Design and Analysis of ZigBee (802.15.4) Based Data Loggers for Wireless Applications

Dr. Ana Martins*[1], João Silva[2] & Ricardo Almeida[3]

[1](Assistant Professor, Department of Mechanical Engineering), University of Porto, Porto, Portugal
[2](PhD Scholar, Department of Electrical Engineering), University of Porto, Porto, Portugal
[3](Associate Professor, Department of Materials Science), University of Porto, Porto, Portugal

ABSTRACT

This dissertation is to produce data logger for remote system. It consists of a temperature sensor for constantly monitoring temperature and Zigbee module for wireless data transfer at short distances. The measured temperature parameter will be sent to microcontroller system. And with the help of Zigbee module temperature is sent to computer from remote area on digital displays. In recent time is sent with the help of inbuilt RTC which ARM7 (LPC2148) microcontroller system has got. There will also provision for setting of RTC time if user required in any case and it is done with the help of remote computer. When microcontroller circuit power up then user is ask to set RTC time after this data transfer take place continuously which appears on computer via Zigbee module connected to computer.

KEYWORDS: Zigbee, microcontroller, monitoring Temp.

I. INTRODUCTION

The original model for this system consists of standalone data logger. As the requests from users are getting higher, the application of wireless communication as medium transmission rather than the use of wires. Other than that, extra features or some kind of bonus will be add to this system which is capable to access to wireless communication for alarming some events. Moreover, these systems which apply low power consumption are easy to manage and install. It is based on wireless sensor networks.

Wireless Sensor Network consists of large numbers of sensor nodes. The nodes are equipped with sensor devices that are used for a certain applications. For example, the sensor device is camera and it is used to retrieve the environment data visually, microphone is used to detect the sound, thermometer and thermocouple are used to detect the changes in temperature. Every sensor nodes are also equipped with wireless module in order to communicate with each other. The communication between the nodes are performed by establishing the routing topology in the system before the data can be transmit from the certain sensor node to the collection point or host

Wireless sensor network (WSN) is an emerging technology and has great potential to be employed in critical situations. Wireless sensor networks have been deployed in various monitoring applications such as industrial, health, environmental, and security The Wireless Sensor Networks comprise of relatively inexpensive sensor nodes capable of collecting, processing, storing and transferring information from one node to another. These nodes are able to autonomously form a network through which sensor readings can be propagated [20]. Therefore, a standard is required that is capable of establishing the network between these nodes as well as provide low cost and less power consumption. Fortunately, there is a standard called ZigBee that is capable of accomplishing all these requirements.

ZigBee standard is developed by the ZigBee Alliance that defines the communication protocols for low-data-rate and short-range wireless networking. ZigBee based wireless devices operate at 868 MHz, 915 MHz, and 2.4 GHz frequency bands. ZigBee is developed on the top of IEEE 802.15.4 standard . is designed for low-power consumption and allows batteries to last up to years using primary cells without any chargers (low cost and easy installation). ZigBee has a wide application area such as home networking, industrial networking, system monitoring and many more having different profiles specified for each field.

II. MATERIALS AND METHODS

ADC working

Most real world data is in analog form. Whether it be temperature, pressure, voltage, etc, their variation is always analog in nature..so we have to convert this analog data into digital format so that computer or microcontroller can understand it and process on it. Sensor gives analog data in form of variation in current and voltage, ADC read this variation and process a digital data according to analog input and send to microcontroller to process it further.

Terms used in ADC

Resolution – The resolution of the converter indicates the number of discrete values it can produce over the range of analog values. A computer is a digital machine that stores a number in binary. If you are storing a digital 2-bit number you can store 4 different values: 00, 01, 10, or 11. Now, you can say ADC have 2-bit resolution and you have a device which converts an analog voltage between 0 and 10 volts into a 2-bit digital value for storage in a computer.

This device will give digital values as follows

Voltage	2-Bit Digital Representation
0 to 2.5	00
2.5 to 5	01
5 to 7.5	10
7.5 to 10	11

Table 1.0

Note:

- Higher the resolution smaller the step size
- Smaller the step size better accuracy

Step size – small amount of change in analog input that can understand
for example 8-bit ADC,
step size=Vref/2^8-1 = Vref/255
Vref – used to detect step size
Conversion –
Dout=Vinput/(step size)
Dout – decimal output digital data
Vinput – analog input voltage
For example – for 8 bit ADC, Vref = 2.56 V, calculate digital output for 1.7 V input.
step size=(2.56 V)/256=10 mV
Dout=(1.7 V)/(10 mV)=170=10101010

Method to inbuilt ADC

In this tutorial we will go through LPC2148 adc programming. Analog to Digital Conversion(i.e. ADC) , as the name suggests , is all about converting a given analog signal into its digital form or say a digital value. So, what does this mean? Well, basically its measuring the voltage of a given analog signal. The analog signal can be differential, single-ended unipolar, etc. The converted digital value represents the measured voltage. This conversion or measurement happens in presence of a fixed and accurate reference voltage. The analog signal is compared to this reference voltage and then estimations are made to get the final measured value.

ADC on LPC214x

ADC on LPC214x is based on Successive Approximation (SAR) conversion technique.

Pins relating to ADC Module of LPC214x :

Pin	Description
AD0.1 to AD0.4 (P0.28/29/30/25) and AD0.6,AD0.7 (P0.4/5)	Analog input pins. **Note from Datasheet:** "If ADC is used, signal levels on analog input pins must not be above the level of Vdda at any time. Otherwise, A/D converter readings will be invalid. If the A/D converter is not used in an application then the pins associated with A/D inputs can be used as 5V tolerant digital IO pins."
Vref	This is the reference voltage pin. It must be connected to an accurate reference voltage source.
Vdda,Vssa	Vdda is Analog Power pin and Vssa is Ground pin used to power the ADC module.

Table 2.0

Registers used for ADC programming in LPC214x

(For AD1 registers replace 0 with 1 wherever applicable)

1) AD0CR – A/D Control Register : This is the main control register for AD0

1. Bits[7 to 0] – SEL : This group of bits are used to select the pins(Channels) which will be used for sampling and conversion. Bit 'x'(in this group) is used to select pin A0.x in case of AD0.
2. Bits[15 to 8] – CLKDIV : These bits stores the value for CLKDIV which is used to generate the ADC clock. Peripheral clock i.e. PCLK is divided by CLKDIV+1 to get the ADC clock
3. Bit 16 – BURST : Set this to 1 for doing repeated conversions. Set this bit to 0 for software controlled conversions , which take 11 clocks to finish.
4. Bits[19 to 17] – CLKS : These bits are used to select the number of clocks used for conversion in burst mode along with number of bits of accuracy of the result in RESULT bits of ADDR.

Value	clocks / bits
000	11 clocks / 10 bits
001	10 clock / 9 bits
010	9 clock / 8 bits
011	8 clock / 7 bits
100	7 clock / 6 bits
101	6 clock / 5 bits
110	5 clock / 4 bits
111	4 clock / 3 bits

Table 3.0

5. Bit 21 – PDN : Set it to 1 for powering up the ADC and making it operational. Set it to 0 for bringing it in power down mode.

6. Bits[26 to 24] – START : These bits are used to control the start of ADC conversion when BURST (bit 16) is set to 0. Below is the table as given in datasheet :

Value	Significance
000	No start (this value is to be used when clearing PDN to 0)
001	Start the conversion
010	Start conversion when the edge selected by bit 27 occurs on P0.16/EINT0/MAT0.2/CAP0.2 pin
011	Similar to above – for MAT0.0 pin
100	Similar to above – for MAT0.1 pin
101	Similar to above – for MAT0.3 pin
110	Similar to above – for MAT1.0 pin
111	Similar to above – for MAT1.1 pin

Table 4.0

7. **Bit 27 – EDGE :** Set this bit to 1 to start the conversion on falling edge of the selected CAP/MAT signal and set this bit to 0 to start the conversion on rising edge of the selected signal. (Note: This bit is of used only in the case when the START contains a value between 010 to 111 as shown above.)
8. Other bits are reserved.

AD0GDR – A/D Global Data Register : This is the global data register for the corresponding ADC module. It contains the ADC's DONE bit and the result of the most recent A/D conversion.

1. Setting up and configuring ADC Module for software controlled mode :

First we will define some values which will help us setup the AD0CR register to configure the AD0 block before we can use it.

```
#define CLKDIV (15-1) // 4Mhz ADC clock (ADC_CLOCK=PCLK/CLKDIV) where "CLKDIV-1" is actually used , in our case PCLK=60mhz
#define BURST_MODE_OFF (0<<16) // 1 for on and 0 for off
#define PowerUP (1<<21) //setting it to 0 will power it down
#define START_NOW ((0<<26)|(0<<25)|(1<<24)) //001 for starting the conversion immediately
#define ADC_DONE (1<<31)
```

Here we define CLKDIV which is divided by PCLK to get the ADC clock <=4Mhz. In our case we will be using a PCLK of 60Mhz hence we divide 60Mhz by 15 to get 4Mhz. But note that the ADC module actually needs a value of (CLKDIV-1). This is because it adds "+1" to the value internally (in case if user uses a CLKDIV of 0 it will be still valid). For our purposes CLKDIV is a 'zero-indexed' value hence we must subtract it by 1 before using it. In our case we need to supply a value of 14 i.e. (15-1) to AD0CR.

BURST_MODE_OFF(bit 16) , PowerUP(bit 21) and ADC_DONE(bit 31) are defined as required. CLKS_10bit has been defined for 10 bit resolution - you can change the bit combination as per your needs. Finally START_NOW is defined as "001" which is for starting the conversion 'now'.

Next we define AD0CR_setup which contains basic configuration for setting up the ADC Module. We feed CLKDIV , BURST_MODE_OFF and PowerUP into AD0CR_setup as follows

```
unsigned long AD0CR_setup = (CLKDIV<<8) | BURST_MODE_OFF | PowerUP;
```

Now we assign AD0CR_setup to AD0CR along with channel selection information to select channels as required. Finally we assign(by ORing) START_NOW to AD0CR to start the conversion process as shown :

AD0CR = AD0CR_setup | SEL_AD06;

AD0CR |= START_NOW;

Note that AD0CR can be assigned/setup in a single step. But I am doing it in three steps to keep things simpler.

2. Setting up and configuring ADC Module for Burst mode :

Configuring ADC Module is similar to what was done in software controlled mode except here we use the CLKS bits and don't use the START bits in AD0CR. ADC_DONE is also not applicable since we are using an ISR which gets triggered when a conversion completes on any of the enabled channels.

#define CLKDIV (15-1) // 4Mhz ADC clock (ADC_CLOCK=PCLK/CLKDIV) where

"CLKDIV-1" is actually used , in our case PCLK=60mhz
#define BURST_MODE_ON (1<<16) // 1 for on and 0 for off
#define CLKS_10bit ((0<<19)|(0<<18)|(0<<17)) //10 bit resolution
#define PowerUP (1<<21) //setting it to 0 will power it down

3. Fetching the conversion result in software controlled mode :

In software controlled mode we continuously monitor bit 31 in the corresponding channel data register ADDR. If bit 31 changes to 1 from 0 , it means that current conversion has been completed and the result is ready. For example , if we have used channel 6 of AD0 then we monitor for changes in bit 31 as follows :

while((AD0DR6 & ADC_DONE) == 0); //this loop will terminate when bit 31 of AD0DR6

4. Fetching the conversion result in Burst mode :

In Burst mode we use an ISR which triggers at the completion of a conversion in any one of the channel. Now , we just need to find the Channel for which the conversion was done. For this we fetch the channel number from AD0GDR which also stores the conversion result. Bits 24 to 26 in AD0GDR contain the channel number. Hence , we shift it 24 places and use a 3bit mask value of 0xF as shown below :

unsigned long AD0GDR_Read = AD0GDR;

int channel = (AD0GDR_Read>>24) & 0xF; //Extract Channel Number

After knowing the Channel number, we have 2 options to fetch the conversion result from. Either we can fetch it from AD0GDR or from AD0DRx of the corresponding channel. In the examples covered in 'examples section' of this tutorial I have used AD0GDR for extracting the conversion result as follows :

int currentResult = (AD0GDR_Read>>6) & 0x3FF; //Extract Conversion Result

Introduction of inbuilt UART

Here is a quick recap of UART basics : Uart uses TxD(Transmit) Pin for sending Data and RxD(Receive) Pin to get data. UART sends & receives data in form of chunks or packets. These chunks or packets are also referred to as 'transmission characters'. The structure of a UART data packet is as shown below :

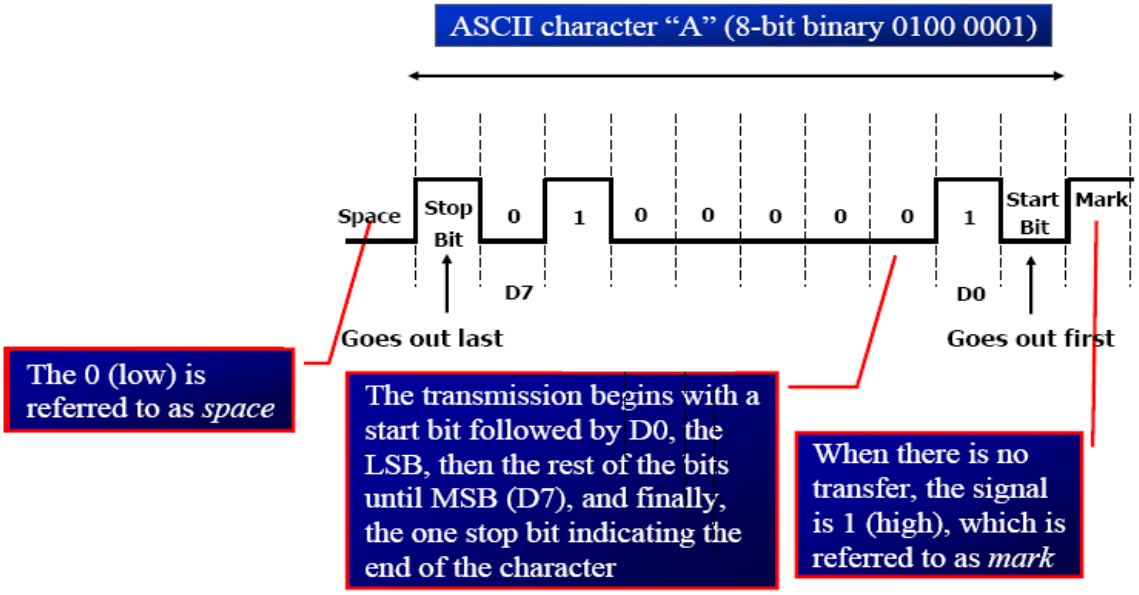

Figure 1.0

Now , Lets start with the main Tutorial. LPC214x has 2 UART blocks which are UART0 and UART1. For UART0 the TxD pin is P0.0 and RxD pin is P0.1 and similarly for UART 1 the TxD pin is P0.8 and RxD pin is P0.9 as shown in the table below :

Pins:	TxD	RxD
UART0	P0.0	P0.1
UART1	P0.8	P0.9

Table 5.0

Registers used for UART programming in LPC214x:

Before we can use these pins to transfer data , first we need to configure and initialize the UART block in our LPC214x microcontroller. But before doing that, lets go through some of the important registers: (for UART1 registers replace 0 with 1)

Data Related Registers :

1) U0RBR – Receiver Buffer Register (READ ONLY!): This register contains the top most byte(8-bit data chunk) in the Rx FIFO i.e the oldest received data in FIFO. To properly read data from U0RBR , the DLAB(Divisor Latch Access) bit in U0LCR register must be first set to 0.

2) U0THR – Transmit Holding Register (WRITE ONLY!): U0THR contains the top most byte in Tx FIFO and in this case its the newest(latest) transmitted data. As in the case with U0RBR , we must set DLAB=0 to access U0THR for write operation.

Baud Rate Setup related registers :

1) U0DLL and U0DLM **–** Divisor Latch registers: Both of them hold 8-bit values. These register together form a 16-bit divisor value which is used in baud rate generation which we will see in later section. U0DLM holds the upper 8-bits and U0DLL holds the lower 8-bits and the formation is **"**[U0DLM:U0DLL]**".** Since these form a divisor value and division by zero is invalid, the starting value for U0DLL is 0×01 (and not 0×00) i.e the starting value in combined formation is **"**[0x00:0x01]**"** i.e 0×0001**.** Please keep this in mind while doing baud-rate calculations. In order to access and use these registers properly, DLAB bit in U0LCR must be first set to 1**.**

2) U0FDR – Fractional Divider Register : This register is used to set the prescale value for baud rate generation. The input clock is the peripheral clock and output is the desired clock defined by this register. This register actually holds to different 4-bit values (a divisor and a multiplier) for prescaling which are:

1. Bit [3 to 0] – DIVADDVAL : This is the prescale divisor value. If this value if 0 then fractional baud rate generator wont have any effect on Uart Baud rate.
2. Bit [7 to 4] – MULVAL : This is prescale multiplier value. Even if fractional baud rate generator is not used the value in this register must be more than or equal to 1 else UART0 will not operate properly.
3. Other Bits reserved.

Remark from Usermanual : *"*If the fractional divider is active (DIVADDVAL > 0) and DLM = 0, the value of the DLL register must be 2 or greater!*"*

UART Baud Rate Generation:

In most cases the actual baudrate will drift a little above or below the desired baud and also, as the desired baudrate increases this drift or error will also increase – this is because of the equation itself and the limitations on MULVAL , DIVADDVAL! For e.g. if the desired baud rate is 9600 and you get a baud like 9590 , 9610 , 9685 , 9615 , etc.. then in almost all cases it will work as required. In short , a small amount of error in actual baudrate is generally tolerable in most systems.

The master formula for calculating baud rate is given as :

$$\text{BaudRate} = \frac{\text{PCLK in Hertz}}{16 \times (256 \times \text{DLM} + \text{DLL}) \times (1 + \text{DIVADDVAL}/\text{MULVAL})}$$

which can be further simplified to :

$$\text{BaudRate} = \frac{\text{PCLK in Hertz}}{16 \times (256 \times \text{DLM} + \text{DLL})} \times \frac{\text{MULVAL}}{\text{MULVAL} + \text{DIVADDVAL}}$$

with following conditions strictly applied :

0 < MULVAL <= 15
0 <= DIVADDVAL <= 15
If DIVADDVAL > 0 & DLM = 0 then, DLL >= 2

As it can been seen this formula has 2 prominent parts which are : A Base value and a Fractional Multiplier i.e: BaudRate = [Base] x [Fraction(i.e. Prescale)]. This Fractional Multiplier can be used to scale down or keep the base value as it is .. hence its very useful for fine-tuning and getting the baudrate as accurate as possible. Where PCLK is the Peripheral Clock value in Hz , U0DLM and U0DLL are the divisor registers which we saw earlier and finally DIVADDVAL and MULVAL are part of the Fractional baudrate generator register.

III. RESULTS AND CONCLUSION

This thesis is developed for remote monitoring of system with respect of time. Therefore user does not need go to remote area to know the temperature of device .The device developed can work efficiently up to a 30m distance depending upon surrounding environmental and 100m for open air, which can be used as a modern technique as per requirement .

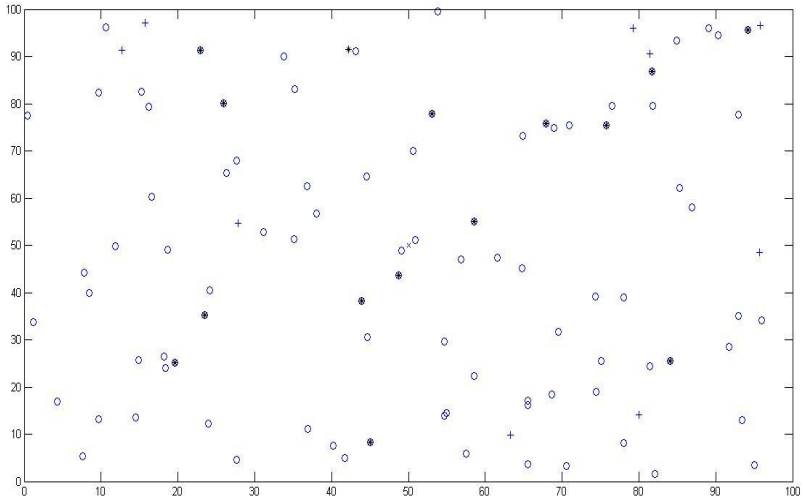

Figure 6.0

Limitations and difficulties

- Received data in computer cannot be stored for future analyses.
- Difficulties encountered while interfacing with the Xbee module and programming of ARM7 .

Future scope of work

- To control remote system
- If system is more than one then it can form WLAN for providing information regarding data like temperature

Some types of research for Xbee are used are:

- Cosmic ray astrophysics
- Gamma ray and X -ray astrophysics
- Optical and ultra - violet astrophysics
- Infrared/sub millimeter
- Atmospheric science

IV. REFERENCES

1. Mahesh G and Mrs. Y. Syamala, "Weather Data Logger Using ARM Processor", International Journal of Advanced Research in Computer Science and Software Engineering,Vol 3, pp. 2277 128X, JNTU, INDIA, 2013
2. Md. Moyeed Abrar and Rajendra R. Patil, "Multipoint Temperature Data Logger and Display on PC through Zigbee using PSoC", International Journal of Advanced Research in Computer and Communication Engineering, *Vol. 2, pp.* 2278-1021, AIET, Gulbarga, India, 2013
3. H.Vijaya Laxmi and M.Narender, "Communication b/w Mobile-Robots'and PC controller Based On ZigBee Network", International Journal of Engineering Research and Applications,Vol. 1, pp. 2248-9622, SREC Warangal, AP, India
4. Arun Kumar, "A Zigbee Based Wireless Datalogging System" International Journal of Scientific & Engineering Research Vol. 3, pp. 2229-5518, India, 2012
5. P.V. Mane-Deshmukh, B.P. Ladgaonkar, S.C. Pathan and S. S. Shaikh, " Microcontroller Pic 18f4550 Based Wireless Sensor Node to Monitor Industrial Environmental Parameters", International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 3, pp. 2277 128X, *Akluj, India, 2013*
6. Nashtara Islam, Robert Watson and Philip Moore,"Z8 Encore Microcontroller Based Data Logging System",AUJT, Loughborough University Leicestershire, United Kingdom, 2006

7. Howitt I and Gutierrez J.A, "IEEE 802.15.4 Low Rate –Wireless Personal Area Network Coexistence Issues", IEEE Conference of Wireless Communications and Networking, Vol. 3, pp. 1481-1486, New Orleans, LA, USA, 2003.
8. Li J and Liu Q, "Application and Research of ZigBee Tehnology in the Miner's Lamp Monitoring", IEEE International Conference on Future Information Technology and Management Engineering, Vol. 1, pp. 317-320, Changzhou, 2010.
9. López M, Gómez J.M, Sabater J and Herms A, "IEEE 802.15.4 based Wireless monitoring of pH and temperature in a fish farm", 15th IEEE Mediterranean Electrochemical Conference, pp. 575-580, Valletta, 2010.
10. Pengfei L, Jiakun L and Junfeng J, "Wireless temperature monitoring system Based on the ZigBee technology", 2nd IEEE International Conference on Computer Engineering and Technology, Vol. 1, pp. 160-163, Chengdu, 2010
11. Singh R, Mishra S and Joshi P, "Pressure Monitoring in Wireless Sensor Network Using Zigbee Transceiver Module", 2nd IEEE International Conference on Computer & Communication Technology, pp. 225-229, Allahabad, 2011.