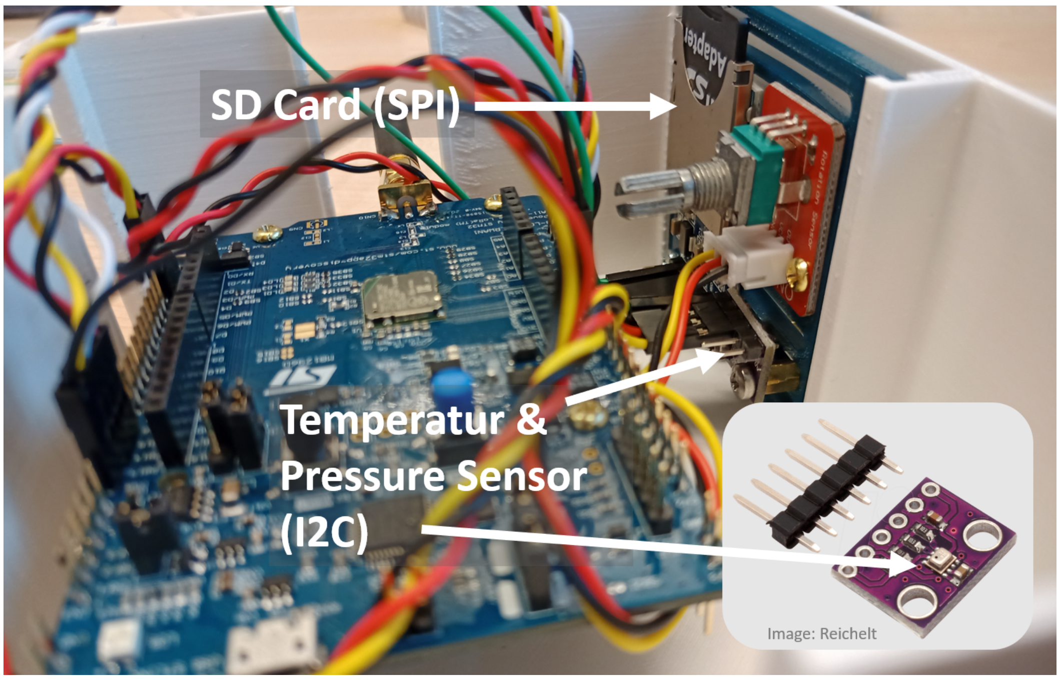
Lab 4: Onboard Communication

May 13th, 2024

Embedded systems typically consist of more than just a microcontroller. They include additional devices, such as sensors and actuators that require specific communication protocols for seamless control and interaction. In this lab, you'll explore the popular I^{2}C and SPI protocols and program a data logger that writes sensor readings to an SD card.



Task 4.1: I^{2}C Sensor

Your first task is to establish an I^{2}C communication with the Bosch BMP280 digital pressure sensor. Configure the sensor to read the pressure and temperature values using the I2C API, the datasheet and these additional hints! The program to be created should contain the following steps:

1. Wake up the sensor from sleep mode.
2. Get the calibration data.
3. Get the raw sensor data.
4. Calculate the physical values using the calibration data and the provided compensation functions.
5. Display the temperature in degrees Celsius (degC) and the pressure in hectopascal (hPa) in the console.

Task 4.2 : Data Logger

An SD card module is connected via the SPI interface to the microcontroller. In this task, the SDBlock-Device and FATFileSystem APIs are required to establish the SPI communication and to write a text file on the SD card. The program example-sd-writer provided in GitLab demonstrates how to format the SD card and create a test file. Run the example program and verify the successful creation of the test file by inserting and checking the SD card content with your PC. Afterwards write your own program for the following task:

- Write the pressure and temperature values obtained in the previous task with the elapsed time as comma-separated values (CSV) line-by-line to a file on the SD card. Including a header line, the log file should look like this:

```
Time (s),Pressure (hPa),Temperature (degC)
0,1013.1,22.1
2,1013.0,22.2
...
```

- Make sure that the measurements are logged at a constant interval, e.g., every 2 s.

Task 4.3 : Bonus (Submission until June, 7th 2024, 23:59)

In this bonus task, the previous program(s) shall be extended by implementing the following features:

- On system start up, the program should wait for a **user input via the console** to set the current date and time of the internal real-time clock (RTC) of the microcontroller. The **datetime format** should be `YYYY-MM-DD hh:mm:ss`, with year (YYYY), month (MM), day (DD), hour (hh), minute (mm), and second (ss). See also the example program provided in GitLab.
- A new log file should be created every time the microcontroller is reset. The filename should contain the file creation datetime, e.g. `2023-06-14_14-40-15.log`.
- In the main application, the BMP280 sensor data as well as the harvester measurements (see previous lab) should be polled periodically (e.g. every 2s), and **stored in a struct** with following fields:
  - `timestamp` string with format `hh:mm:ss` (hours:minutes:seconds),
  - `temperature` value with one decimal place in °C,
  - `pressure` integer value in hPa,
  - `solar_current` integer value in mA,
  - `cap_voltage` value with one decimal place in V,
  - `charge_level` integer value, the state of charge (SoC), in percent.
- You can write the data struct line-by-line in a CSV format similar to the previous task. However, **additional points** will be granted if you choose to use the **JSON format** instead:
  - JSON is a widely adopted, language-agnostic format for structured data that is commonly used in data science and IoT applications. To get started with JSON, you can also find an example in GitLab.

- All data written to the log file should be echoed by the console.
- Finally, when the **blue user button** is pressed, all log files should be deleted from the SD card and a new one should be created right away.

To learn more about the bonus grading, see also the **Bonus Guidelines** document in StudIP.