_test.py" program and analyze the function. Modify the program if required. Briefly describe the function. Draw a diagram of the data flow.
- Change the parameters and payload in the transmission (bitrate; can_id, remote, extended, data_len, data); document your investigations systematically and completely

Exercise 2 – CAN Physical Layer (Loopback CAN1-CAN2)

- Use two channels of the oscilloscope to carry out a differential measurement of the voltage between CANH and CANL
- Run the "CAN_test.py" program and follow the process on the oscilloscope.
- Attach a screen dump of a complete CAN telegram to your log and label the individual segments of the CAN telegram.
- Change the "bitrate" parameter from 500000 to 250000 and make a comparison measurement with the oscilloscope. Document your results with screen dumps and describe the differences.



Exercise 3 – Setup sensors and transmit data to the CAN-Bus (Loopback CAN1-CAN2)

- Setup DHT11 and CCS811 according to lab exercise ID S3.
- Modify the python program so that all 5 sensor values are first recorded and then transferred to the CAN bus (CAN1). To do this, first define your CAN telegram(s) with the assignment of the bits in the payload. You may have to distribute your payload across several telegrams.

- Check whether the sensor data corresponds to the data read back on the CAN bus (CAN2).
- Document your tests as examples.
- Draw the data flow diagram. How do you get the data onto the CAN bus, how do you read it back?

Exercise 4 – Automated error detection (Loopback CAN1-CAN2)

- Automate the process of exercise 3 so that deviations are automatically detected and reported (displayed on the screen).
- Run the test with at least 100 messages. Are messages lost? Do you observe transmission errors?
- Include the documented source code with your lab report.

Exercise 5 – Large(r) CAN-Network (CAN1 to common bus, please disable termination resistors BEFORE connecting to the common bus lines)

- Coordinate your activities with the other teams in the lab: Which bit rate do you want to use? Who uses which identifiers?
- Document the result in a table.
- Describe the coding of the sensor data in your CAN messages using your python code.
- Record the entire CAN network with all participants.
- Transfer your data to the bus and check whether at least one group receives it correctly. Carry out the test in both directions. Document the results.