

# Open-air Temperature and humidity measurement and processing on FPGA board

Jizhong Lin([jlin13@syr.edu](mailto:jlin13@syr.edu)), Yunfei Liu([yliu137@syr.edu](mailto:yliu137@syr.edu))

## Abstract

The program that measures and processes the temperature and humidity data produced by sensor will burn to flash memory on FPGA board, which will be automatically loaded via SPI protocol when the power supply is provided. The interface of sensor and FPGA board is Pmod connector with 12 pins. Data is transmitted via I2C communication protocols from sensor to board. Switch on the board will convert different scales of temperature and measurements of humidity, the calculating process takes place in microprocessor. Then processed data is pushed to another serial terminal every second via UART communication rule, setting up the data rate and packet information. The board is connected to a smart device by USB A to Micro-B cable providing both power and data transmission channel. Finally, human read printing information originally from the sensor on the digital screen of smart device and learn real-time environment status.

## Introduction

To rapidly deliver a product based on embedded system, the engineer will consider implementing the hardware design on the FPGA board at the beginning of development and apply software design on the hardware to develop a specific application. Temperature and humidity measurement is one of environmental sensing interface, and different sensor or component should be added to this system to have greater measure of environment. The function of environment tracing system can be changed and redesigned on FPGA board with the programmable gates embedded on. ASIC design is available to explore as the application design is completed, which is lower-cost manufacturing product than the prototype designed from FPGA board.

## Background

The project has applied I2C, UART, SPI communication protocols to transmit the data between sensor, FPGA board and display device. It is developed on MicroBlaze softcore of Xilinx design, and Windows, Android operation system. The hardware implementation is Xilinx ArtyA7-100T Rev.E FPGA board, and software development tool is Vivado 2021.2 Xilinx design suite. The sensor HYGRO Digital Humidity and Temperature Sensor is adopted as primary external component for project, the core HDC1080 is developed by Texas Instrument.

### A. I2C

I2C, abbreviated from Inter-Integrated Circuit, is a synchronous, multi-controller/multi-target, serial communication bus. It is widely used in short-distance and intra-board applications that attaching low-speed peripheral sensor to microcontroller. There are different modes to determine the maximum speed from 100 kbit/s to 3.4Mbit/s, communicating in bidirectional with serial data line (SDA) and serial clock line (SCL) and 7-bit addressing.

I2C bus has two types of nodes, controller node and target node. Four modes of operation are applied to it, as controller transmit/receive, target transmit/receive. After every 8 bits of data in one direction, an acknowledge bit is transmitted in another direction.

Table 1 7-bit addressing

Field:	S	I <sup>2</sup> C address field							R/W'	A	I <sup>2</sup> C message sequences...	P
Type	Start	Byte 1								ACK	Byte X etc...	Stop
Bit position in byte X		7	6	5	4	3	2	1	0		Rest of the read or write message goes here	
7-bit address pos		7	6	5	4	3	2	1				
Note		MSB							LSB		1 = Read 0 = Write	

Table 2 Line state

Type	Inactive bus (N)	Start (S)	Idle (i)	Stop (P)	Clock stretching (CS)
Note	Free to claim arbitration	Bus claiming (controller)	Bus claimed (controller)	Bus freeing (controller)	Paused by target
SDA	Passive pullup	Falling edge (controller)	Held low (controller)	Rising edge (controller)	Don't care
SCL	Passive pullup	Passive pullup	Passive pullup	Passive pullup	Held low (target)

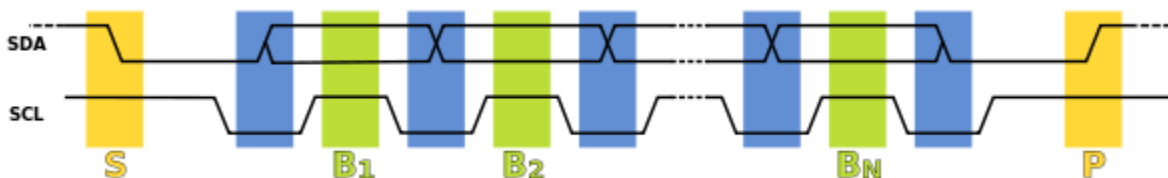


Figure 1 Timing diagram

## B. UART

UART, or universal asynchronous receiver-transmitter, is a hardware communication protocol that uses asynchronous serial communication with configurable speed. Asynchronous means there is no clock signal to synchronize the output bits from the transmitting device going to the receiving end.

When properly configured, UART can work with many different types of serial protocols that involve transmitting and receiving serial data. In serial communication, data is transferred bit by bit using a single line or wire. In two-way communication, we use two wires for successful serial data transfer. Depending on the application and system requirements, serial communications need less circuitry and wires, which reduces the cost of implementation.

Embedded systems, microcontrollers, and computers mostly use UART as a form of device-to-device hardware communication protocol. Among the available communication protocols, UART uses only two wires for its transmitting and receiving ends. Two UARTs directly communicate with each other as below.

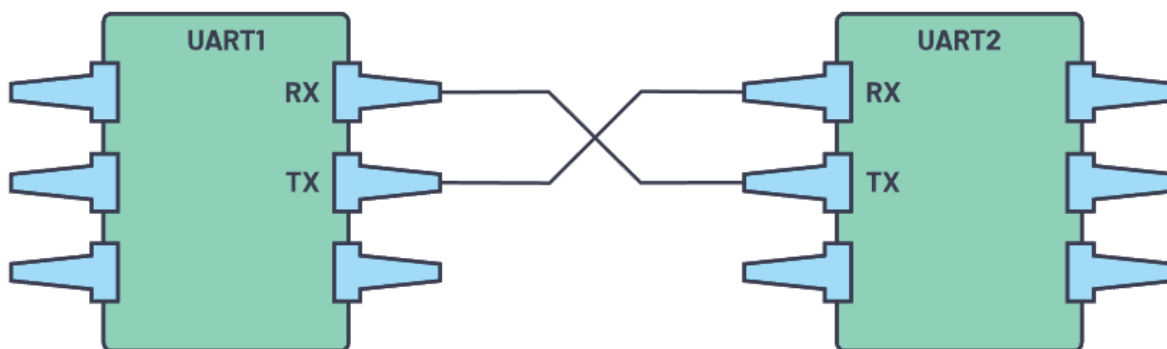


Figure 2 UART connection between two device

Table 3 Sample UART frame protocol

Header 1	Header 1	command	Data length	Data n ... Data n+1	Trailer 1	Trailer 2	Cyclic Redundancy Checking
----------	----------	---------	-------------	------------------------	-----------	-----------	----------------------------

The main purpose of a transmitter and receiver line for each device is to transmit and receive serial data intended for serial communication. Header is the unique identifier that determines if you are communicating with the correct device. Command will depend on the list of command designed to create the communication between two devices.

Data length will be based on the command chosen. You can maximize the length of data depending on the command chosen, so it can vary based on the selection. In that case, the data length can be adjusted. Data is the payload to be transferred from devices. Trailers are data that are added after the transmission is ended. Just like the Header, they can be uniquely identified.

The cycling redundancy checking formula is an added error detecting mode to detect accidental changes to raw data. The CRC value of the transmitting device must always be equal to the CRC computations on the receiver's end.

### C. SPI

The Serial Peripheral Interface (SPI) is a synchronous serial communication interface specification used for short-distance communication, primarily in embedded systems. The interface was developed by Motorola in the mid-1980s and has become a de facto standard. Typical applications include Secure Digital cards and liquid crystal displays.

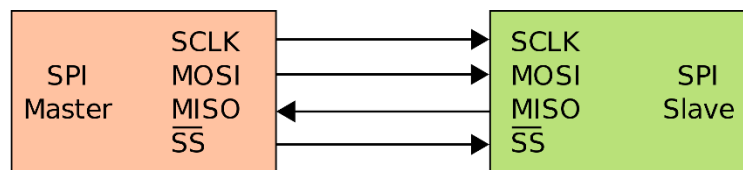


Figure 3 single master single slave communication

SPI devices communicate in full duplex mode using a master-slave architecture usually with a single master (though some Atmel devices support changing roles on the fly depending on an external (SS) pin). The master (controller) device originates the frame for reading and writing. Multiple slave-devices may be supported through selection with individual chip select (CS), sometimes called slave select (SS) lines.

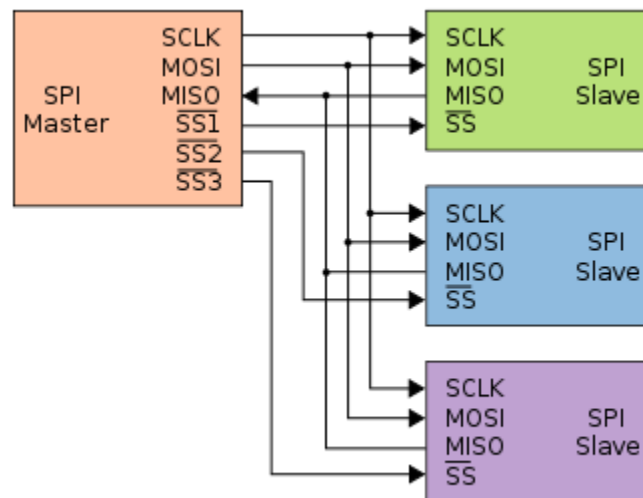


Figure 4 single master multi-slave communication

Sometimes SPI is called a four-wire serial bus, contrasting with three-, two-, and one-wire serial buses. The SPI may be accurately described as a synchronous serial interface, but it is different from the Synchronous Serial Interface (SSI) protocol, which is also a four-wire synchronous serial communication protocol. The SSI protocol employs differential signaling and provides only a single simplex communication channel. For any given transaction SPI is one master and multi slave communication.

## Technical Details

### A. UART configuration

It is Running on 100 MHz as the same as hygrometer, connected via Micro-USB TypeB connector. Baud rate 9600, 8 data bits--LSB first, 1 stop bits, no parity check. The state machine controls ASCII text transmission has 3 States: RDY, LOAD\_BIT, SEND\_BIT.

Table 4 UART module I/O information

Port	Width	Mode
SEND	1	in
DATA	8	in
CLK	1	in
READY	1	out
UART_TX	1	out

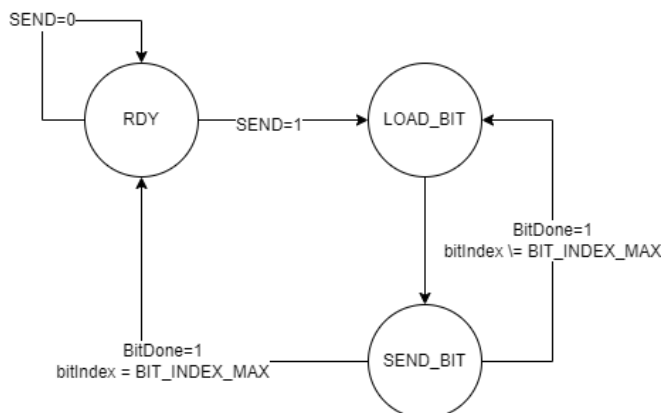


Figure 5 Finite State Diagram of UART

### B. SPI configuration

16 MB Serial quad-SPI flash memory is used to store the configuration file, the MCS file will configure FPGA board by power-on or reset event. Arty A7 supports x1, x2, and x4 bus widths and data rates of up to 50 MHz for programming. Memory parts: s25fl128sxxxxx0-spi-x1\_x2\_x4, which is manufactured by Spansion.

### C. I2C configuration

It is multi-controller/multi-target, serial communication bus, and widely used in short-distance and intra-board applications that attaching low-speed peripheral sensor to microcontroller. The maximum speed from 100 kbit/s to 3.4Mbit/s. Communicate in bidirectional with serial data line (SDA) and serial clock line (SCL) and 7-bit addressing. I2C bus has two types of nodes, controller node and target node

### D. Hygrometer interface

Sensor Temperature accuracy is in 0.2°C, humidity in accuracy  $\pm 2\%$ , and has 14-bit measurement resolution, works in 400kHz clock frequency at most. The data provided by sensor is 14-bit of raw data stored in the 16-bit register in most significant bits. It is converted to regular scaling system as formula provided below. The I/O information of hygrometer sensor driver is provided below.

Table 5 Formula of raw data transformation

Print out format	Transform raw data to human readable formula
TEM_HUM sensing now: 01100100001001 01011000111010 01100100001010 01010110010111 01100100001010 01010110100111	$Temperature(^{\circ}C) = \left( \frac{TEMPREG[15:0]}{216} \right) * 165^{\circ}C - 40^{\circ}C$ $Relative\ Humidity(\% RH) = \left( \frac{HUMREG[15:0]}{216} \right) * 100\%RH$

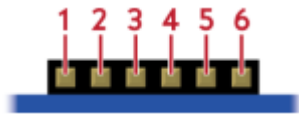


Figure 6 Pins of hygrometer diagram

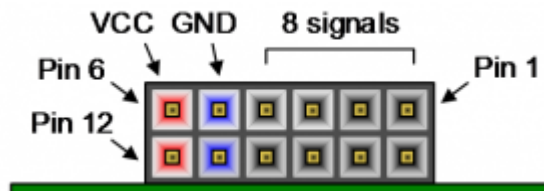


Figure 7 Pins of Pmod connector on FPGA board

Table 7 Port of hygrometer module

Port	Width	Mode
Clk	1	in
Reset_n	1	in
Scl	1	inout
Sda	1	inout
I2c_ack_err	1	out
Relative_humidity	N	out
temperature	M	out

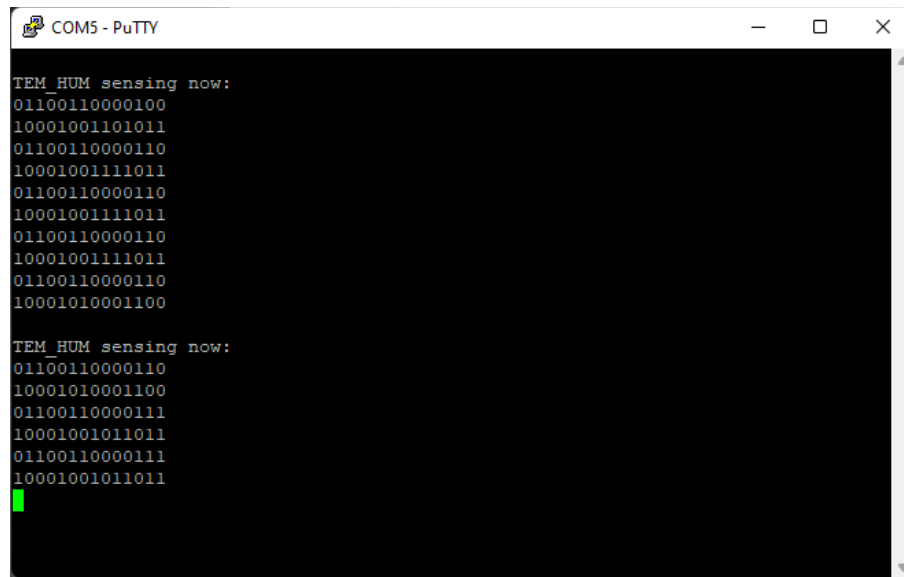
Table 6 Pins of hygrometer physical interface

Pin1	NC
Pin2	NC
Pin3	SCL
Pin4	SDA
Pin5	GND
Pin6	VCC

The reset\_n input port must have a logic high for the Hygrometer Pmod Controller component to operate. A low logic level on this port asynchronously resets the component. During reset, the component aborts the current transaction with the Pmod and clears the relative\_humidity and temperature data outputs and the i2c\_ack\_err output. Once released from reset, the Hygrometer Pmod Controller restarts its operation. It reconfigures the Pmod and resumes collecting and outputting relative humidity and temperature data.

## Progress record

### A. Read raw data



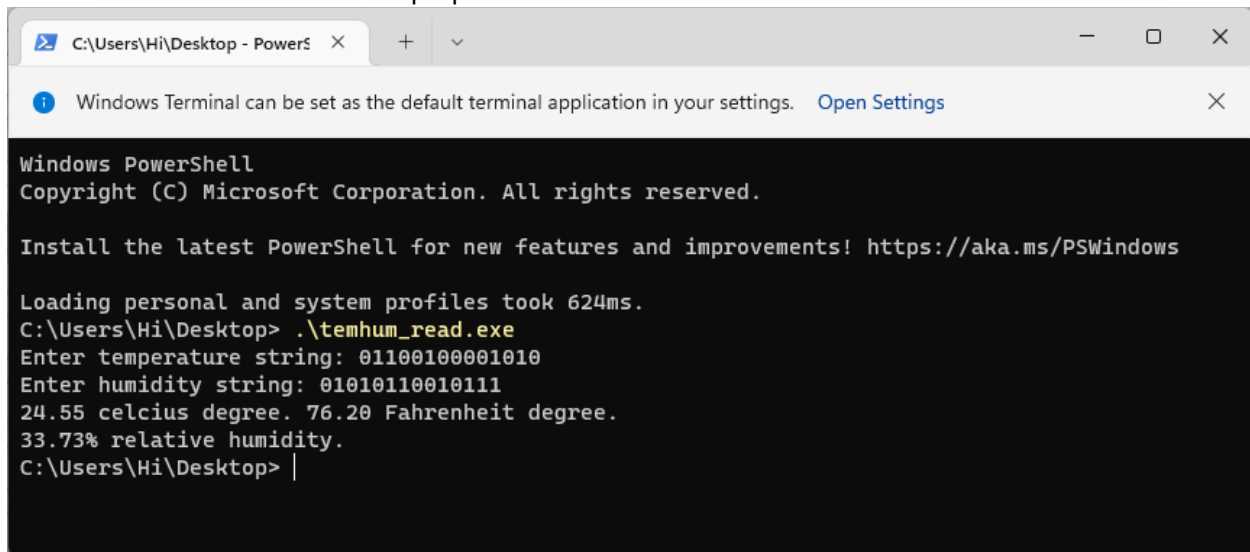
```
COM5 - PuTTY

TEM HUM sensing now:
01100110000100
10001001101011
01100110000110
10001001111011
01100110000110
10001001111011
01100110000110
10001001111011
01100110000110
10001010001100

TEM HUM sensing now:
01100110000110
10001010001100
01100110000111
10001001011011
01100110000111
10001001011011
```

Figure 8 Screenshot of raw data print on putty in Windows platform

It is measuring the temperature and humidity provided by sensor on development platform Windows. The communication protocol between FPGA board and laptop is UART. The connection port on the Windows platform can be found on the device manager, Universal Serial Bus controllers. In this case, the FPGA board is connected to the laptop on COM5.



```
C:\Users\Hi\Desktop - PowerShell

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

Loading personal and system profiles took 624ms.
C:\Users\Hi\Desktop> .\temhum_read.exe
Enter temperature string: 01100100001010
Enter humidity string: 01010110010111
24.55 celcius degree. 76.20 Fahrenheit degree.
33.73% relative humidity.
C:\Users\Hi\Desktop> |
```

Figure 9 Read the temperature and humidity through an executable file on Windows OS

The 14-bit data of temperature and humidity printed on serial terminals are pasted to a running executable file on the powershell in Windows OS. The powershell output is the human readable temperature in two scaling system, and humidity in relative units. The executable file only receive input exactly fit in the 14-bit of Boolean value in two lines in the order indicated by the powershell. The project is designed to display the temperature in Celsius and Fahrenheit scaling on the serial terminal, the change of scaling system is controlled by a switch on FPGA board.

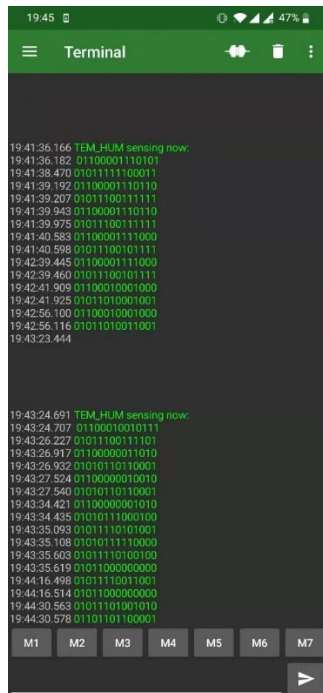


Figure 10 Screenshot of raw data print on Serial terminal on Android platform

It is the sensor measurement implementation on an embedded device. The power provided to the FPGA board and sensor is the battery of Android smart phone we daily used. The mobile app Serial USB Terminal developed by Kai Morich, connect the board and cellphone by UART protocol. The data printing on the picture above is split into two parts, one of which is measured on the indoor environment, another is measured in the refrigerator.

## B. Convert raw data into human readable number

The hardware design is failed to synthesis due to the variable length of signal in function. Variable strlen in function readtem is assigned by the length of integer and fraction part of arithmetic result, which is non-synthesizable design for hardware implementation. Similar design and problem is occurred on another function readhum, although the source code passed the syntax check and simulation, it is not synthesizable for the FPGA board.

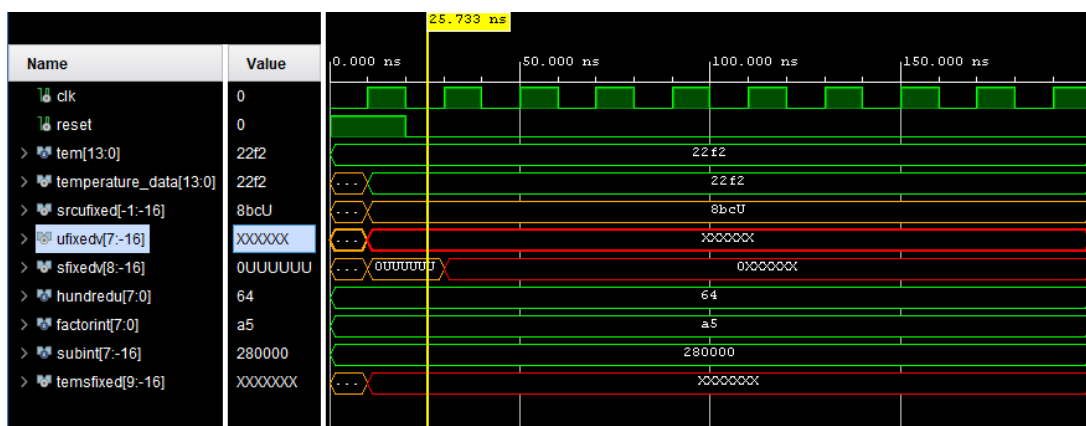
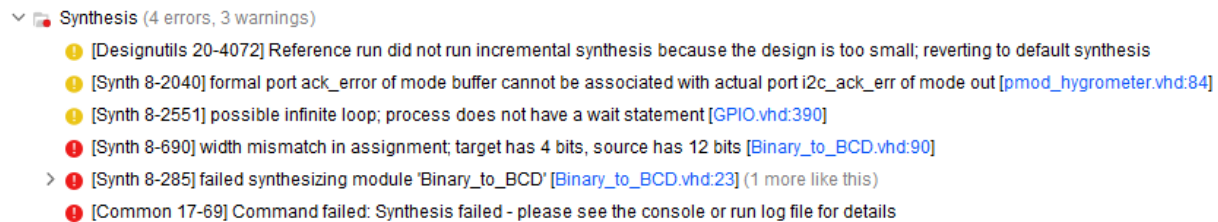


Figure 11 Waveform of unfixed signal calculation



Testbench is written in another project to scope out the calculation process of ufixed signal provided by IEEE FIXED\_PKG. Constant 14-bit signal tem is first passed to the process on testbench. Other constants as multiplication factor and subtraction terms are part of the elements in arithmetic calculations. In the first clock rising edge, it is caught by temperature\_data signal, which is the register stores the digital data of sensor on original project; Then it is converted into ufixed signal srcufixed from -1 to -16 bit as a fraction number, the least significant bits are padding with zeros; Intermediate ufixed signal ufixedv is assigned after multiplication operation on srcufixed signal, it is changed from U to X but failed to represent the expected valid value; On the second clock rising edge, sfixed signal sfixedv is changed from U to X following ufixedv signal, adding a positive sign bit on the most significant bit; The final result is signal temsfixed that is generated after multiplication and subtraction, it is changed from U to X on first clock rising edge.



*Figure 12 Screenshot of error message of implementing binary\_to\_BCD module on Vivado*

The Binary\_to\_BCD module is referenced from the VHDL code provided on the internet. According to the definition on binary\_to\_bcd module, the target and source should have 4 bits of data, however the source bit is 12, it passes the entire decimal digit to the process rather a single digit. The error message indicates that the module might have logical faults on the design. To find out if this design is fault-free, a separate testbench for binary\_to\_bcd module is required before working on the main project any further.

## Conclusion

The project has achieved reading sensor data and display it on an external device with screen via several communication protocols. The hardware design is successfully implemented on the FPGA board within relatively short development cycle. Rapid function verification of hardware design is already done in current stage of development. Hardware routing and new implementation could be included for better applications on certain scenarios depending on the library technology provided.

## Future work

Data conversion from raw data measured by the sensor to human readable format will be implemented on FPGA board rather than another smart device that provide display as well. The design will add temperature and humidity calibration function independently. Light ambient module can also be included as part of the environmental sensing elements, and the data is process on the FPGA board by allocating amount of clock cycles to it. Other sensor can be included if the connector on FPGA board is available, and the clock cycles is sufficient to assign tasks to different modules. The project will include more software design, rather than put the most effort on hardware design.

## Resources and references

- UART communication protocol  
<https://www.analog.com/en/analog-dialogue/articles/uart-a-hardware-communication-protocol.html>
- SPI communication protocol  
[https://en.wikipedia.org/wiki/Serial\\_Peripheral\\_Interface](https://en.wikipedia.org/wiki/Serial_Peripheral_Interface)
- I2C communication protocol  
<https://en.wikipedia.org/wiki/I%C2%B2C>
- ArtyA7 reference manual posted by Digilent  
<https://digilent.com/reference/programmable-logic/arty-a7/start?redirect=1>
- Datasheet of Pmod HYGRO Digital Humidity and Temperature Sensor  
<https://digilent.com/shop/pmod-hygro-digital-humidity-and-temperature-sensor/>
- Pmod interface specification  
[https://digilent.com/reference/\\_media/reference/pmod/digilent-pmod-interface-specification.pdf](https://digilent.com/reference/_media/reference/pmod/digilent-pmod-interface-specification.pdf)
- TI HDC1080 Low Power, High Accuracy Digital Humidity Sensor with Temperature Sensor  
<https://www.ti.com/lit/ds/symlink/hdc1080.pdf>
- GPIO driver in VHDL  
<https://github.com/Digilent/Arty-A7-100-GPIO/tree/master/src/hdl>
- UART\_TX communication protocol in VHDL  
[https://github.com/Digilent/Arty-A7-100-GPIO/blob/master/src/hdl/UART\\_TX\\_CTRL.vhd](https://github.com/Digilent/Arty-A7-100-GPIO/blob/master/src/hdl/UART_TX_CTRL.vhd)
- Pmod HYGRO sensor driver in VHDL  
<https://forum.digikey.com/t/humidity-and-temperature-sensor-pmod-controller-vhdl/13064>
- Binary to BCD module in VHDL, includes integer part only  
<https://www.nandland.com/vhdl/modules/double-dabble.html>
- STL\_LOGIC\_VECTOR to ASCII in VHDL  
<http://computer-programming-forum.com/42-vhdl/838e970e310e791c.htm>
- String to ASCII in VHDL  
<https://stackoverflow.com/questions/22900938/vhdl-is-there-a-convenient-way-to-assign-ascii-values-to-std-logic-vector>
- The project repository on GitHub  
<https://github.com/Alex-Lin5/TemHum-hybrid-sensor-on-FPGA-board>