Sensor Nodes Laboratory – Final Report

Sebastian Thomas Thekkekara, Nitish Nagesh, Michael Haider, and Markus Becherer
TUM Department of Electrical and Computer Engineering, Technical University of Munich, Munich, Germany
sebastian.thekkekara@tum.de, nitish.nagesh@tum.de, michael.haider@tum.de, and markus.becherer@tum.de

***Abstract*—Temperature sensors play an important role in domestic and industrial applications. Measuring data in specific intervals is useful in analyzing the trends. The trigger to sample temperature data is given by a microcontroller which is a CC1350 Launchpad in this case. A second order butterworth low-pass filter using sallen-key topology was designed to send appropriate frequency samples to analog-to-digital converter (ADC). Inter-integrated-circuit (I2C) communication protocol was used to communicate between the launchpad and Programmable-System-on-Chip (PSoC) integrated development environment (IDE) . The temperature data was encapsulated, enqueued and sent wirelessly over a server to a receiver. The data is observed on a terminal. Fallback strategies and scaling-up method are discussed to enable widespread use of wireless sensor nodes.**

***Index Terms*—temperature data, PSoC, CC1350 Launchpad, I2C, wireless sensor node**

I. Introduction

The number of connected devices are increasing rapidly. The internet has brought a revolution ushering in renewed enthusiasm for staying connected while transcending geographical and temporal boundaries. Internet of Things (IoT) is a field which enables devices to be connected over the internet. IoT is the basis for applications such as smart homes, smart vehicles, smart meters, smart grids and so on. Each of these applications uses sensors to measure parameters such as temperature, humidity, pressure, emissions etc. which are activated based on various stimuli. These parameters are continuously monitored and in a feedback loop while checking for threshold values. Based on the operating conditions and algorithm used, specific actions can be taken to utilize sensor data. In the end, a database is available which can be used for future reference and can also be utilized for training neural networks enabling better data analysis and easier anomaly detection in case of wide deviations from reference values.

In this project, we intend to mimic an industrial workflow to develop a prototype from a given set of components and tools. The goal is to develop a fully functional prototype of a wireless sensor node by the end of summer semester 2019 i.e. September 2019. A resistance temperature detection (RTD) sensor Pt1000 is provided, along with PSoC 5LP (Programmable System-on Chip) and Texas Instruments CC1350 Launchpad. The aim is to measure the surrounding temperature every two seconds using the temperature sensor provided and display the output on a terminal. A fallback plan is also devised in case data measurements are insufficient or inappropriate. The project is divided into sub-tasks and illustrated accordingly for logical and streamlined workflow. They include:

- Developing a read-out front-end circuit
- Implementing a communication interface between the read-out circuit and wireless communication board
- Encapsulating the data from a server and visualizing it on a terminal

This would contribute to the already existing connected devices and in-turn facilitate smoother implementation in various applications.

II. Analog Front End

In the first few weeks, we spent time on understanding the basics of Analog-to-Digital Converters (ADCs), different types of filters and filter design principles. We also installed PSoC Creator 4.2 by Cypress Semiconductors and Code Composer Studio (CCS) by Texas Instruments (TI). We then utilized application notes related to temperature measurement using RTD in PSoC Creator. Different RTD configurations were available such as 2-wire, 3-wire and 4-wire configurations. 3-wire and 4-wire configurations allowed higher accuracy of resistor measurements by eliminating voltage drop across the lead connections and taking the voltage purely across the RTD for measurement [1]. We chose the 2-wire configuration for the sake of simplicity. To measure the resistance accurately, the current source and the ADC measuring the voltage must be accurate. There is a possibility to include a reference resistor to minimize the gain/offset error caused by the Analog-to-Digital Converter (ADC) and the Current Digital to Analog Converter (IDAC). The current passed through the RTD is taken as 1 mA. The value of current is chosen such that it does not cause self-heating across RTD while still ensuring that the maximum range of the ADC can be used.

A preliminary design involving IDAC, RTD and ADC was first developed. An Arduino was first used as the wireless board and the temperature data was sent to it using Universal Asynchronous Receiver Transmitter (UART) protocol. The output was viewed on a terminal. The process required multiple iterations and rewiring before the output was obtained. It was interesting to note that values of the datatype float were not displayed on the terminal and only values of type integer were displayed. In the next step, a filter was designed to supply filtered output to the ADC.

For this purpose, a low pass filter was used with quality factor Q = 1/2. The temperature data is read every 2 seconds which corresponds to 0.5 Hz. The trigger is given to the ADC from the TI launchpad. According to the Nyquist criteria, the filter should be designed such that its maximum cut-off frequency of the filter is the ADC sampling frequency/2 i.e. 0.25 Hz. To maintain anti-aliasing effect, we chose the cut-off frequency f_c as 0.2 Hz. A second order butterworth low pass filter was implemented using sallen-key topology as it is an active filter topology. Using [2], the two resistance values were chosen as 8.2 kΩ and the two capacitance values were chosen as 100 μF. The schematic of the analog front-end circuit is as shown in Fig. 1.

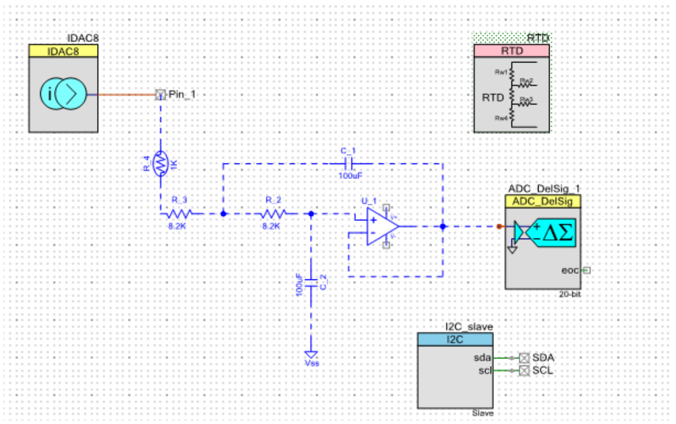

Fig. 1. Analog front-end schematic

III. Communication Interface

After designing the filter, we established the interface between the PSoC, off-chip components including RTD and filter, and the CC1350 Launchpad via I2C communication. This process as well had its share of challenges. We first tried implementing I2C communication using Arduino. The process was straightforward with using the Arduino as the master and a slave in the PSoC with SDA being used to transmit or receive serial data and SCL being the master generated clock. When we started implementing the same I2C protocol on the CC1350 Launchpad, we were not able to load the program even after building it due to missing libraries. After updating the drivers and rewiring the connections, the output was erratic. Finally, the launchpad had to be replaced with a new one before we could start tweaking the C code already provided.

The code was already provided to ensure compliance with the German laws such that data is transmitted only in the allowed and free band of 868 MHz. A deviation from this standard could result in interference with other communication and military frequency bands and is best avoided at all costs. As stated before, the temperature value is read every two seconds. The circuit is wired as shown in Fig. 2. A constant voltage of 5 V is applied across the IC741 operational amplifier (op-amp). The sallen-key filter designed is implemented which is then connected to the CC1350 Launchpad consisting the ADC. The voltage is measured across the Pt1000 by passing a current of 1 mA from the IDAC. It is observed that voltage is 1.113 V corresponding to a resistance of 1113 Ω. This corresponds to a temperature of 30°C [4].

The I2C master triggers the ADC and the I2C slave on the PSoC responds based on the master clock signals. I2C allows multiple masters and slaves to be connected without disturbing the communication pathway. This is possible because I2C uses a unique address transmitted before sending the data and also has a shared bus allowing many devices to be connected at a time. As a result, I2C uses lesser number of wires and allows data transmission on a single while enabling lower pin count [3]. The ADC in the PSoC can be operated in either continuous or single sample mode. In single sample mode, the data is read at the intervals triggered by the master. In continuous mode, the data is read uninterrupted and code has to be changed to calculate cumulative value of temperature. Average value of the data is considered to eliminate effects of noise and ensure zero mean. In this project, since the temperature is read every two seconds, even if the ADC is operated in continuous mode, the sampling becomes unnecessary and is in effect equivalent to simple sampling implemented.

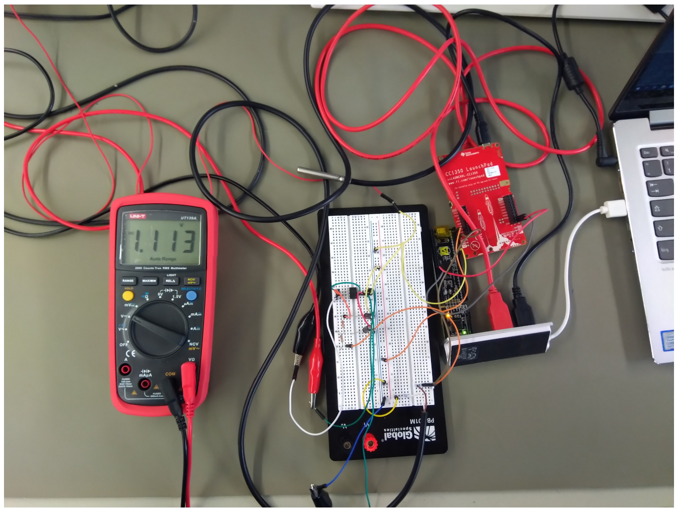

Fig. 2. Wired circuit with temperature sensor, filter design and communication interface

IV. Data Encapsulation and Enqueuing

Once the analog-front end was completed and the communication was setup with the CC1350 Launchpad, we started rewriting the code to obtain the temperature every two seconds. The I2C master was activated with a data rate of 400 kbps. The slave address in the PSoC was entered in the code so as to transmit data over the bus. The internal clock of the PSoC calculates the needed frequency of the I2C clock source and generates appropriate resources needed for its implementation. Pull-up resistors for I2C bus are not taken into consideration in this project. The data is enqueued on the bus and first the resistance was calculated. The corresponding temperature is calculated according to the equation [5]

$$R_T = R_0(1 + AT + BT^2) \tag{1}$$

where R_T is the resistance at T °C and R_0 is resistance at 0 °C. In our project R_0 is 1000 Ω.

Equation 1 is valid for temperatures above 0 °C. There exists a similar equation with additional parameters for temperatures below 0 °C [5] which is out of the scope of this project.

The resistance calculated in mΩ is at two byte data. This data is split into two components of one byte each and then transferred to the CC1350 Launchpad. In the launchpad, they are again combined to form a 16 bit value which is used to calculate the temperature. The corresponding values of resistance and temperature was compared with values in the look-up table [4]. It was found that there was only a slight deviation of both values when measurements were carried out at room temperature. One reason for the deviation can be attributed to interference from external noise. Another reason is the offset voltage between the inverting and non-inverting of the op-amp.

The communication with I2C is also not ideal as some data is lost in the interface of communication and on the bus. The launchpad was assigned with a specific IP address and the temperature data was transmitted over the server to the receiver. There exist concerns with respect to data security and privacy for data stored in the cloud.

V. Fallback strategies and Scaling-up

While implementing this project, we faced considerable challenges with respect to hardware and software tools. It is of utmost importance that the sensor data is available at regular intervals. in this project, if the temperature is not read every two seconds an alert can be generated requesting a trigger from the terminal manually. The program can also be modified such that the Launchpad and connected devices are sent into sleep mode after actuation by an undesirable input. When normal operation is to be run, the I2C communication, for instance can be woken up form sleep mode. The IDAC current can also be supplied intermittently based on request. This also enables power saving.

Power saving is especially crucial while using sensor nodes for practical applications such as smart lighting, smart meters, home automation etc. It helps increase longevity of the device under use thereby contributing to significant cost saving in corporations and institutions. The I2C communication protocol enables easy scaling-up of the prototype due to the availability of multi-master slave. Additional sensors such as humidity, ambient light, pressure etc. can be added to the existing prototype by using current multiplexer (MUX) and ADC MUX. The code has to be modified appropriately and each parameter can be observed simultaneously.

VI. Conclusion

Over the course of the semester, we learnt about different aspects of a wireless sensor node and successfully built a prototype to read temperature data every two seconds. We utilized the hardware and software components provided and overcame challenges while implementing each of the sub-tasks assigned. We are now able to evaluate a sensor node in an IoT framework on both device and circuit level while applying it to a specific application.

Acknowledgment

We thank Michael Haider for guiding us throughout the entire project. Michael helped us understand the basics required for completing the prototype along with providing the hardware components and software tools required to complete this project. He was available whenever requested and was instrumental in improving our understanding of developing a wireless sensor node. We sincerely thank professor Markus Becherer for giving us an opportunity to participate in this laboratory course and allowing access to labs at the chair which helped us work efficiently and effectively.

References

[1] http://www.cypress.com/AN70698

[2] http://sim.okawa-denshi.jp/en/OPstool.php

[3] https://maker.pro/arduino/tutorial/common-communication-peripherals-on-the-arduino-uart-i2c-and-spi

[4] Kongsberg Maritime AS, ”Platinum resistance temperature sensors Pt100 (Pt1000)” .

[5] http://www.cypress.com/AN70698