Real-time environmental data collection for Meteo Station Application

SVEATOSLAV VIZITIU
Electrical Engineering
Technical University
Str. G. Baritiu nr. 26-28, 400027 Cluj-Napoca
ROMANIA
sveatoslav.vizitiu@gmail.com

Abstract: - This paper introduces a data acquisition process for collecting environmental data using the following sensors: optical dust sensor GP2Y1010AU0F, a digital humidity sensor SHT15 and a miniature SPI digital barometer MPL115A1 connected to a FPGA board, each requiring an implementation of their own specific communication protocol. In the first part we will present the general purpose or data acquisition process. Also both the hardware and software implementation for a real-time environmental meteorological station will be shown. The physical interconnections between the different components, required for the construction of the whole system, will be introduced. Regarding the software implementation, all the communication protocols needed for communicating with the sensors, will be described in detail.

Key-Words: Leave acquisition, sensors, FPGA, LabVIEW, data communication, signal conditioning, meteo station, real-time

1 Introduction
Data acquisition is the process of sampling signals measuring physical entities in the environment and converting resulting samples in digital numerical values that can be manipulated by a computer. Data acquisition systems (abbreviated with acronyms DAS or DAQ) convert analog signals into digital values to be processed. Components of a data acquisition system are:
- sensors that convert physical parameters to electrical signals
- signal conditioning circuits to transform the sensor signals into a form that can then be converted into digital values
- analog to digital convertors - ADC

Data acquisition starts with the physical phenomenon or physical property to be measured (temperature, light intensity, gas pressure, liquid flow, force). Whatever type of physical property to be measured, the physical state must first be transformed into a unified form that can be sampled by the data acquisition system.

DAQ hardware usually interfaces signals with the PC. It can be a module that connects to the computer ports (parallel, serial, USB, etc.) or boards connected to slots on the motherboard (S-100 bus, AppleBus, MCA, ISA, PCI, PCI-E, etc.).

DAQ software is necessary for communication of the DAQ hardware with the PC. Device drivers perform read and write operations at the low-level registries, exposing a standard API for developing user applications.

A FPGA board (Field-Programmable Gate Array) is a semiconductor device containing programmable logic components and programmable interconnections. Logic blocks can be programmed to perform simple or complex functions. Most boards FPGA logic blocks also include memory elements, from flip flops to memory blocks[3].

In the last 10 years FPGA's have been seen as an alternative capable microprocessor, as long as the calculations do not involve floating point arithmetic. Recently, due to increased density and adding embedded arithmetic logic units made FPGAs effective systems with performance computing operations. Abundant hardware resources on the present reconfigurable systems allow new opportunities for high-performance parallel implementations.

The proposed system takes full advantage of FPGA capabilities and reads relevant data from the environment using specific sensors and then transmits it to the server so that it can be distributed to users.

2 Theoretical study
LabVIEW is a graphical programming environment used by millions of scientists and engineers to develop sophisticated measurement, testing and
control systems using intuitive graphical icons and wires that resembles a logic diagram[1].

LabVIEW and the LabVIEW FPGA Module deliver graphical development for field-programmable gate array (FPGA) chips on reconfigurable I/O (RIO) hardware targets[2].

With LabVIEW FPGA, it is possible to create custom measurements and to control the hardware without knowledge of low-level hardware description languages and board designs.

Inputs and outputs (I/O) on FPGA targets allow you to connect the FPGA target to other hardware in the system. FPGA I/O resources are fixed parts of the FPGA targets that you use to transfer data among the different parts of the system. Resources translate physical quantities to or from a digital value that you manipulate in FPGA Module software. An I/O resource has a terminal for receiving or generating a physical quantity. Many I/O resources on FPGA targets have physical terminals to which you can directly connect elements of the system. For example, digital signals, sensors, actuators, and so on, on some resources are associated with terminals built into the device, such as an on-board temperature sensor.

3 Hardware architecture

The main component of the system is an FPGA board which is used for its parallel computation capabilities, to communicate with the different sensors and send the relevant data back to the server.

A termination board must be used to better connect the pins located on the sensors to the I/O terminals on the FPGA board[3].

Other components that are essential and cause the system to be scalable are the various sensors depending on the environment data that needs to be collected: temperature, dust, humidity, etc.

For the implementation of the system presented in this article, we will use the following sensors: a compact optical dust sensor GP2Y1 a digital humidity sensor SHT15 and a miniature SPI digital barometer MPL11.

Since the SHT15 and MPL115A1 sensors only have 3.3V digital signals they can be connected only to the DIO terminals on the FPGA board. On the other hand, the GP2Y1010A0U0F sensor has both digital and analogue signals, so it needs to be connected to the DIO and AIO terminals located on the FPGA board. All the sensors will get their power directly from the board.
data terminal to be read. Data should be read in a very short interval (0.4 ms) because the signal LED must only be active for a period of 0.32 ms. The interval between readings should be at least 10 ms for sensor to not burn.

![Image](Fig 4. Sampling and execution times)

Analog data that has been read from the sensor data must be converted to represent the density of dust in the air. This conversion can be done using the function in Fig 5.

![Image](Fig 5. Conversion function)

### 3.2 Digital Humidity Sensor

SHT15 is part of Sensirion’s family of surface mountable relative humidity and temperature sensors[6]. The sensor integrates sensor elements plus signal processing on a tiny foot print and provide a fully calibrated digital output. A unique capacitive sensor element is used for measuring relative humidity while temperature is measured by a band-gap sensor. The applied CMOSens® technology guarantees excellent reliability and long term stability. The sensor is seamlessly coupled to a 14bit analog to digital converter and a serial interface circuit. This results in superior signal quality, a fast response time and insensitivity to external disturbances.

The supply voltage of SHT15 must be in the range of 2.4 – 5.5V and the recommended supply voltage (VDD) and Ground (GND) must be decoupled with a 100 nF capacitor – see Fig 6. The serial interface of SHT15 is optimized for sensor readout and effective power consumption. The sensor cannot be addressed by I2C protocol; however, the sensor can be connected to an I2C bus without interference with other devices connected to the bus. The controller must switch between the protocols.

![Image](Fig 6. Typical application circuit, including pull up resistor RP and decoupling of VDD and GND by a capacitor)

To communicate with the sensor, it must first be started up. In order to initiate a command the 'Transmission Start' sequence must be emitted. After this the controller must wait for the sensor to do the actual measurements and then can read the data corresponding to the relative humidity and temperature.

If communication with the sensor is lost, the controller can always send a 'Connection reset' to reset the serial interface.

![Image](Fig 7. Example of RH measurement sequence)

### 3.3 Digital Humidity Sensor

The MPL115A1 is an absolute pressure sensor with a digital SPI output targeting low cost applications[7]. Low current consumptions of 5 μA during Active mode and 1 μA during Shutdown (Sleep) mode are essential when focusing on low-power applications.

The MPL115A1 employs a MEMS pressure sensor with a conditioning IC to provide accurate pressure measurements from 50 to 115 kPa. An integrated ADC converts pressure and temperature sensor readings to digitized outputs via a SPI port.
Utilizing the raw sensor output and calibration data, the host microcontroller executes a compensation algorithm to render Compensated Absolute Pressure with ±1 kPa accuracy.

The MPL115A1 pressure sensor’s small form factor, low power capability, precision, and digital output optimize it for barometric measurement applications.

**Fig 8. Block Diagram and Pin Connections**

The MPL115A interfaces to a host (or system) microcontroller in the user’s application. All communications are via SPI. A typical usage sequence is as follows:

- **Initial Power-up** - All circuit elements are active. SPI port pins are high impedance and associated registers are cleared. The device then enters standby mode.
- **Reading Coefficient Data** - The user then typically accesses the part and reads the coefficient data. The main circuits within the slave device are disabled during read activity.

- **Data Conversion** - This is the first step that is performed each time a new pressure reading is required which is initiated by the host sending the CONVERT command. The main system circuits are activated in response to the command and after the conversion completes, the result is placed into the Pressure and Temperature ADC output registers.

- **Compensated Pressure Reading** - After the conversion has been given sufficient time to complete, the host microcontroller reads the result from the ADC output registers and calculates the Compensated Pressure, a barometric/atmospheric pressure value which is compensated for changes in temperature and pressure sensor linearity.

- **Shutdown** - For longer periods of inactivity the user may assert the SHDN input by driving this pin low to reduce system power consumption. This removes power from all internal circuits, including any registers.

4 **Software implementation**

4.1 **Digital Humidity Sensor**

Making a measurement requires the coordination between the LabVIEW FPGA and LabVIEW Real-Time modules in order to synchronize the signals sent to the sensor. The first operation is to initialize the I/O terminals from the sensor.

There are four control signals:

- 'Start reading' control - initiate communication with the sensor
- 'Stop reading' control - finish the communication with the sensor
- 'Data available' indicator - sensor data were processed and can now be extracted
- 'Done' indicator - communication with the sensor has ended

The 'Done' and 'Data available' indicators are first marked '0' to not influence the rest of the execution. When 'Start reading' becomes '1' then the LED signal is '1', after which we must wait for 0.28 ms for the sensor to do the measuring and to obtain the result.

**Fig 9. Sequence Flow Chart**

After this time interval, the 'Data available' indicator is marked '1' so that the sensor value to be read from a AIO terminal. The program then enters

**Fig 10. Communicating with GP2Y1 (part 1)**
a while loop which exits when one of the following have been satisfied:

- the 'Stop reading' control is '0' - means that the sensor data was read successfully
- more than 10 ms have passed and data wasn't read, so we terminate the process and send the 'Timedout' error

Regardless of the exit condition of the while loop, '0' is sent on the LED terminal to not exceed the execution time of the sensor and the 'Done' output is marked with '1' to signal the end of the current operation.

In a for loop, the program iterates 8 times, and in each iteration, after the rising edge of the SCK signal, the data is read from the DATA signal and then it marks SCK with '0' for a new transaction.

In case the current read operation must acknowledge the correct receipt of data, the DATA signal must be marked with '1', or with '0' otherwise.

At the end of the operation, another clock tick should be performed in order to transmit data confirmation (if any) and then the DATA signal must be assigned '1' in order to release the communication interface.

The operation of writing a byte begins by assigning '0' to the SCK signal, and then in a loop that is executed 8 times, the SDI signal receives the current bit of the transmitted command which is then followed by a rising edge and a downward edge of the signal SCK. All instructions are executed at an interval of 1 ms in order to ensure successful transfer of data.

An operation consists of sending a command (which is represented on 8 bits) to the sensor and then reading a byte from the sensor memory.
Reading the pressure involves performing two operations: sending commands 0x80 and 0x82 and reading the MSB and LSB of the raw pressure which will then be converted into an integer value.

The main process will begin its execution by initializing all signals that are used to communicate with sensor, sending the command to begin measuring and then reading the pressure, the temperature and the coefficients stored inside the sensor.

5 Conclusion
The purpose of this article is to develop an application that can communicate with various sensors and to easily obtain environmental data which will then be distributed to mobile applications.

The application is run on an FPGA board and uses digital and analog pins located on the board to communicate with temperature, pressure, humidity and dust density sensors. Once the files containing the implementation for each sensor have been compiled, they can be loaded into the FPGA board. Due to large .bit file size and the limited space on the FPGA board, communication with only one sensor can be achieved at any moment in time.

Communication with the sensors is made using sensor-specific protocols and interfaces. Communication involves the transmission of commands (made up of 8 bits), waiting for a time interval for the sensors to perform the physical measurements and then read the data stored in the sensor (must be performed two read operations, each reading 8 bits). Data stored in the sensor can be the recorded measurement or coefficients needed to calibrate the final result.

Sensor measurements are made at intervals of one minute for two reasons: it’s not required to read the values more often and to not overload the sensors.

After the data is collected from the sensors, it must be processed to obtain the correct measurements. Data will be transmitted to the server in XML format to be easily processed and passed on mobile applications.

Due to software and hardware modular architecture, the integration of a new sensor in the current design can be done very easily and require no significant changes.

References:
[8] Converting a Sensor Voltage Input to Physical Units, National Instruments Corp.