Universität Bremen

*EXCELLENT.
An Excellence Initiative
Success Story

Universität Bremen
Fachbereich Physik, Elektro- und Informationstechnik
Institut für Mikrosensoren, -Aktoren Und -Systeme (IMSAS)

Master Thesis

Sub-Gigahertz Wireless Sensors For Monitoring Of Food Transportation

Submitted by

Faisal Ali Khan

1st Supervisor: Prof. Dr.-Ing. Walter Lang

2nd Supervisor: Dr.-Ing. Reiner Jedermann

Date : April 15, 2015

Statement

I hereby certify that I have my Master Thesis be justified by me for part of a group project - have made without assistance, and that I have no other than the stated sources and aids.

All sites that are taken literally or in spirit from publications, I have been identified as such..

The Thesis Report may not be changed after submission.

Bremen, April 15, 2015 _____
(Name)

Abstract

This thesis presents a work on *"Sub-Gigahertz Wireless Sensors for Monitoring of Food Transportation"*. Previous research studies reveal that high water content inside banana pallet restricts the communication between wireless sensor nodes operating at 2.4 Ghz frequency below 0.5 meter by attenuating the radio signal. This issue hinders the remote monitoring of temperature inside refrigerated container for food transportation.

New approach was required to solve this issue of signal attenuation. The versatile low power Dash7 protocol, operates at 433 Mhz, becomes the solution to reduce attenuation and achieve long range communication by integrating it in wireless sensor nodes. To measure temperature and humidity inside the refrigerated container, node is interfaced to the Sht25 sensor and Tmp100 sensor by writing software driver for the I2C protocol and the EEPROM is also interfaced for the permanent storage of these parameters by writing the same driver to node. The sensor nodes measure the real measurement values from the interfaced sensors and send it to the Gateway in real time after every minute by using Dash7 module. The communication range of 20 meter tested inside the building was attained successfully without any loss of packet. Experiments also carried out in a noisy open field area detected by the Rf analyser, yielded the communication range of 60 meters without any packet loss. Maximum communication range tested at 100 meter caused only one packet to receive at the gateway end with -82 dB of node RSSI value. After optimizing the timing of communication operation between sensor node and gateway, nodes draw only 200 μA current in sleep mode majority of the time and 17.6 mA in the active mode only for a very short span of time which makes the battery life of nodes quite long.

The magnificent improvement in the communication range between node and gateway by using Dash7 protocol makes it completely beneficial to use it in Intelligent Container with its high performance of reading range and low battery consumption.

Acknowledgement

First of all I am very thankful to Almighty Allah, whose blessings helped me to carry out the Master Thesis successfully.
I would also like to express my deeply gratitude to Prof.Dr.-Ing.Walter Lang, the head of the IMSAS department, for granting me the opportunity to do my thesis under his supervision.
Special thanks to my second supervisor Dr.-Ing.Reiner Jedermann for giving me the guidance, support, his sincere dedication to supervise the thesis with the outcome and his availability during the whole period of my thesis.
Last but not least, I am very thankful to my parents, my whole family and friends for giving me the love, courage, prayers and complete backing during my thesis.

Faisal Ali Khan

Contents

1 Introduction

The scope of this chapter is to get familiar with the background of the thesis, thesis motivation and its objectives. Additionally, structure of the whole report is described.

1.1 Thesis Background

Intelligent Container, a prototype, is used for the transporting of foods and fruits such as bananas. Foods such as bananas and meat are loaded on the containers in ships from the Costa Rica are to be delivered in Europe. It is the utmost requirement to transport foods from one place to another in the quality condition without the food ripening. There are several factors of food ripening inside the container such as varying temperature, relative humidity, atmospheric conditions such as O2 and CO2, the emission of the Ethylene gas by the fruits such as bananas also causes the ripening of other fruits. This gas is ejected when they ripen. The green life of bananas estimation, the time span until the bananas start to ripen, allows to take subsequent action such as First Expired First Out: the technique in which the food with low life cycle is delivered first [1]. Studies have been carried out already to estimate the green life of bananas, such as keeping the temperature to 13°C lasts the green life to 31.9 days in the storage or warehouse provided that the temperature must be 14°C in the 14 days of the sea transportation of bananas [2]. The humidity plays also an important role in storing bananas, high relative humidity of $90-95$ percent is proposed [3]. Atmospheric conditions of 5% CO2 and 2% O2 concentration also become a factor to delay the ripening. Temperature below 13°C causes the chilling injuries, so it is avoided[2]. Finally, it can be concluded that the temperature inside the container is of significant importance along with other parameters in order to prevent the ripening.
It was required to develop a system which can monitor all these conditions remotely. Wireless sensors, microcontroller with interfaced sensors, is the perfect choice to monitor humidity and temperature inside the refrigerated container.

1.2 Motivation

Reduction of signal attenuation and distance for the radio waves propagation of the wireless sensors operating at 2.4 Ghz inside a banana pallet is considered as a major problem. Due to increased water content in banana, the radio propagation of the radio chips inside sensor

nodes can fall below 0.5 m. The sensors placed inside a banana pallet is a TelosB hardware which uses frequency of 2.4 Ghz with IEEE 802.15.4 standard. The sensors send data to the gateway over multiple hop due to signal attenuation[4]. The communication diagram of such a monitoring is shown in figure 1.1.

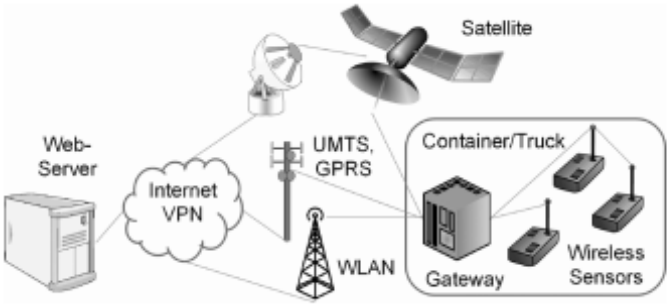


Figure 1.1: Remote Monitoring [5]

The gateway data can be sent to the the wireless lan, satellite or gprs and can be accessed on a web server placed at a remote location. Testing of the signal attenuation of these sensors was performed in [6]. It was found out that -72.5 db signal strength was achieved when sensors placed in the middle of the banana pallet at a distance of 0.25 m from each other and reduced further to -83.6 dB at distance of 0.5 m with only 52 percent of packets received. Due to these limitations, sensor nodes with the high communication range were badly needed.

1.2.1 Proposed Solution

Theoretical studies have shown that sensor nodes operating below 1 GHz frequencies are much less influenced by the fruits containing water contents than the sensors currently operating at 2.4 GHz range. A higher communication range and higher reliability of the radio link is expected from wireless modules according to the dash7 standard, operating at 433 Mhz [6]. DASH7 is an open source wireless sensor networking standard, which operates in the 433 MHz with the ISO 18000-7. It is power efficient radio protocol, transmits maximum 200 kbps and penetrates in concrete and liquids[7]. Dash7 also gives the feasibility of the communication between the Gateway and the sensor node with multi-hop. Due to these considerations of the Dash7, OSS-7 open source implementation of the Dash7 protocol has been implemented in the supported hardware such as WizziMote in the project. The WizziMote is a board which has the CC430F5137 controller chip with the radio frequency core chip like CC1101 in one package.

1.3 Thesis Overview

An overview of the thesis can also be seen in figure 1.2. The gateway communicates with each sensor node on 433 Mhz frequency to get the required data. The sensor node, interfaced with sensors and EEPROM by I2C interface, responds by sending the sensor data to the gateway and writing it in the EEPROM. The sensor values which is received on gateway is sent to a PC through serial communication (UART) of gateway. On the PC, the Java software creates a log file and saves the data obtained from the gateway.

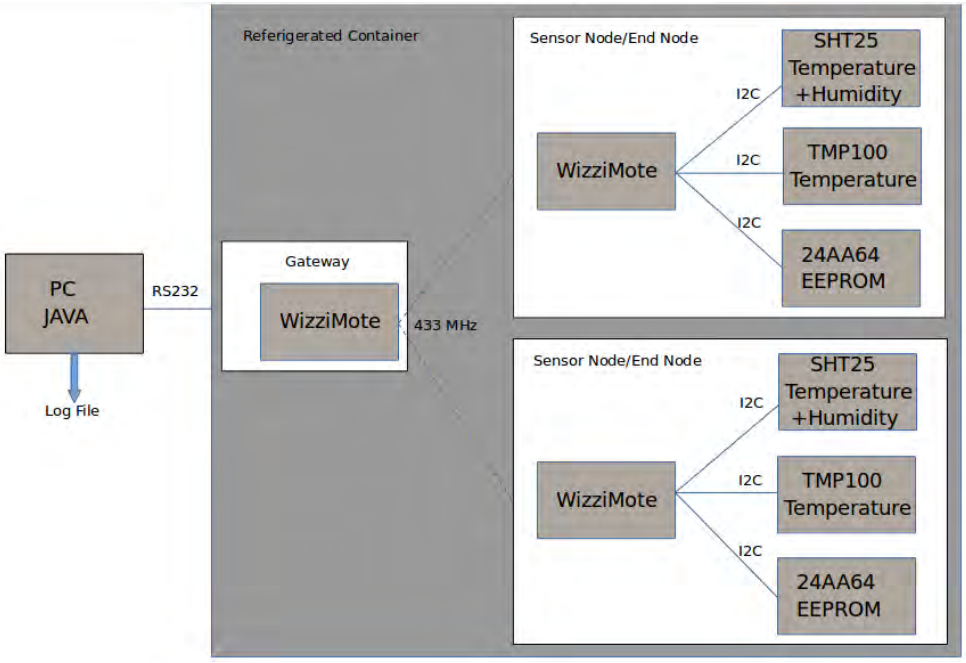


Figure 1.2: Overview of the thesis

1.4 Tasks of the thesis

Keeping in mind of the problems stated above, different tasks are defined for the thesis.

- Study of complete implementation of wireless sensor nodes.
- Connect combined temperature and humidity sensor such as Sht25 to the wizzimote board for measuring temperature and humidity inside the banana pallet and write the I2C software driver.
- Connect low cost Temperature sensor such as Tmp100 to the wizzimote board for measuring temperature of the sensor node and write the I2C software driver.

- Connect EEPROM memory such as 24AA64 to the wizzimote and write the I2C software driver. EEPROM is used for storing the sensors data permanently, which is needed in case of weak radio link.
- Integrate the sensor queries for direct communication between the sensor node and a gateway. The sensor node should transmit data to the gateway in real time as well as store the data in the EEPROM.
- Implement energy efficient programming by making use of low power modes in the microcontroller and use hardware watchdog for any system failure.
- Modify already existing Java software to receive sensor data on a PC from the Gateway and write them in a log file.
- Demonstration with four or more nodes either in banana container at the project partner Dole or in a building.
- Performance study analysis of the received signal strength with the reading range.
- Measure current consumption in sensor nodes in active and sleep mode.
- Documentation of complete hardware and software modifications.

1.5 Structure of the Report

After introduction, the thesis work is organized as following:

- Chapter-2: Gives detailed description of the used peripherals of micrcontroller and their configurations settings.
- Chapter-3: Describes the hardware components used in the project such as Tmp100, Sht25, 24AA64 EEPROM etc and their implementation of I2C interface.
- Chapter-4: Describes the DASH7 alliance protocol.
- Chapter-5: Description of the communication between the Sensor node and Gateway and the software written for it.
- Chapter-6: Describes the testing performed.
- Chapter-7: Conclusion.

2 CC430F5137 and Its Peripherals

In this chapter, all the peripherals and the functionalities of the microcontroller which have been implemented in the project are described briefly here. CC430F5137 microcontroller has been used in this project, this controller is mounted on the WizziMote board. The sensors such as Sht25 and Tmp100 and the EEPROM memory are interfaced with this microcontroller.

2.1 Wizzi Mote Board

The Wizzi Mote consists of Texas Instrument CC430F5137 Micocontroller and wireless antenna matching circuit with the range of 433Mhz. This Kit is the most suitable for Dash7 development applications because of its size, price and its capability to switch easily from one application to another.
CC430 family provides integration of the microncontroller core with the RF core, therefore making it a true System-on-chip solutions and highly efficient in wireless applications. One can use this board to develop its wireless applications aiming at 433Mhz and this becomes the solid reason to use such a microcontroller in the project [8]. The Wizzi Mote Board is shown in figure 2.1.

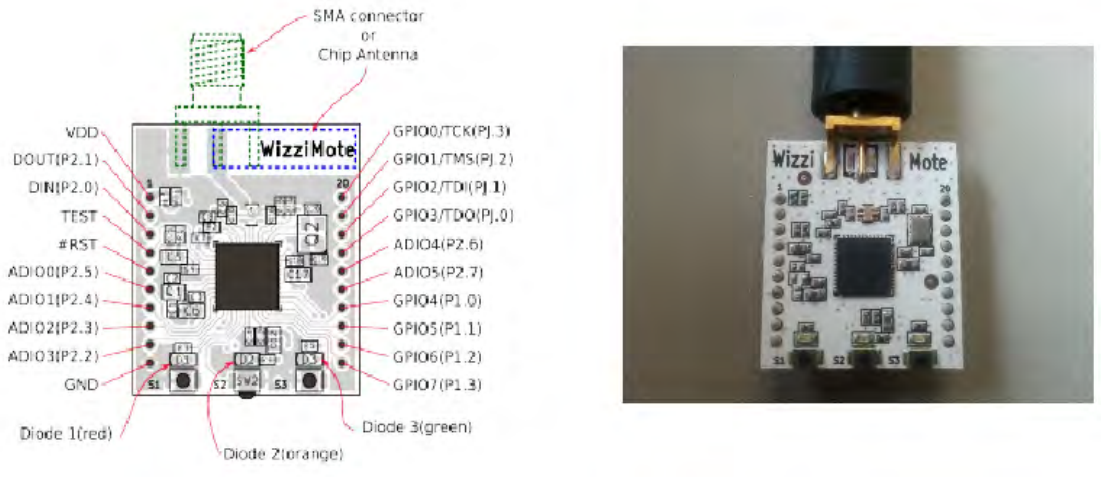


Figure 2.1: Wizzi Mote [9]

Some of the important pins used in the project have been described here. DIN (P2.0) and DOUT (P2.1) in Wizzi Mote Board are specifically assigned for Serial Communication

(UART) through port mapping technique of microcontroller. Port mapping is a technique by which the functionality of UART pins UCBORX (P1.5) and UCBOTX (P1.6) are assigned to the pins of DIN (P2.0) and DOUT (P2.1), but before implementing port mapping it is necessary to check the port pins of the microcontroller in the data sheet that whether these pins can be mapped or not. GPIO 0 to 4 pins on this board have been used for the JTAG access. GPIO 6 (P1.2) and GPIO 7 (P1.3) used for the I2C interface. The board has three. push buttons as well as three leds [9].

2.2 CC430F5137 Microcontroller

CC430F5137 microcontroller is of 48 pins and can be powered from 1.8V to 3.6V. It is a 16 bit low-powered microcontroller. CC430F5137 has Port1, Port2, Port3 of 8 pins, while Port5 is of 2 pins. Some of the important features and peripherals of the microcontroller which have been used in the project are described below.

2.2.1 Clock Sources

Almost every Msp430 microcontrollers and CC430 family controllers have UCS (Unified clock System) module in it. This UCS module can provide three clock signals (MCLK, SMCLK, ACLK) which can be sourced from five different sources such as XT1CLK, REFOCLK, VLOCLK, DCOCLK, DCOCLKDIV, and XT2CLK.

- MCLK: Master Clock

  It is used as a high clock source for the CPU and system.

- SMCLK: Subsystem master clock

  It is also a high frequency clock and is used in peripheral modules. SMCLK can be selected by software.

- ACLK: Auxiliary Clock

  It is a low frequency clock of 32Khz. It can also be selected by software when we use peripheral modules.

In the project, SMCLK clock with a frequency of 4 Mhz has been used for normal processing of instructions. But we can use other clocks also for different peripherals.

2.2.2 Low Power Modes

Low Power Mode functionality helps to save the power in the controller. In the project, low power mode has been implemented whenever the controller is in idle state. CC430F5137 microcontroller has five low power modes. In the project low-power mode 0 has been used.

If the low power mode is activated, microcontroller remains in low power mode or in sleep mode unless it receives any interrupt.

- Low-power Mode 0

  In this mode CPU and MCLK are disabled, while SMCLK and ACLK remains active.

In the active mode of the microcontroller, it draws 160µA/Mhz [10]. When LPM0 (Low Power Mode 0) is activated then it draws approximately 85µA current [11].
The low power mode is activated by writing the below mentioned instruction.
__bis__SR__register(LPM0__bits+GIE);
As soon as the above instruction is executed, microcontroller goes in sleep mode with the interrupt enabled. The microcontroller comes to active mode when it receives any interrupt such as Watchdog timer interrupt, Timer interrupt, Serial UART or I2C serial interrupt and RF receiving data interrupt. The low power mode 0 can be turned off by the following instruction. ***__bis__SR__register__on__exit(LPM0__bits);***

2.2.3 Timers

Timer also plays an important role in the project to give a time delay by activating it in low power mode. The microcontroller has two timers TA0 and TA1 with 16-bit register TAR. Both timers have been used in the project. For the radio operations timer TA1 was already implemented in the software interface of Dash7 whereas TA0 has been implemented in the project for I2C operations to give a specified delay until the interfaced sensors give response and the value written in the EEPROM. These timers can be sourced from different clock sources. There are three different operating modes of timer which are stated below [12].

- Continuous Mode
- Up Mode
- Up/Down Mode

In the project, up mode of the timer with enabled timer interrupt in compare mode has been used to save the power consumption in the sensor nodes.

2.2.3.1 Up Mode

In the up mode of the timer, any desired value can be set in the capture/compare register TAxCCRx of timers, in order to give a delay to generate any output in the program. In the capture mode, any external event whether rising or falling edge can terminate the timer delay process by generating timer interrupt, whereas in compare mode the timer generates interrupt when the value of TAR register becomes equal to the value stored in TAxCCRx

register. TAR also rolls to zero after reaching the value which is set in the TAxCCRx register. Up mode of the timer is shown in figure 2.2.

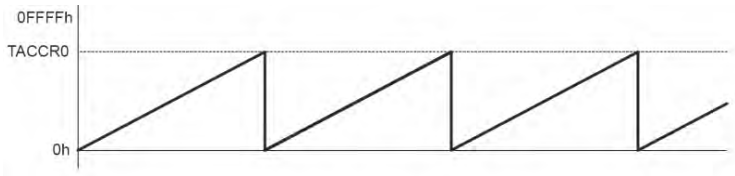


Figure 2.2: Up Mode

2.2.3.2 Timer registers

Different registers are used for the configuration of timer. Some important timer registers used in the project are described below [12].

2.2.3.2.1 TAxCTL Register

TAxCTL is a control register used for setting the clock source of timer, dividing the frequency clock and the different operating modes of timer can also be set in this register. It is a 16 bit register as shown in figure 2.3.

15	14	13	12	11	10	9	8
Reserved						TASSEL	
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
ID		MC		Reserved	TACLR	TAIE	TAIFG
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	w-(0)	rw-(0)	rw-(0)

Figure 2.3: TAxCTL Register [12]

TASSEL Bits

9^{th} and 8^{th} bits are used for the clock source of the timer.

- **00:** TAxCLK
- **01:** ACLK
- **10:** SMCLK
- **11:** INCLK

TAxCLK and INCLK are both external clocks. In the project, ACLK clock with default frequency of 32768 Hz has been used.

ID Bits

7^{th} and 6^{th} bits are used for dividing the input clock which in our case is ACLK clock .

- **00:** /1
- **01:** /2
- **10:** /4
- **11:** /8

`ID_3` divider mode (/8) has been in the project which makes the frequency from 32768 Hz to 4768 Hz.

MC Bits
5^{th} and 4^{th} bits are used for controlling the mode operation of timer.

- **00:** Stop Mode
- **01:** Up Mode
- **10:** Continuous Mode
- **11:** Up/Down Mode

As stated earlier, up mode of timer has been implemented in the project.

TACLR Bit
2^{nd} bit is used to clear the timer value stored in the TAR register.

TAIE Bit
1^{st} bit is used to enable the interrupt in timer. Interrupt is triggered when the timer value reaches 0xFFFF.

TAIF Bit
0^{th} bit is the timer interrupt flag and this bit is set when the timer overflows after reaching 0xFFFF.

2.2.3.2.2 TAxEX0 Register

TAxEX0 is a 16 bit register used for further dividing the clock frequency. 13 MSBs (15 to 3) are reserved bits and set to zero. The three least most significant bits (2 to 0) of this register with 8 different combinations can be used to divide the clock frequency further according to the requirement. In the project, we use the combination of 011 which is 4 in decimal. So the clock frequency is further divided by 4 which makes the ACLK frequency from 4768 Hz to 1768 Hz.

2.2.3.2.3 TAxCCTLn Register

TAxCCTL is a 16 bit Capture/Compare control register as shown in figure 2.4.

15	14	13	12	11	10	9	8
CM		CCIS		SCS	SCCI	Reserved	CAP
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	r-(0)	r-(0)	rw-(0)
7	6	5	4	3	2	1	0
OUTMOD			CCIE	CCI	OUT	COV	CCIFG
rw-(0)	rw-(0)	rw-(0)	rw-(0)	r	rw-(0)	rw-(0)	rw-(0)

Figure 2.4: TAxCCTLn Register [12]

Compare mode is activated by setting the 8^{th} bit of this register to "0" which is called CAP bit. 4^{th} bit is the Capture/Compare interrupt enable bit (CCIE), which activates the interrupt for the Capture/Compare mode. In the project, this bit is activated.

2.2.3.2.4 TAxCCRn Register

TAxCCRn is a 16 bit register which is used to hold the desired delay time in the timer and this value is compared with the TAR register.

2.2.4 WatchDog Timer

The main function of watchdog timer is to prevent our system against any failure. If any software problem occurs in our system then the watchdog timer allows the system to reset. It was required to use this timer in the project because if any of the interfacing devices (Sht25, Tmp100 and EEPROM) or any peripheral does not work properly so a system reset is required instead of hanging. Watchdog timer has a counter of 16-bit register WDTCNT. The watchdog timer becomes active as soon as the controller is turned on. This counter must be continuously cleared in our program so that our microcontroller should not be reset without any malfunction. This process is called Kicking the Dog. Watchdog Timer also allows different clock sources for its counter. The WatchDog Timer has a 16 bit control register used for the configuration of this timer as shown in figure 2.5.

15	14	13	12	11	10	9	8
WDTPW							
7	6	5	4	3	2	1	0
WDTHOLD	WDTSSEL		WDTTMSEL	WDTCNTCL	WDTIS		
rw-0	rw-0	rw-0	rw-0	r0(w)	rw-1	rw-0	rw-0

Figure 2.5: WDTCTL Register [12]

WDTPW Bits

WDTPW bits (15 to 8) are used for the protection against any accidental writes by writing the password 0x5A in the upper byte. Reading this upper byte shows 0x69 value. Value other than 0x5A will cause the controller to reset.

WDTHOLD Bit

7^{th} bit is the WDTHOLD bit, which is the hold bit. Setting the value of WDTHOLD to "1" stops the watchdog timer and its operation is resumed again by assigning the WDTHOLD bit to zero.

WDTSSEL Bits

6^{th} and 5^{th} bits are the WDTSSEL bits used for selecting the clock sources of the watch dog timer.

- **00:** SMCLK
- **01:** ACLK
- **10:** VLOCLK
- **11:** X_CLK

By default SMCLK is the clock source of this watchdog timer but this can be changed by the WDTSSEL bits.

WDTTMSEL Bit

WDTTMSEL is the 4^{th} bit which provides the option to change it into normal timer mode and does not reset the controller.

- **0:** Watchdog mode
- **1:** Interval timer mode

WDTCNTCL Bit

3^{rd} bit is the WDTCNTCL bit. Setting WDTCNTCL bit to "1" clears the watchdog timer counter to 0000h. In the software program, this bit is continuously set to "1" to clear the counter again to prevent overflow which causes the controller to reset.

WDTIS Bits

WDTIS bits (2 to 0) provide the division factor of the clock frequency and the interval of the watchdog timer can be selected from any of these combinations. The clock frequency is ACLK for the below mentioned watchdog timer interval.

- **000:** clock source/(2^{31}) (18 hr 12 min 16 sec)
- **001:** clock source/(2^{27}) (1 hr 8 min 16 sec)
- **010:** clock source/(2^{23}) (4 min 16 sec)

- **011:** clock source/(2^{19}) (16 sec)
- **100:** clock source/(2^{15}) (1 sec)
- **001:** clock source/(2^{13}) (250 ms)
- **110:** clock source/(2^{9}) (15.625 ms)
- **111:** clock source/(2^{6}) (1.95 ms)

In the project, WDTIS bits are assigned to 2(010) which sets the watchdog timer interval to 4 minutes and 16 sec.

2.2.5 I2C Serial Communication

For the interfacing of sensors and memory to the microcontroller, I2C serial communication interface has been used. I2C stands for Inter-integrated Circuit Bus. It was invented by Philips in 1982. The original throughput of I2C is 100 kbps, but some devices also support 400 kbps and 3.4 Mbps. I2C is also called two wire interface protocol because it uses only two lines for the data transfer between devices, so it has a two wire bidirectional lines:

- Serial Data Line (SDA)
- Serial Clock Line (SCL)

In the project, we have one master device (CC430F5137 microcontroller) and three slave devices (two sensors and one EEPROM memory) as shown in figure 2.6.

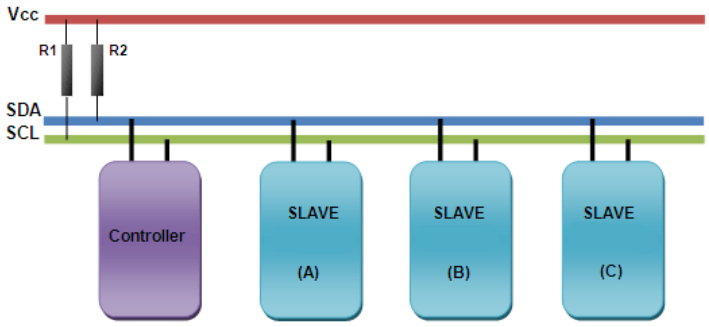


Figure 2.6: I2C bus with Master and Slaves [13]

Data is transferred through the SDA line whereas SCL line is the synchronisation clock for the data transfer. Devices attached to the I2C bus can be either Masters or slaves. The Master drives the SCL clock line. The master accesses every slave by its unique address and initiates the data transfer through SDA to these slaves and is also responsible for terminating the data transfer. Slaves can drive the data line, when it sends any data to the master. There

can be more than one master on I2C bus but only one master has a control of bus [14]. The pull up resistors are connected to both SCL and SDA lines. The slave can pull the data line low as acknowledgement signal, when the master accesses it by sending its address. If none of the slave devices sets the SDA line low (0) then it remains high (1) through pull up resistors. The value of these pull up resistances are selected by checking the bus capacitance and the rise time of the slave device in the data sheet. Normally it is selected in the range of 1-10K . In this project, pull up resistors with 4.7K values have been used.

2.2.5.1 I2C Protocol

In this section, the transmission of I2C protocol has been described.

- Start Condition

  At the start of transmission, the master sends the start condition informing all the slaves that something is going to be transmitted on the I2C lines. In the start condition(S), SDA line is pulled down while the SCL remains high as shown in figure 2.7.

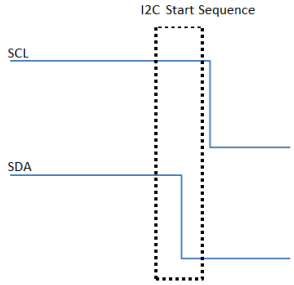


Figure 2.7: Start Condition

- Stop Condition

  When all the data is transferred between master and slave, then the master issues a stop condition and the bus becomes idle. This is done by releasing the SCL line followed by SDA line as shown in figure 2.8.

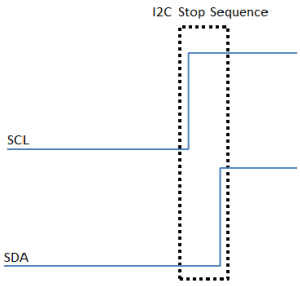


Figure 2.8: Stop Condition

- Device addressing

  The device addressing can be 7 or 10 bits. In the project, 7 bit addressing has been used. Every device has unique 7 bit address. The Master sends the device address on I2C bus after sending start condition. The MSB is transmitted first in the address and the last bit indicates the direction bit R/$\overline{W}$, which shows whether we want to read/write the data from/to the slave. The direction bit is not part of the slave address.

- Data Transfer and Acknowledgement

  The data sent over the I2C bus must be of 8 bits. The MSB is always sent first on the SDA line. The data on the SDA line must be valid or can be read when the SCL line is high and the state of the SDA line can be changed while the SCL line is low. On receiving every byte, the slave acknowledges it by pulling the SDA line low. When the slave sends data to the master, the master also acknowledges it in the same way. The master gives no acknowledgement signal when it receives the last byte from the slave followed by the stop condition.

2.2.5.2 I2C Protocol Example

Figure 2.9 shows the I2C example with one slave. In this example, the master is in receiver mode.

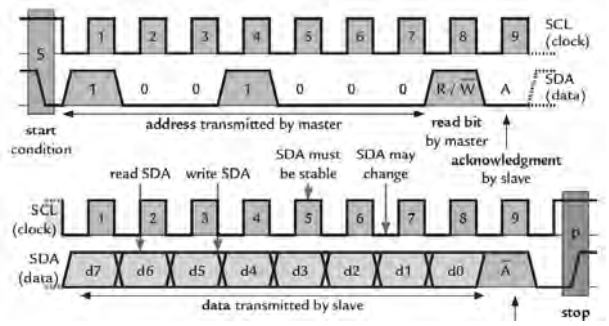


Figure 2.9: I2C Example Diagram [15]

In the above figure, SCL line takes 8 clock cycles for the the master to send 7 bit device address of slave followed by the read direction bit (R/$\overline{W} = 1$) after the start condition. The slave acknowledges its address by pulling the SDA line low and SCL detects it on 9^{th} clock cycle. The data on SDA line is changed when the SCL is low and remains valid until SCL becomes high. In the next cycle, the slave transmits 8 bits data to the master and the master terminates this process by sending not acknowledgement bit followed by stop condition.

2.2.5.3 CC430F5137 I2C Module

USCI_BO module in the controller supports both I2C and SPI. This module supports maximum 400 Kbps in I2C mode. There are four different modes of I2C in USCI_BO module which are master transmitter, master receiver, slave transmitter, or slave receiver mode [12]. In the project, master transmitter and receiver mode has been used. Some important registers used in this project have been described below.

2.2.5.3.1 UCBxCTL0 Register

UCBxCTL0 is a 8 bit control register 0 as shown in figure 2.10.

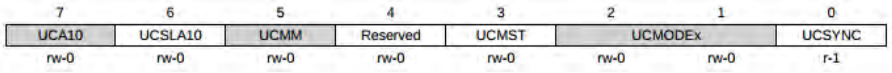


Figure 2.10: UCBxCTL0 Register [12]

UCA10 Bit
UCA10 is the 7^{th} bit, which allows to select the master own address in 7 bit or 10 bit.This bit is used when multiple masters are used.

- **0:** 7 bit address
- **1:** 10 bit address

UCSLA10 Bit
6^{th} bit is the UCSLA10 Bit. This bit is used to select the 7 bit or 10 bit address of slave.

- **0:** UCTXSTT bit slave address
- **1:** 10 bit slave address

In this project, 7 bit slave address has been used.

UCMM Bit
5^{th} bit is the UCMM bit. Multi master mode can be selected by setting UCMM bit to "1".

- **0:** Single Master
- **1:** Multiple Master

In this project, this bit is always zero because single master (controller) has been used.

UCMST Bit
UCMST is the 3^{rd} bit. This bit makes the USCI_BO module to operate as Master or Slave.

- **0:** Slave mode
- **1:** Master mode

As only one microcontroller has been used in this project, so it is necessary to operate this module as Master.

UCMODEx Bits

The 2^{nd} and 1^{st} bits (UCMODEx) enable to use this module in I2C mode or SPI mode. The first three combinations are used for different modes in SPI mode while the fourth combination is used for I2C mode.

- **00:** SPI mode
- **01:** SPI mode
- **10:** SPI mode
- **11:** I2C mode

In this project, the fourth combination (UCMODE_3) has been used.

UCSYNC Bit

The 0^{th} bit (UCSYNC) is the selection of transmission in synchronous or asynchronous mode. I2C is synchronous protocol so this bit is set to "1" to enable it in the synchronous mode.

- **0:** Asynchronous
- **1:** Synchronous

2.2.5.3.2 UCBxCTL1 Register

UCBxCTL1 is also a 8 bit control register 1. The bits in this register is shown in figure2.11.

7	6	5	4	3	2	1	0
UCSSELx		Reserved	UCTR	UCTXNACK	UCTXSTP	UCTXSTT	UCSWRST
rw-0	rw-0	r0	rw-0	rw-0	rw-0	rw-0	rw-1

Figure 2.11: UCBxCTL1 Register [12]

UCSSELx Bits

Bit 7^{th} and 6^{th} (UCSSELx) are used for the selection of clock sources in the module.

- **00:** UCLKI
- **01:** ACLK

- **10:** SMCLK
- **11:** SMCLK

UCSSEL_2 (SMCLK) is the clock source with a frequency of 4Mhz used in this project.

UCTR Bit

Bit 4^{th} (UCTR) sets the master in either transmitting or receiving mode depending on the value of this bit.

- **0:** Master in receiver mode/Master can read data
- **1:** Master in transmitter mode/Master can write data

It can also be described that the UCTR bit allows to switch the selection of read or write operation for the Master.

UCTXNACK Bit

Bit 3^{rd} (UCTXNACK) when set to "1", enables the master to transmit the not acknowledge signal to the slave when it receives all the required data from the slave.

- **0:** Send acknowledge signal
- **1:** Send not acknowledge signal

UCTXSTP Bit

Bit 2^{nd} (UCTXSTP) when set to "1", the master generates the stop condition. This bit is automatically set to zero again after generating the stop condition. This bit must be sent by the master, when the master receives the last byte from the slave or transmits the last byte to the slave as an indication of the termination of the data.

- **0:** No stop condition
- **1:** Send stop condition

UCTXSTT Bit

Bit 1^{st} (UCTXSTT) when set to "1", the master generates the start condition which alerts all the slaves that something is going to be transmitted on the bus.

- **0:** No start condition
- **1:** Send start condition

This bit is also automatically set to zero when the start condition as well as the slave address is transmitted.

UCSWRST Bit

Bit 0^{th} (UCSWRST) is used to reset the USCI_BO module when set to "1".

- **0:** clear reset
- **1:** Enable reset

2.2.5.3.3 UCBxBR0 Register

UCBxBR0 is a 8 bit baud rate register, used for dividing the SCL clock frequency making it compatible to the maximum allowed transmission rate in the USCI_BO module. As it was stated earlier, USCI_BO module supports maximum 400 kbps. UCBxBR0 with a value of 12 assigned to it in the project, which makes the frequency around 301.7Khz.

2.2.5.3.4 UCBxSTAT Register

UCBxSTAT Register is a status register to check the status of some bits as shown in figure 2.12.

7	6	5	4	3	2	1	0
Reserved	UCSCLLOW	UCGC	UCBBUSY	Reserved			
rw-0	r-0	rw-0	r-0	r0	r0	r0	r0

Figure 2.12: UCBxSTAT Register [12]

In this project, only 4th bit (UCBBUSY) has been used. UCBBUSY bit informs the status of bus. When set to "1", it means the I2C bus is busy otherwise idle. When the master sends the start condition bits, UCBBUSY is always set to "1" and cleared automatically upon sending the stop condition bits. This bit is always checked before sending the start condition to the slave devices, if this bit is found "1" then it is necessary to reset it otherwise the slave does not acknowledge its own address.

2.2.5.3.5 UCBxRXBUF Register

UCBxRXBUF is a 8 bit receive register. When the complete 8 bit value from the receive shift register is written in the UCBxRXBUF register, receiver interrupt flag UCRXIFG flag is set. When this register is accessed by the user to read the value in this register, UCRXIFG flag is automatically reset.

2.2.5.3.6 UCBxTXBUF Register

UCBxTXBUF is a 8 bit transmit register. 8 bit data is written by the user in this register to transmit, which clears the transmit interrupt flag UCTXIFG also.

2.2.5.3.7 UCBxI2CSA Register

UCBxI2CSA is a 16 bit slave address register. 7 bit or 10 bit slave address is written in this register and the bits are right justified. Bit 10 to 15 of this register are reserved. For slaves used in this project, 7 bit slave address has been used. Bit 6 of this register represents the most significant bit of the slave address.

2.2.5.3.8 UCBxIE Register

In the project, transmitting or receiving of data in the I2C has been performed with the help of interrupts. UCBxIE is a 8 bit register used for activating the interrupts as shown in figure 2.13. 7^{th} and 6^{th} are reserved in this register.

7	6	5	4	3	2	1	0
Reserved		UCNACKIE	UCALIE	UCSTPIE	UCSTTIE	UCTXIE	UCRXIE
r-0	r-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Figure 2.13: UCBxIE Register [12]

UCNACKIE Bit

5^{th} bit (UCNACKIE) is the not acknowledge enabling interrupt. If the slave devices send not acknowledge signal, not acknowledge interrupt flag (UCNACKIFG) is set to "1" and can be determined by enabling this interrupt which causes the program execution to jump in the interrupt service routine. This bit was used in the project when testing the slaves individually and then removed afterwards.

UCALIE Bit

4^{th} bit(UCALIE) is arbitration enabling interrupt. It is used when there are more than one master. This bit is of no use in this project.

UCSTPIE Bit

3^{rd} bit (UCSTPIE) is the stop condition enabling interrupt. If activated, the code execution jumps to the ISR when the stop condition is initiated by the master.

UCSTTIE Bit

2^{nd} bit (UCSTTIE) is the start condition enabling interrupt. If activated, the program execution also jumps to the ISR when the start condition is initiated by the master.

UCTXIE Bit

1^{st} bit (UCTXIE) used for activating transmit interrupt. When this interrupt is enabled, the program jumps to the ISR when the master sends the start condition and the slave address followed by the write direction bit.

UCRXIE Bit

0^{th} bit (UCRXIE) used for enabling receive interrupt. When this interrupt is enabled, the program jumps to the ISR when the master sends the start condition and the slave address followed by the read direction bit.

2.2.5.3.9 UCBxIV Register

UCBxIV is a 16 bit interrupt vector register as shown in figure 2.14.

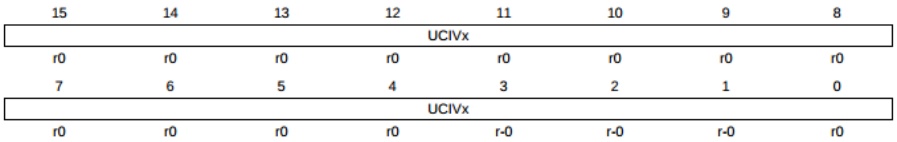


Figure 2.14: UCBxIV Register [12]

- 00h: No interrupt
- 04h: Not acknowledge interrupt flag (UCNACKIFG)
- 0Ah: Data received interrupt (UCRXIFG)
- 0Ch: Transmitter buffer is empty and new data can be written in the UCBxTXBUF register. This interrupt has the lowest priority

2.2.6 UART Communication

For sending the data from the Gateway to the PC, UART communication has been used. USCI module of CC430F5137 also supports UART serial communication. UART is an asynchronous serial communication when it is connected to an external device via PM_ UCA0TXD and PM_ UCA0RXD pins and it offers the communication on the same baud rate. The UART can send the data in 7 or 8 bit. The data format of sending a character in UART is shown in figure 2.15.

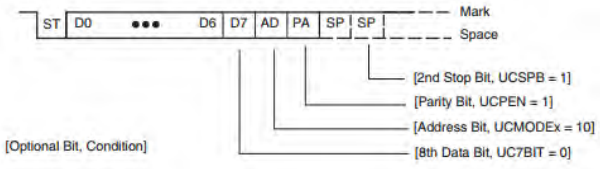


Figure 2.15: Data Format [12]

- ST: Start bit
- D0-D6: 7 bit data, LSB is transmitted first
- D7: D7 is the eight bit and it is sent when UC7BIT = 0 in the UCA0CTLO register
- AD: AD is the address bit which is sent when UCMODEx=10
- PA: PA is the parity bit(odd or even) which is sent when UCPEN=1 in the UCA0CTLO register
- SP: SP is the stop bit which indicates the termination of transmission and the second stop can also be send when UCSPB=1 in the UCA0CTLO register

2.2.6.1 UART Baud Rate and Clock

BRCLK is the clock source of UART, which is generated from SMCLK in the project. The different baud rates can be selected by the user according to the requirement which can be seen in the CC430 user's guide. In the project, baud rate of 115200 has been configured.

2.2.6.2 Transmit Data Register

UCA0TXBUF is a 8 bit register which contains the user data for transmission and the data is transmitted to the UCA0TXD pin of the controller via transmit shift register. When the UCA0TXBUFF register is empty, the corresponding UCTXIFG (transmit flag) sets to "1" indicating that the transmit data register is ready to accept the new data.

3 Hardware Components and I2C Implementation

In this chapter, all the main components of interfacing hardware with the microcontroller are described in detail. The key components of Hardware used are mentioned below.

- Tmp100 Sensor
- SHT25 Sensor
- Serial EEPROM
- Battery Cells
- PCB Designing Of Interfaced Devices

3.1 TMP100 Sensor Implementation

Tmp100 is a digital temperature sensor with I2C serial communication interface. It has operating temperature range of -55°C to +125°C. The user can select the resolution of this sensor according to requirement. It has resolution from 9 to 12 bits. Its accuracy is ±2.0°C and ±3.0°C with the temperature ranges of -25°C to +85°C and -55°C to +125°C respectively. It can be operated from 2.7V to 5.5V[16]. The circuit connection of Tmp100 is shown in figure 3.1.

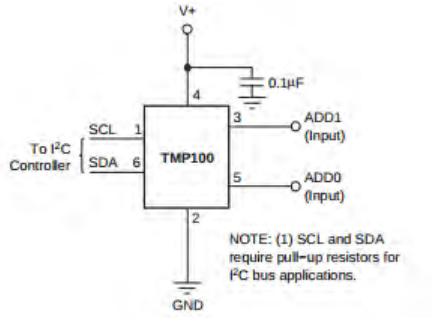


Figure 3.1: Tmp100 internal circuit [16]

Pin 1 (SCL) and Pin 6 (SDA) lines of this sensor require pull-up resistors and are connected to the micrcontroller SCL and SDA pins. Pull-up resistors of value 4.7 kΩ have been used in

the project. Pin 5(ADDO) and Pin 3(ADD1) address lines are used to interface 8 TMP100 sensors with different combinations but in the project we use only one combination. Pin 4 and Pin 2 represents Vcc and Gnd respectively. By pass capacitor of 0.1 μF is also used at the supply voltage.
This sensor is selected in the project because of its low cost, high operating range of temperature, user selectable resolution bits, I2C communication protocol and its low power consumption. The sensor is used for measuring the temperature inside the sensor node housing.

3.1.1 Internal registers of TMP100 Sensor

Tmp100 sensor has some internal registers. Three registers which have been used in the project are described below. The pointer register which can point or address the other registers, configuration register for the sensor and temperature register for reading the temperature value from the sensor. Register structure of Tmp100 is shown in figure 3.2.

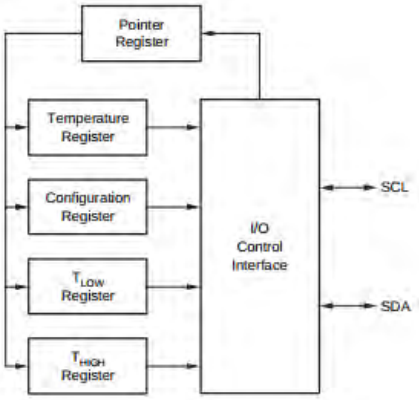


Figure 3.2: Internal Registers TMP100 [16]

3.1.1.1 Pointer Register

Pointer Register is a 8 bit register which is used to access all the registers of TMP100. For accessing the registers, only its 2 least significant bits are used out of 8 so the remaining 6 MSBs are assigned to zero as shown in figure 3.3.

P1	P0	REGISTER
0	0	Temperature Register (READ Only)
0	1	Configuration Register (READ/WRITE)
1	0	T_{LOW} Register (READ/WRITE)
1	1	T_{HIGH} Register (READ/WRITE)

Figure 3.3: Pointer Register addressing [16]

Upon power-up reset of Tmp100, pointer register always addresses to Temperature register. It means two LSBs are always zero after Tmp100 reset [16].

3.1.1.2 Temperature Register

Temperature register that holds the temperature value is a 12 bit read register. Two data bytes are needed to read this temperature register in which the 12 most significant bits correspond the value of temperature while the four LSBs are assigned to zero. After reading the two data bytes from Tmp100 and then shifting it four times by right shift operator yields the real temperature value [16].

3.1.1.3 Configuration Register

Configuration register is also an 8 bit register which is accessed through the pointer register. Bit 5 (RO) and Bit 6 (R1) in this register allows to set the resolution of the temperature values as shown in figure 3.4.

R1	R0	RESOLUTION	CONVERSION TIME (typical)
0	0	9 Bits (0.5°C)	40ms
0	1	10 Bits (0.25°C)	80ms
1	0	11 Bits (0.125°C)	160ms
1	1	12 Bits (0.0625°C)	320ms

Figure 3.4: Resolution bits [16]

If the resolution of sensor is increased, the conversion time required for the temperature value also increases. In the project, the resolution of the sensor has been set to 0.25°C (10 bits)[16].

3.1.2 I2C Communication with TMP100

Tmp100 sensor behaves either as slave transmitter or slave receiver device and is capable of supporting I2C protocol with the frequency from 400kHz to 3.4MHz. Timing diagram of I2C communication with Tmp100 in read mode is shown in figure 3.5.

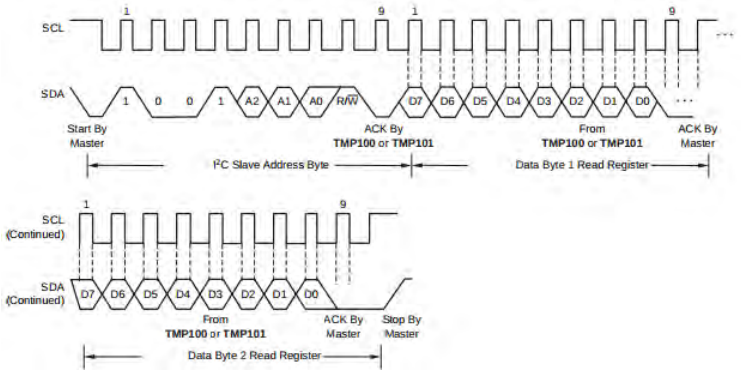


Figure 3.5: Read Data Timing Diagram [16]

When the microcontroller sends the slave address (Tmp100) over the I2C interface in read mode then the sensor starts to send the temperature values to the microcontroller. This

process is slave transmitter mode in Tmp100, because after acknowledging the address the slave transmits data to the microcontroller[16]. The diagram depicts that the master sends the stop bit after receiving two data bytes from the Tmp100 sensor.

3.2 SHT25 Sensor Implementation

Sht 25 is a digital temperature and humidity sensor with the I2C interface protocol and gives the reading very accurate and precise. Its operating voltage range is 2.1 to 3.6 V. It supports the I2C protocol with the clock of maximum 400 kHz. It is also 6 pin device with the DFN type package. It also consumes very low power. Its temperature range is -40°C to 125°C while the humidity range is 0 to 100% RH. Temperature accuracy is ±0.2°C and humidity accuracy is ±1.8% RH. The reason for selecting this sensor in the project is to monitor high accuracy humidity and temperature values inside the banana pallet, resolution of 0.01°C for temperature and 0.04% RH for humidity and low power consumption in active or inactive state[17]. The connection diagram of Sht25 is shown in figure 3.6.

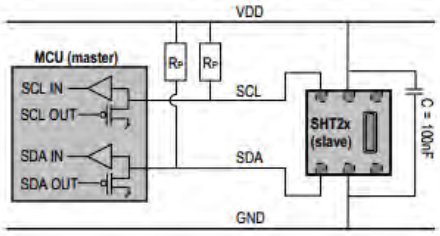


Figure 3.6: Sht25 and its connection [17]

It is clearly seen from the figure, Pin 1 (SDA) and Pin 6 (SCL) of the sensor are connected to the microcontroller SDA and SCL lines respectively. Pin 2 and Pin 5 corresponds to the Gnd and Vcc lines respectively. Pin 3 and Pin 4 remains unconnected [17].

3.2.1 Communication Modes in SHT25 sensor

There are two modes in Sht25 when it communicates with the microcontroller.

- Hold Master Mode
- No Hold Master Mode

3.2.1.1 Hold Master Mode

In the hold master mode, the sensor Sht25 halts the SCL line until it finishes the measurement process. Sht25 sets the SCL line to zero and the master is put on the hold state during the measurement of temperature or humidity and then releases the SCL line when the measurement process finishes. After the measurement process, the microcontroller can read the

temperature or humidity data from the sensor. The whole timing diagram is shown in figure 3.7.

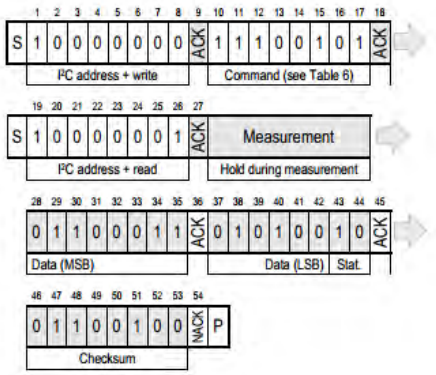


Figure 3.7: Hold Master Mode [17]

The microcontroller sends the Sht25 address with writing mode enabled. The sensor acknowledges the address on bit 9 and then the microcontroller transmits the hold master mode data of 8 bit starting from bit 10 for measuring temperature or humidity. The sensor acknowledges the transmitted data by setting the SDA line low on bit 18. To read the value of temperature or humidity, the microcontroller has to send the start bits again with the reading mode enabled and the sensor acknowledges the address on bit 27. After bit 27 it is clearly visible from the figure that no transmission is possible because the SCL is kept to zero value by the sensor in the measurement process. When the sensor releases the SCL line then the microcontroller reads the two data bytes of the temperature or humidity. The third byte CRC checksum is optional and can be omitted by sending the not acknowledgement followed by stop bit[17]. In the project, we use no CRC so we send the not acknowledgement bit followed by the stop bit.

3.2.1.2 No Hold Master Mode

In no hold master mode, the sensor does not force the SCL line to pull down so other communication can also be possible. In this case if we try to read the temperature or humidity values, after sending the start condition followed by the data which triggers the sensor in no hold master mode, the microcontroller continuously checks the internal processing of measurement. If the internal processing is not finished then the microcontroller receives the not acknowledge signal from the sensor. To overcome the polling process to some extent, it is advisable to give a delay of 20 µs after receiving the acknowledgement in no hold master mode of sensor. This communication is shown in figure 3.8.
Here in no hold master mode, the CRC checksum is an option but can be avoided in the same way by transmitting not acknowledgement and stop bit after bit 44. The 14 MSBs in the two data bytes received from the sensor indicate the data for either temperature or humidity

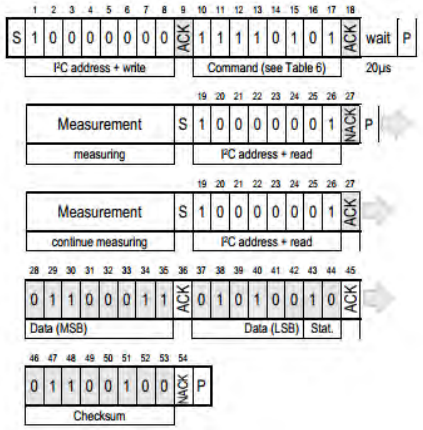


Figure 3.8: No Hold Master Mode [17]

value while the two LSBs are the status bits. Bit 43 indicates us that the data which we have obtained in 14 MSBs is of either temperature or humidity. If it is '0' then it indicates temperature data otherwise humidity data. Bit 44 is set to '0'. This also holds true in hold master mode[17].

In the project, hold master mode communication is used for the temperature and humidity values because in this mode the microcontroller does not have to do polling for receiving the data bytes from the sensor. Whereas in no hold master mode the sensor gives three or four times no acknowledgement signal when it is in the process of getting measurement before receiving the measured values.

3.2.2 Resolution bits in SHT25 sensor

The user register in Sht25 gives us the possibility to select the resolution of temperature from 11 bit to 14 bit and 8 to 12 bit for humidity. Default resolution of temperature in Sht25 sensor is 0.01°C (14 bit) and for humidity is 0.04% RH (12 bit). The recommended value of time delay to get such a resolution is typical 85 msec for temperature and 22 msec for humidity[17]. In the project, the default values of resolution for temperature and humidity values have been used.

3.2.3 Data Conversion for Temperature and Humidity

Before the conversion, we set the two least significant bits to zero which correspond the status bits in the two data bytes received from the sensor. The equation for getting the physical value of temperature, we use the below mentioned equation.

3.2.3.1 Temperature Output

Below mentioned formula gives us the reading in °C.

$$T = -46.85 + 175.72 \times \frac{S_T}{2^{16}} \tag{3.1}$$

In the above equation, S_T represents the value of 2 byte received from the sensor. T will give us reading in °C [17].

3.2.3.2 Humidity Output

The physical value of relative humidity can be received by the following mentioned equation.

$$RH = -6 + 125 \times \frac{S_{RH}}{2^{16}} \tag{3.2}$$

S_{RH} is the value of 2 byte received from the sensor. The value of RH denotes the relative humidity above liquid water [17].

3.3 Serial EEPROM Implementation

EEPROM stands for Electric Erasable Programmable Read Only Memory. It is a non-volatile memory, which means it stores its data after power shut down. So data storage in such a memory is permanent. CC430F5137 has also 32 Kb flash memory and it initializes its peripherals from this memory like Boot code and Boot strap loader etc. So special precaution is necessary to write data inside this memory. This flash memory has also 4 blocks named as info flash of size 128 bytes from info A to info D and in these blocks one can write and read its data easily [12]. But the size of these blocks is not enough for our application and the old data must be erased before writing the new data. Interrupt mode can not be used in these flash memory and this is also a big disadvantage for an application like our project, where low power operations are highly required.
EEPROM has the feature of erasing the older data replacing with the new data. It means we have the possibility of re-writing the data into the EEPROM. In this project 24AA64 Serial EEPROM has been used. It is also a device with an I2C interface. It has a capacity of 64Kb (64 kilo bits) or 8KB (8 kilo bytes), which means it can store 8 bit data in its 8192 locations. Such a EEPROM was required in the project to store temperature value of Tmp100 sensor and humidity and temperature values of Sht25 sensor, because sometimes RF link is weak from the node to gateway and there is the possibility of losing the data. In order to prevent this loss, such a permanent storage was necessary in EEPROM. This 24AA64 has the supply voltage from 1.8V to 5.5V and supports the I2C communication with clock frequency

of 400KHz. It has 1 million erase cycles which is sufficient in this project. It can retain its data over 20 decades[18]. The pin diagram of 24AA64 is shown in figure 3.9.

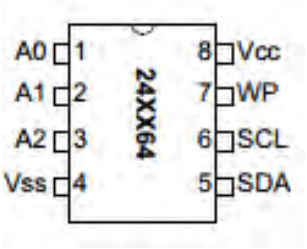


Figure 3.9: 24AA64 Pin diagram [18]

24aa64 is 8 pin IC. A0, A1 and A2 represent the address configurations of 24aa64 EEPROM and make 8 possible addresses to access 8 EEPROMS. WP (write protection) pin 7 is used to allow writing data in EEPROM when it is set to zero but no data can be written when it is enabled. SDA (Pin 5) and SCL (Pin 6) are used for I2C communications with the pull-up resistors. Vcc (Pin 8) and Vss (Pin 4) are used for the supply voltage and ground respectively. It is also a low power consumption device, because in active state it draws current of 1 mA and when it is in standby mode it draws 1 µA current.

3.3.1 Device and Memory addressing in EEPROM

The EEPROM addressing is started by first sending the start bits for I2C protocol and then the slave address of EEPROM. The slave address of EEPROM consists of four control bits, three address bits and the eighth bit indicates the writing or reading mode of EEPROM. The device addressing is depicted in figure 3.10.

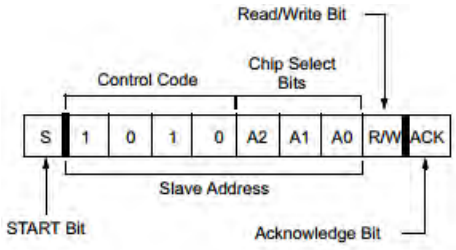


Figure 3.10: Device Addressing [18]

Whether data is written or read from the memory, it is necessary to address the memory cell of EEPROM before writing or reading in that memory cell. This is described in figure 3.11. After sending device address with either read or write mode, the address of the memory location for reading or writing. As in 24aa64, EEPROM has 8192 unique addresses in it so it requires 12 bits from A0 to A12 to access these memory cells while the other four bits are

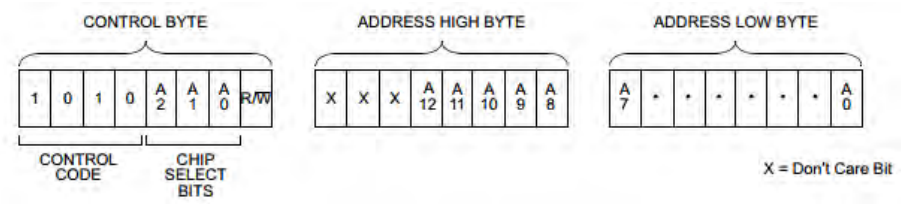


Figure 3.11: Memory location Addressing [18]

don't care bits. The first byte to be sent after the device addressing is the higher address byte and then the lower address as the second byte [18].

3.3.2 Write cycle in EEPROM

Data can be written in the memory cell of 24aa64 in two ways.

- Page write
- Byte write

3.3.2.1 Page write

In the page write operation of 24aa64 EEPROM, it is possible to write 32 bytes data in the memory at once without the need for starting the I2C device address again and again in order to send 32 bytes. Page write cycle is shown in figure 3.12.

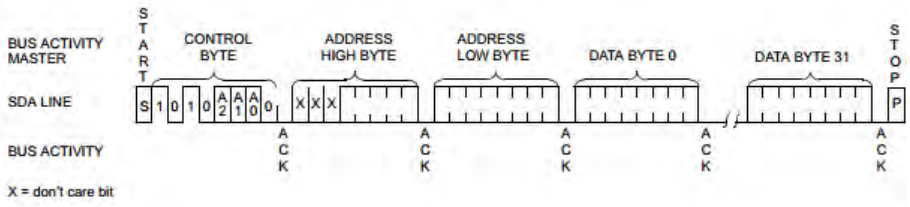


Figure 3.12: Page Write Cycle [18]

From the above figure, it is clearly visible that we send only the start, control byte and device address bits once and then the 32 bytes data can be written in the memory locations. After sending 32 bytes, stop bits transmission are sent to finish this page write.

3.3.2.2 Byte write

In this project we are using byte write operation because this operation meets our requirement to write byte data in EEPROM. Byte write cycle is shown in figure 3.13. The stop transmission is generated to complete this byte write cycle after writing the data byte in the memory location addressed by the address bits A0 to A12. It is recommended to give maximum delay of 5msec after every byte write or page write [18].

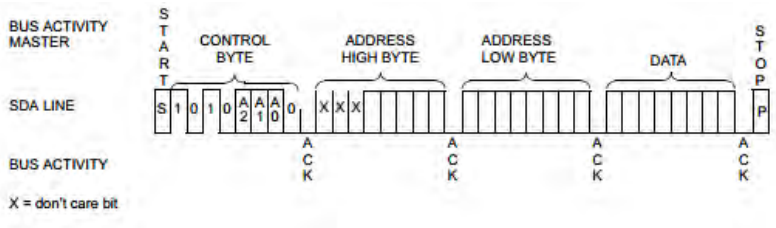


Figure 3.13: Byte Write Cycle [18]

3.3.3 Read cycle in EEPROM

24aa64 EEPROM provides three ways to read data from its memory location, which are stated below.

- Current Address Read Method
- Random Address Read Method
- Sequential Address Read Method

3.3.3.1 Current Address Read Method

In the current address read method, a single byte can be read from the memory location of EEROM after sending the device address and memory address bytes of that location. On reading for the next byte from the EEPROM, it is not required to send the desired memory address bytes because 24aa64 has the internal counter which stores the last accessed location in EEPROM. So when the start transmission of the device in read mode is sent over I2C interface, the counter points to the next memory location and the next byte can be read from the memory. Upon every received byte acknowledgement, it is necessary to send the stop bits. It is shown in figure 3.14.

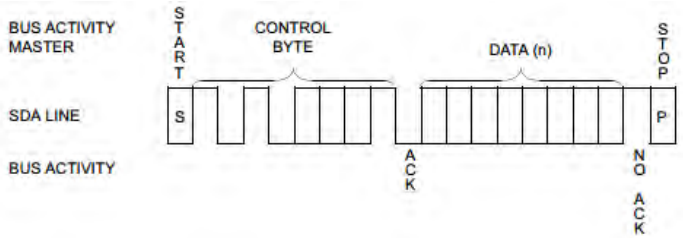


Figure 3.14: Current Address Read [18]

3.3.3.2 Random Address Read Method

The random address read method is used to read the data byte from any random memory address. The device address or control byte is sent following the start bits with the writing

mode enabled, the two memory address bytes are sent after the device address acknowledgement. Upon acknowledgement of the address bytes, the microcontroller again restarts the start transmission of I2C with the control byte in reading mode. The EEPROM sends the 1 byte data to microcontroller from the addressed location of memory. After receiving the byte, the microcontroller generates the stop transmission. This whole process can be easily understood from the figure 3.15.

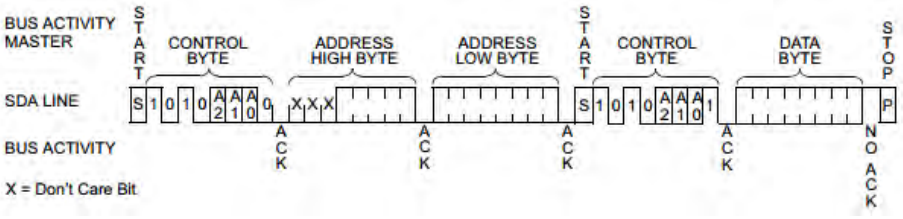


Figure 3.15: Random Address Read [18]

3.3.3.3 Sequential Address Read Method

Unlike random address read method, in the sequential address read method the acknowledge bit is sent after receiving the first byte of data from EEPROM and then sequentially receives the data bytes up to a given value as shown in figure 3.16.

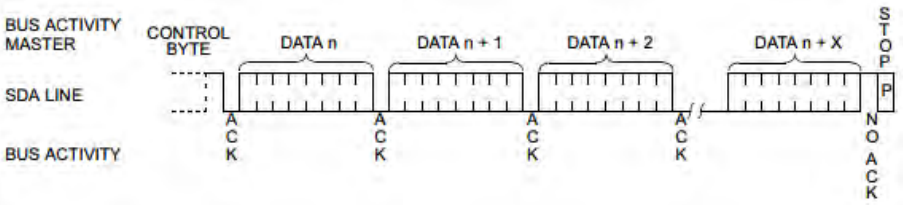
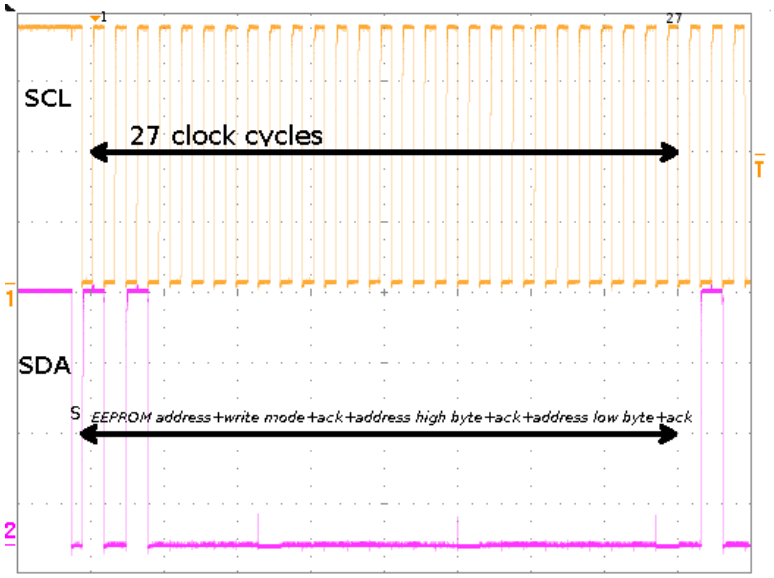


Figure 3.16: Sequential Address Read [18]

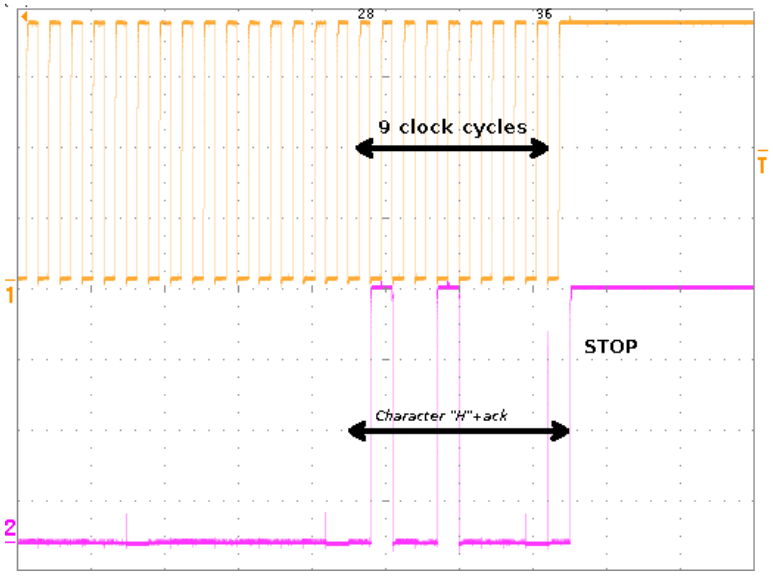
The whole starting process is same in sequential address read like random read. This approach has been used in this project to read the data from the memory, because more than one byte was required to receive the data in one shot.

3.3.4 Demonstration of Timing Diagram on Oscilloscope

The timing diagram on oscilloscope for writing a single byte of value "H" in the EEPROM location (0x0000) is shown in figure 3.17.
In the figure 17-a, start condition (S) is initiated by pulling the serial data line (SDA) line low while the serial clock line (SCL) remains high at that time. Following the start condition, the master sends the EEPROM address (1010000) and the write enabled bit (0). After addressing the EEPROM, the high byte address (00) and low byte address (00) is transmitted to access



(a) Single Byte Write in EEPROM



(b) continued:Single Byte Write in EEPROM

Figure 3.17: Oscilloscope Timing Diagram

the first memory location of EEPROM for writing data in it. The slave (EEPROM) gives acknowledgement to the microcontroller by pulling SDA line low after receiving every byte from the microcontroller. Total 27 clocks of SCL required to transmit three bytes of data to the EEPROM. In the figure 17-b, the SCL line requires 9 clocks to write the character value "H" (01010000) in the memory location of EEPROM. The transmission is terminated when the rising edge of SDA is detected by the SCL line.

3.3.4.1 Rise Time of Clock and Data

The rise time of SCL clock and data line SDA can be seen from the figure 3.18

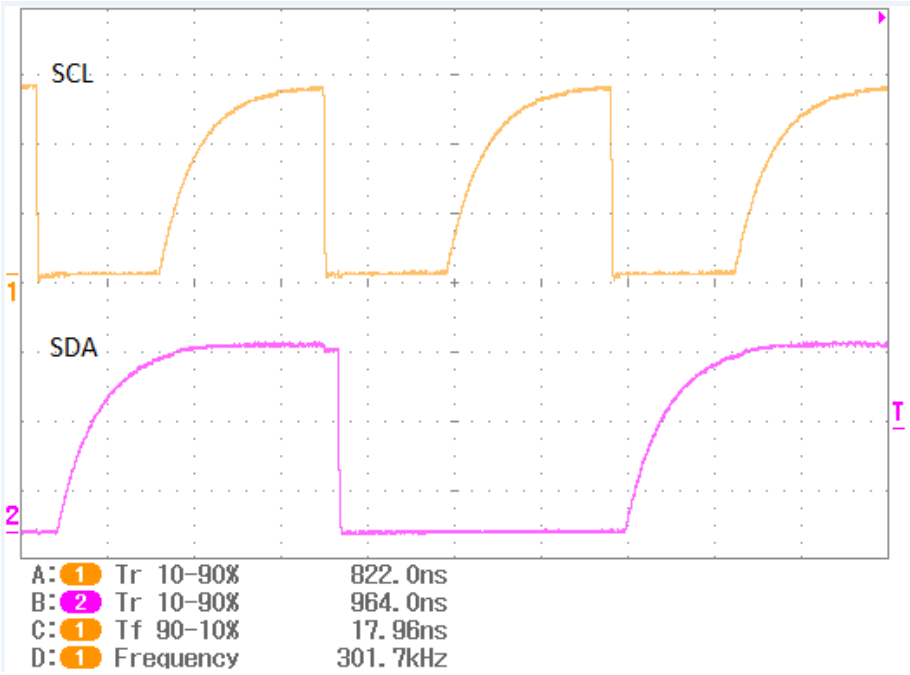


Figure 3.18: Rise Time of SCL

The rise time of SCL and SDA with the values 822 ns and 964 ns respectively was measured when writing a byte in EEPROM at a SCL frequency of 301.7Khz. The fall time of SDA with the value 17.96 ns was observed.

3.3.5 Prevention of Overwriting data in EEPROM

When the sensor node is powered off after storing sensors data in the EEPROM, the EEPROM retains all data permanently. But the problem arises when the sensor node is restarted again, sensors data are written to the same locations in the EEPROM starting from the beginning (overwritten data) instead of storing the sensor values from where it left before. This happens because the program execution starts from the beginning and all the devices are initialized again. To avoid such overwriting of data in EEPROM, two procedures were proposed.

- Find raw values (0xFF) in EEPROM memory cells
- Tracking of last accessed location in EEPROM

3.3.5.1 Find raw values (0xFF) in EEPROM memory cells

By default, the value in all the memory cells of EEPROM is either 0xFF or 0x00. In 24aa64 EEPROM, the raw value of 0xFF lies in all the memory locations as shown in figure 3.19.

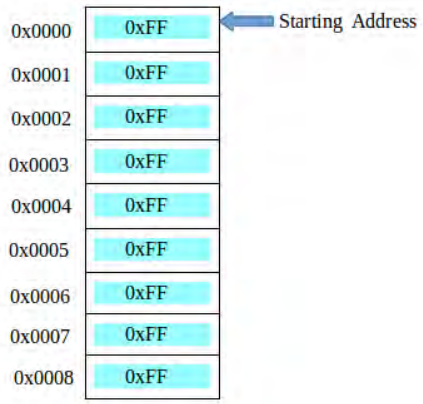


Figure 3.19: Raw Default Value in EEPROM

Overwriting of the sensor values in the memory can be prevented by finding the value 0xFF in memory location, when this value is found it becomes the starting address to write the next block of the sensor values ending with the last byte 0xFF as shown in figure 3.20.

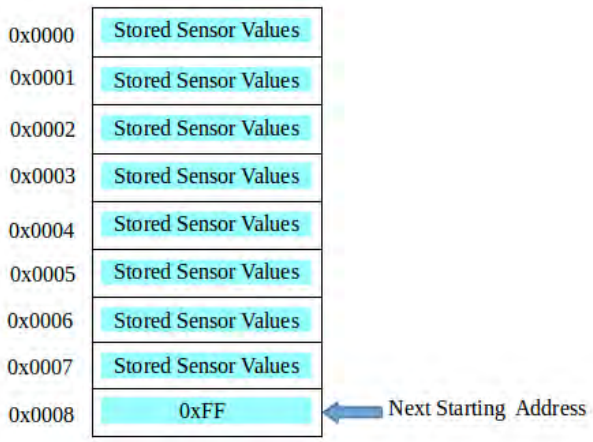


Figure 3.20: Starting address of the next sensor data

The last byte 0xFF must be added with the sensor data block because if all the locations in the EEPROM are filled with sensor values and the algorithm does not find any location with 0xFF value, it is not possible to write the sensor value furthermore. In this approach, the algorithm always reads the locations of the EEPROM to find 0xFF in order to write sensor values.

3.3.5.2 Tracking of last accessed location in EEPROM

To prevent overwriting in EEPROM, tracking of last accessed memory location is also the solution. In this case, the first two locations (0x0000 and 0x0001) in the memory cell of EEPROM have been reserved for storing the address of the next memory location following the last accessed memory location as shown in figure 3.21.

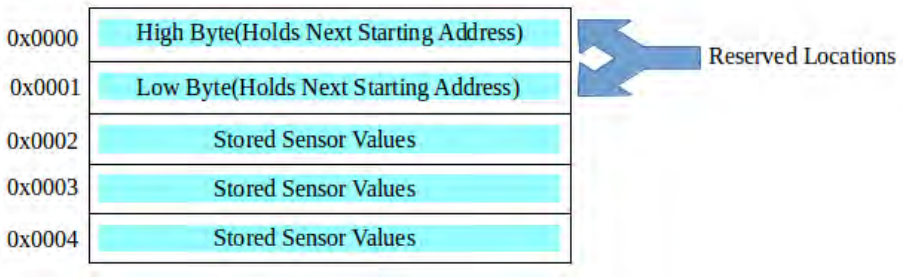


Figure 3.21: Magic Locations

When the sensor node restarts it first reads the first two reserved cells in the EEPROM, which informs the node to store the sensor values in the EEPROM at the memory location cell where it stopped before. This procedure has been implemented in the project because of its faster computation than reading the memory every time to find the 0xFF value. In the project, block size of 16 bits are stored in the memory and then the next corresponding address is stored in the first two locations of EEPROM. Description of the block size of 16 bits will be explained in the coming chapters.

3.3.5.3 Data Storage in EEPROM

The data in the EEPROM is stored in a size of 16 bytes after taking two measurement values from the Sensor nodes. Figure 3.22 shows the data storage pattern in EEPROM. The first two locations (0x0000 and 0x0001) of EEPROM always informs the next starting address to write the values in the EEPROM. The data in the other locations are described as below.

- Restart Node Status: 0x0002 cell has the Restart Node Status which informs whether the Sensor node has been reset or not. Value "1" denotes reset state and "0" denotes not reset state.
- RSSI Gateway value: 0x0003 location has the RSSI Gateway value which tells the received signal strength indicator signal of Gateway when the Gateway addresses any Sensor Node.
- Tmp100 Sensor data: 0x0004 and 0x0005 cells have the high byte and low byte value of the temperature from Tmp100 respectively.

- Sht25 sensor Temperature value: 0x0006 and 0x0007 cells have the high byte and low byte value of the temperature from Sht25 respectively.

- Sht25 sensor Humidity value: 0x0008 and 0x0009 cells have the high byte and low byte value of the humidity from Sht25 respectively.

The same sequence starts again from 0x000A location of the EEPROM.

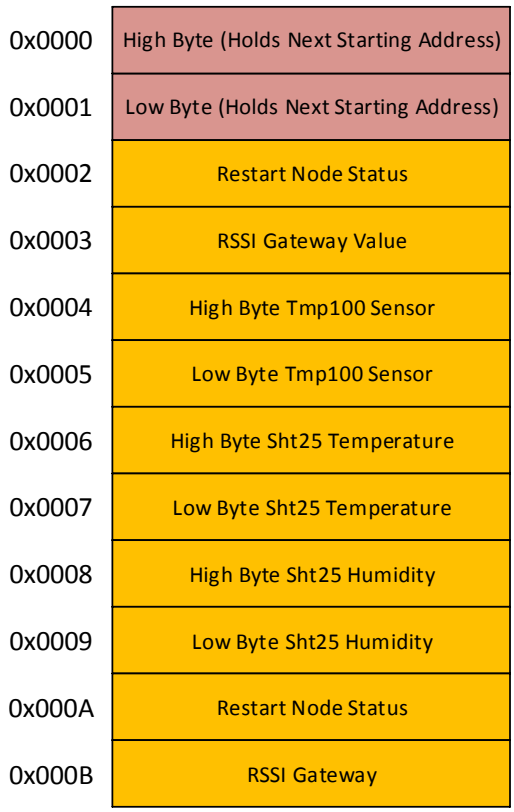


Figure 3.22: Data Storage in EEPROM

3.4 Battery Cells

Lithium 3V battery with a capacity of 850 mA has been used to power up the nodes in the project. This battery has dimension of 14.5x25mm. This battery along with sensor node was placed inside the housing.

3.5 PCB Designing Of Interfaced Devices

For the interfaced devices such as Tmp100, Sht25 and EEPROM, two layered PCB (Printed Circuit Board) with dimension of 26x42 mm was designed in the Eagle software. This software can be downloaded free from the internet and it provides 2 layer PCB option on student version which was sufficient in the project. Two PCBs were needed in the project for interfacing. First PCB is to monitor the temperature inside the housing and the second to measure temperature and humidity inside the banana pallet. So both circuits were made on a single PCB (26x42 mm) for the ordering of the PCB and then afterwards cut for separating them.

3.5.1 PCB Schematic Design

The schematic diagram of the two circuits made in the Eagle software are shown in figure 3.23 and figure 3.24..

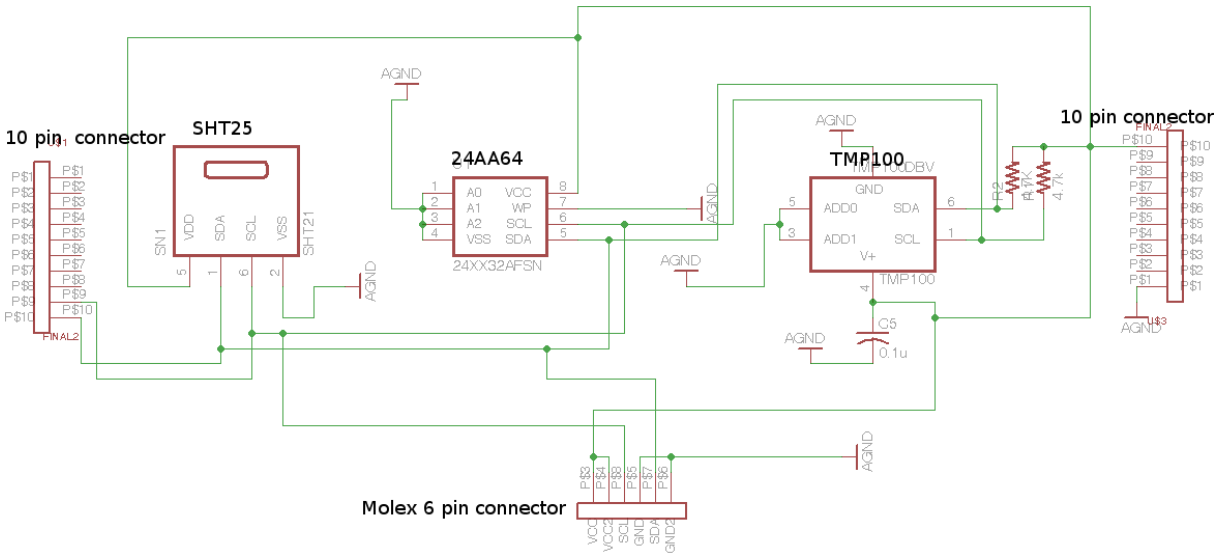


Figure 3.23: First Board Schematic Diagram

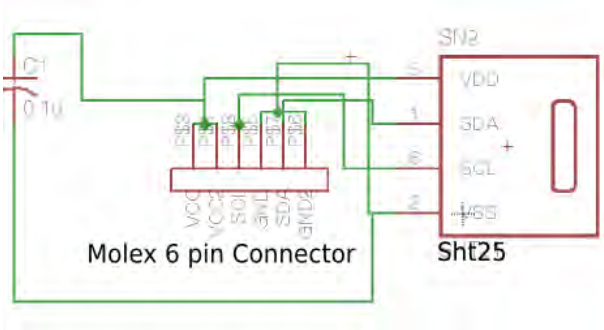


Figure 3.24: Second Board Schematic Diagram

In the first board Tmp100 sensor, EEPROM, two pull-up resistors and a bypass capacitor are in the circuit. Two ten pin connectors are attached to the Wizzimote (Controller). The 6 pin molex connector used to supply the voltage, ground and I2C bus lines (SCL and SDA) to the second board.
In the second board the Sht25 sensor must be outside the housing to measure the humidity and temperature inside the banana pallet so separate board was needed for this sensor. The 6 pin molex connector is used to power up the sensor and driving the sensor by I2C bus lines.

3.5.2 PCB Board Design

The board design of PCB is shown in figure 3.25.

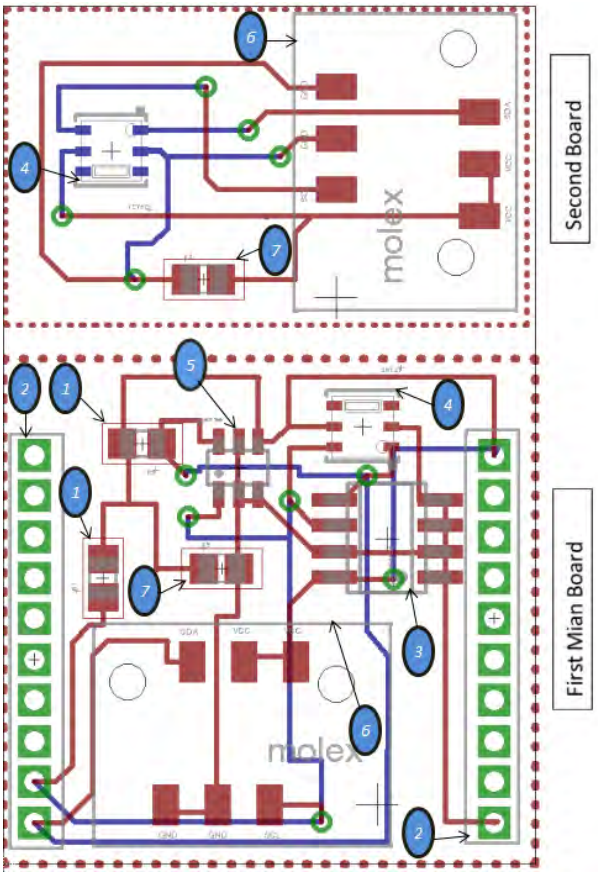


Figure 3.25: Board Diagram

The red line connections represent the upper layer on the PCB while the blue line connections represent the lower layer. The numbered symbols represent the foot prints of the component used.

1. Pull-up resistors foot print (package 0805)
2. 10 pin connector and its footprint were also developed in Eagle

3. 24aa64 EEPROM foot print
4. Sht25 sensor foot print
5. Tmp100 sensor foot print
6. Molex 6 pin connector and the foot print was developed in Eagle
7. Capacitor foot print (package 0805)

3.5.3 Complete PCB design

The complete PCB layout diagram is shown in figure 3.26. The ground layer lies in majority on the top surface area layer of PCB.

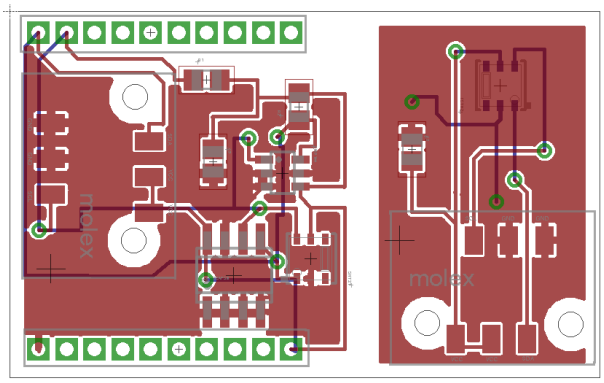


Figure 3.26: Complete Layout Diagram

The two PCB boards with the components soldered is shown in figure 3.27. The soldering of Sht25 was performed in the IMSAS (Intitute for microsensors, actuators and systems) clean room because the size of Sht25 sensor is 3x3 mm.

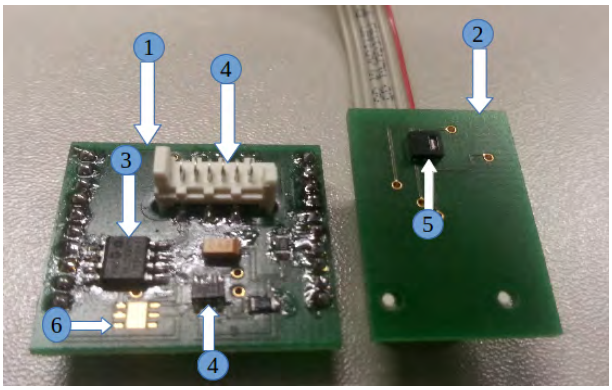


Figure 3.27: PCBs With Components

1. Main PCB board
2. Second board
3. 24aa64 EEPROM
4. Tmp100 sensor
5. Sht25 sensor
6. Sht25 footprint was developed on the main PCB board, if required in the future

The Sensor node housing is shown in figure 3.28. Sht25 board has been glued to the right wall of the sensor housing after drilling a small hole on it so that it can measure the outside temperature and humidity.

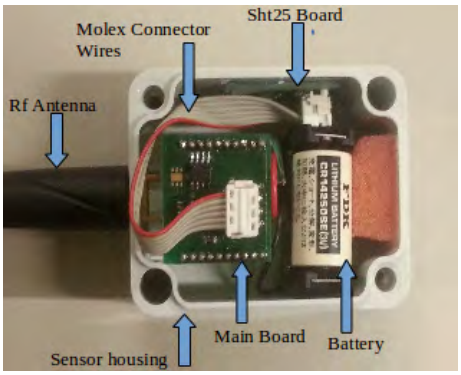


Figure 3.28: Sensor Node Housing

4 DASH7 Alliance Protocol

This chapter gives a short description for the Dash7 and its alliance protocol (OSS-7). Dash7 alliance protocol has been used to send the data over RF link at a frequency of 433Mhz.

4.1 Dash7 Introduction

Dash7 stands for "Developer Alliance for standard harmonization of ISO 18000-7" and developed to increase the low power consuming wireless devices. Dash7 alliance protocol software stack implementation has been used in the project and the Rf core modules of the CC430F5137 microcontroller is also configured according to the Dash7 protocol. Dash7 works on a BLAST concept

- Bursty: The communication is done abruptly and can not transfer like audio or video data.
- Light: It sends the light data with a packet size limitation of 256 bytes.
- Asynchronous: Transmission is done asynchronously.
- Transitive: Devices using Dash7 alliance protocol can be mobile and does not necessarily need to be fixed.

Dash7 uses the frequency spectrum in the range of 433-434.97 Mhz in Europe[19]. Due to less bandwidth, the high data rate transmission is not possible which is not a big concern in low power wireless devices. Range of Dash7 communication depends on several factors mainly the transmitted power or receiver sensitivity and the data rate transmission. High transmitted power increases the range but power consumption of the device also increases while the high data rate transmission becomes the cause for the reduction of range. 433.16 MHz spectrum has been configured in the Rf radio core of microcontroller whereas the antenna transmission power is set to 10 dBm (Decibel milli watts) which is equal to 10 mW. The default data rate in the Dash7 module is set to 55 Kbps and can be changed accordingly.
The communication mode implemented in the project is Pull based in which the Base Station (Gateway) sends the query to the sensor node with its device id and the accessed node replies accordingly in response to the query of the Base Station. Dash7 supports different network topologies which can be used such as Star, Mesh and Tree. In the project, Star network topology has been implemented as shown in figure 4.1.

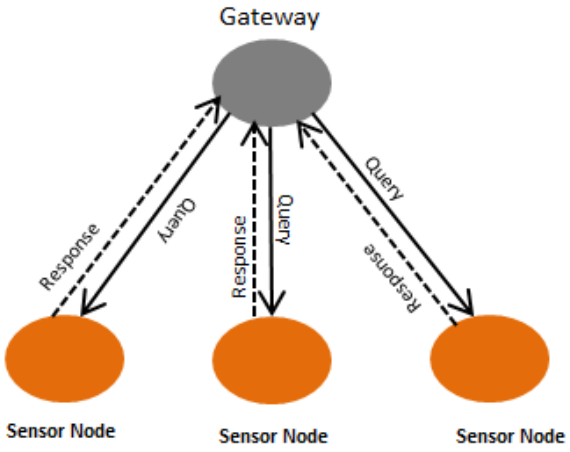


Figure 4.1: Star Topology

4.2 Dash7 Software Stack

OSS-7 (Open Source Stack for Dash7) is the software stack implementation of Dash7 and it is based on OSI layers as shown in figure 4.2.

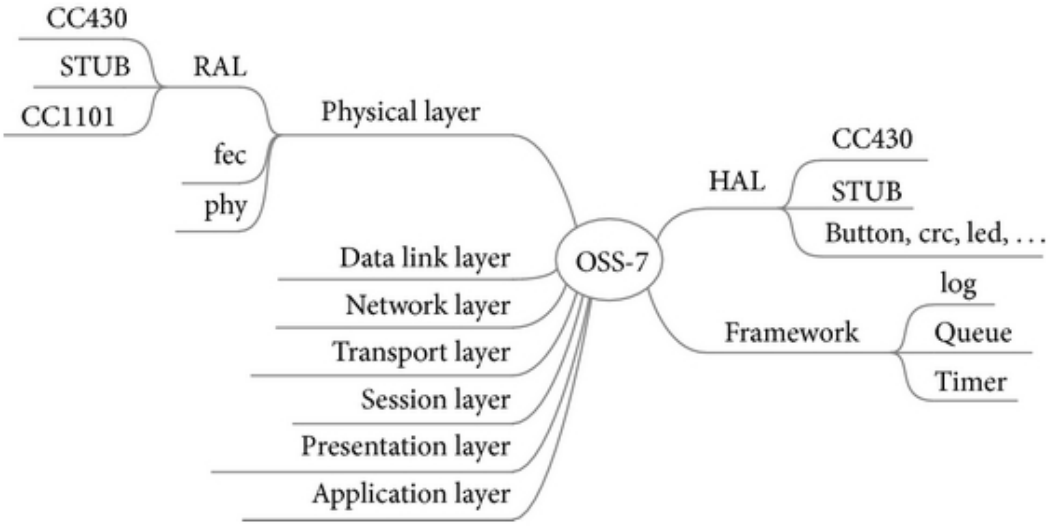


Figure 4.2: OSS-7[20]

In addition with the OSI layers, there are two more layers HAL (Hardware abstract layer) and Framework layer. In HAL the hardware APIs are implemented such as Timer, I/O, Uart, Led and button etc. One has to choose this HAL according to its microcontroller platform, for example WizziMote platform hardware has been selected in the project. Framework layer is used for logging, queuing and timer event function. Both these layers are available to use

in any OSI layer[20]. All the configurations of the radio core like transmitting or receiving data have been set in the Physical layer of OSS-7.

4.3 Dash7 Applications

Dash7 can be used in many applications but here are presented few of them.

4.3.1 Home Automation

High frequency signal is obstructed by the obstacle, but the Dash7 quality of penetrating walls makes it a good choice in the home automation system.

4.3.2 Parking Guidance

It can be used to guide the driver about the vacant parking slots in the parking area instead of searching the parking slot one by one. It can be done by placing the sensor in the parking slot.

4.3.3 Automotive Senors

Dash7 has already been employed in the automotive sector like tire pressure monitoring system (TPMS).

4.3.4 Military

Dash7 is also used in the military applications and in the future it can also be used to find the friendly personnel nearby by sending queries.

4.4 Dash7 Version

The Dash7 module used in the project was pulled on 18 June, 2014 from the official oss-7 Git Repository at https://github.com/CoSys-Lab/dash7-ap-open-source-stack.git. Master branch of this version has been used in the project.

5 Gateway and Node Communication

In this chapter, the detailed description of the communication between the Gateway and Sensor node will be explained.

5.1 Communication Overview

In the project five sensor nodes have been used and each of them is interfaced with sensors (Tmp100 and Sht25) and EEPROM. Each sensor node has its own unique device id. The gateway addresses each sensor node with its device id and the corresponding node responds to the request query. The communication between gateway and sensor node is performed with the help of Dash7 Alliance protocol. Figure 5.1 shows an overview of the communication between the gateway and sensor nodes. Online data measurements are read by the gateway from all the nodes.
It can be seen in figure 5.1, the gateway first sends the request query to Node Id 15 and then waits maximum 600 ms for the answer from Node 15. The Node 15 receives the request from the gateway and sends the current value measurements of interfaced sensors, by addressing through I2C interface, along with other appended data bytes which will be described later. The data from the interfaced sensors along with the gateway rssi value and Restart Node Status is also written to the EEPROM. The timer in Node 15 is set to one minute after sending the data to the base station, which implies that the next current measurenemt value of the sensors can be taken after one minute from Node 15. The gateway sends the next request query to Node Id 20, which responds it by sending the stored EEPROM data of 16 bytes and sets the timer to 3 sec. Node Ids 21, 10 and 43 follows the same procedure by sending 16 bytes of the saved EEPROM data upon receiving a request from the gateway. Due to radio receive time of 600 ms in the gateway, the gateway sends the next query to each node after every 3 seconds (600ms x 5) in case of five nodes. This becomes the reason to set the timers of Nodes 20, 21, 10 and 43 to three seconds which make them sleep in this duration and save the power consumption of the nodes. When the sequence of sending query request repeats at Node Id 15 as marked in the figure also, the data is not sent from the Node 15 to the gateway because of not timing out of 1 minute set in its timer before. The gateway then sends the next request to Node Id 20 and gets the response from it.
During the testing of the communication between nodes and gateway in the project, different radio receive time of the gateway were set such as 300 ms, 600 ms and 1200 ms after sending

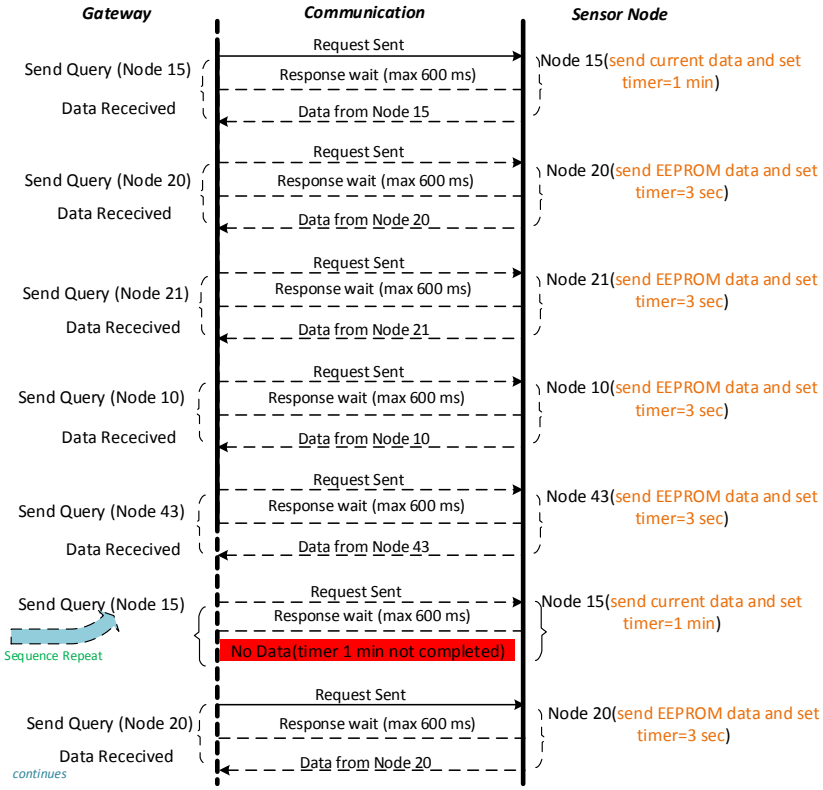


Figure 5.1: Communication Overview

the request to the nodes to check the nodes response with their increased distance from the gateway. The waiting or sleep time of node has to be corrected according to the gateway interval in case of sending EEPROM data to the gateway.

5.2 Detailed Description of the Communication

5.2.1 Gateway State Machine

The Gateway module implemented in the project acts as a State Machine as shown in figure 5.2. When the gateway is powered on, timer event function of Dash7 module (d7aoss) is programmed to 1 msec. The controller goes in low power mode for 1 msec and after 1 ms it jumps to the state 1.

5.2.1.1 State 1

The state 1 of the gateway is responsible for sending the queries to the sensor nodes, receiving the data from them and process the data. The gateway module accesses the Sensorstruct module in the software to get the device id of the sensor node for sending request. Sensorstruct

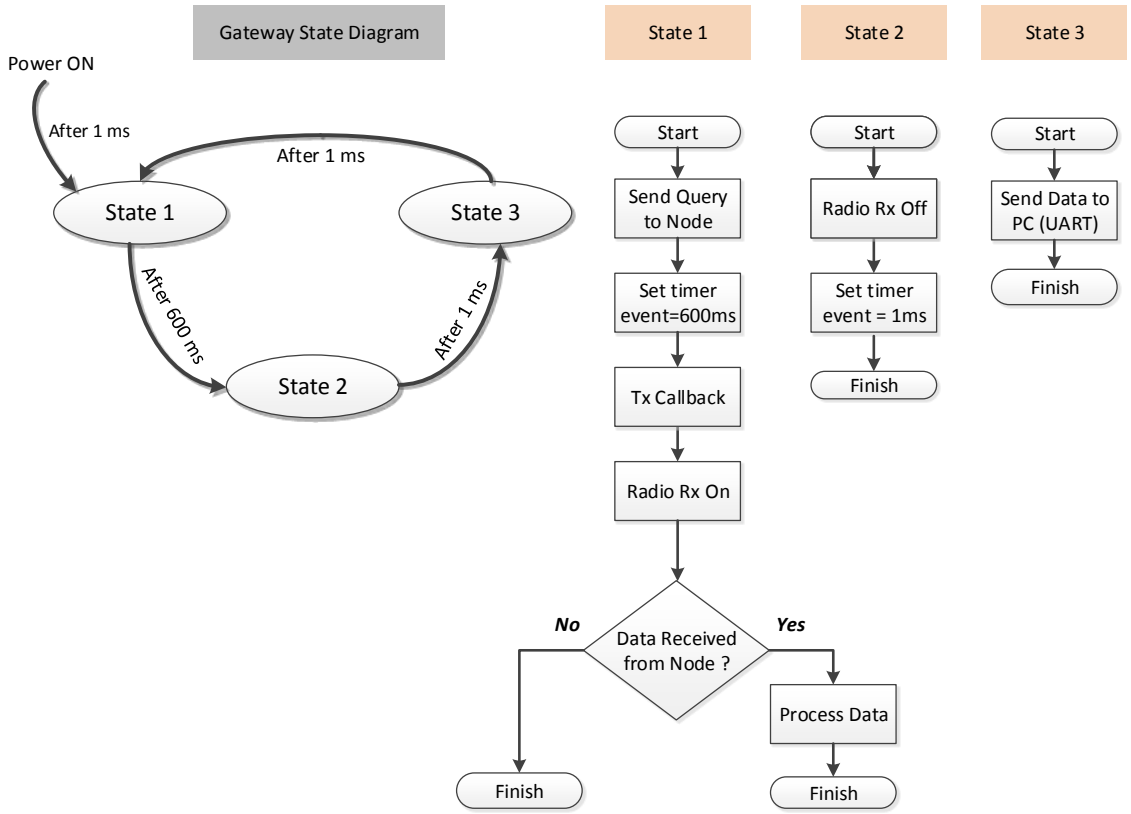


Figure 5.2: Gateway state Diagram

module has a predefined storage of all sensor nodes Ids. The gateway sends the data request to the sensor node with its device id and then the timer event is set to 600 ms . The timer event in state 1 is set by taking important considerations such as the time required for the sensor node to get the sensor values from the interfaced sensors through I2C communication, time required to write the values in the EEPROM through I2C communication and signal latency when the sensor node distance is increased from the gateway. The tx call back function in the Dash7 module informs that the request has been sent to the sensor node and brings the controller in the active mode. The radio of the gateway is turned on in receiving mode with the interrupt enabled for preparing it to receive the data from the sensor node and the controller goes in sleep mode. If the data is not received from the sensor node within 600 ms, the gateway changes its state and goes to the next state but on the other side if the data is received from the sensor node, the gateway processes the data by adding the RSSI value of sensor node and stores it in a data buffer size of 20 bytes. Sending and receiving of data through Dash7 module has been programmed to 20 bytes in the project. The gateway is able to receive the current values and the previously stored values in the EEPROM of the sensors

in real time. The structure of data blocks received from the current measurements of sensor node is shown in figure 5.3.

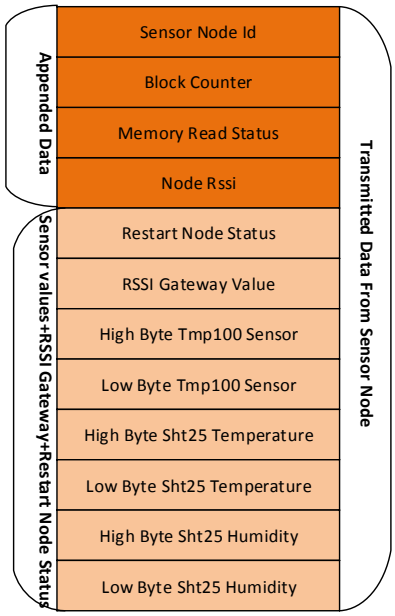


Figure 5.3: Data Received from Sensor Node

As it was discussed before, the sensor node also sends some appended data which are the the first four data bytes or blocks in the complete block. These four data bytes are described below.

Sensor Node Id: It is the device identification number or device id of the sensor node.

Block Counter: Block counter informs the gateway about the number of blocks already sent by the sensor node.

Memory Read Status: Memory read status informs that the data received from the sensor node is either current or from EEPROM.

- **0:** Current values of sensors
- **1:** Data obtained from the EEPROM

Node RSSI: The raw value is obtained in the fourth data block received from the sensor node. The gateway modifies only this data by writing the node RSSI in it.

The sensor values obtained from the sensor node in figure 5.3 is current data measurement because of its size of 12 bytes. When the gateway receives the EEPROM data of any sensor node, the total blocks received from the sensor node consists of 20 bytes (16 bytes from EEPROM+4 bytes appended data). After completion of 600 ms, the timer event is triggered which causes to change the state from 1 to 2.

5.2.1.2 State 2

In state 2, radio of gateway which is in receiving mode is turned off and timer event is set to 1 ms. After 1 ms, timer event triggers and the state changes.

5.2.1.3 State 3

In state 3, the data obtained from the sensor node is sent to a PC through UART at a baud rate of 115200. The timer event was already configured to 1 ms which stays valid in state 3 too and becomes the reason to change the state from 3 to 1 again after 1 ms. The complete module has been implemented in low power mode and the watchdog timer has also been used to reset the device in case of any defect in the system or interfaced sensors.

5.2.2 Sensor Node Operation

The sensor node working in response to gateway request can be seen from figure 5.4. After turning on the power of sensor node, the radio is turned on in receiver mode so that it can catch the query request from the gateway. The node remains in low power mode until it receives any request. If the request is not received for 4 min and 16 sec, the watchdog timer resets the device.
When the sensor node receives the gateway request, the watchdog timer is also kicked or reset. The first responsibility of sensor node is to match its own device id with the id which is sent by the gateway in query request. If the device id is not matched, sensor node is again put into the receiving mode to receive the next query request from the gateway. In case of matched id, restart node status is monitored. In case of node restart, the sensor node is programmed in such a way that it sends all sensor data written into the EEPROM to the gateway in chunks of 16 bytes. After sending every 16 bytes of data from memory, controller goes in sleep mode for 3 sec.
When the stored data is completely read from the memory, the sensor node starts to address its interfaced sensors and sends the current measurement values of the sensors to the gateway upon its request. The measurement values are also saved in the EEPROM. After sending current values of the sensors to the Gateway, the timer keeps the controller in sleep mode for 1 minute.

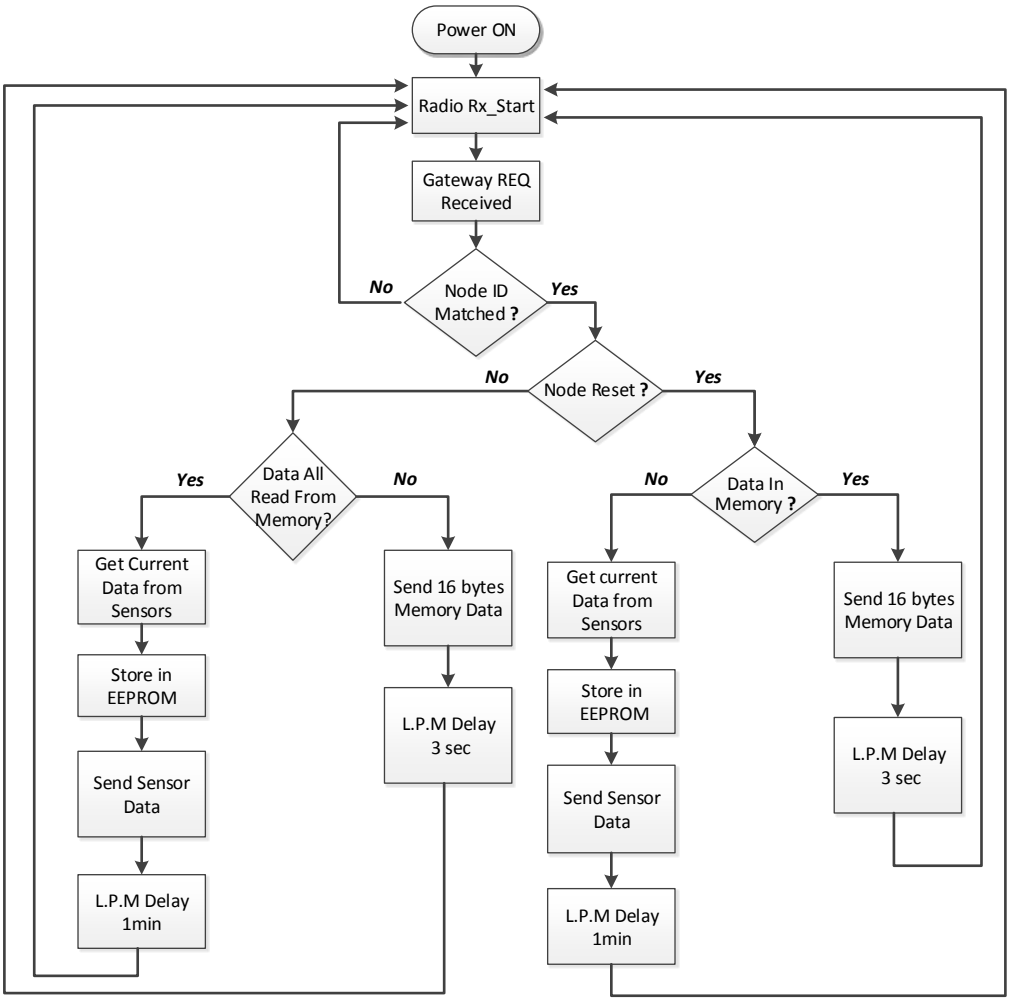


Figure 5.4: Sensor Node Operation

5.3 Software Modules

5.3.1 List of All Software Modules

Figure 5.5 shows the files which have been modified in the software and figure 5.6 shows the newly created modules in the software.

- Sensor Struct Module: It is the definition of structure to hold sensor data.
- Main Slave: It handles the gateway request and takes subsequent action for the response.
- Gateway: It is responsible for sending query to the nodes and receiving reply from the nodes as a response.
- Transmit RF data: It sends the data through Dash7 module.

	Source Files	Header Files
Sensor Struct Module	sensorStruct.c	sensorStruct.h
Main Slave	MainSlave.c	
Gateway	Gateway.c	
Transmit RF data	txRF_data.c	
Receive RF data	rxRF_data.c	

Figure 5.5: Modified Files

- Receive RF data: It receives the data through Dash7 module.

	Source Files	Header Files
Sensor Communication Module	sensor_communication.c	sensor_communication.h
SHT25 Module	sht25.c	sht25.h
TEMP100 Module	tmp100.c	tmp100.h
EEPROM Module	eeprom.c	eeprom.h
I2C Module	i2c.c	i2c.h

Figure 5.6: Newly Created Software Modules

- Sensor Communication Module: It is used as a interface to access the interfaced sensors and EEPROM.
- SHT25 Module: It is a Sht25 sensor driver for getting temperature and humidity measurements.
- TMP100 Module: It is a Tmp100 sensor driver for getting measurement.
- I2C Module: It is a I2C driver to send command over the I2C bus such as connected device address and its operating commands.

5.3.2 Software Modules for Getting Current Sensor Measurements

Figure 5.7 describes the software modules implemented to get the real time current values of the interfaced sensors (Sht25 and Tmp100) from the nodes.

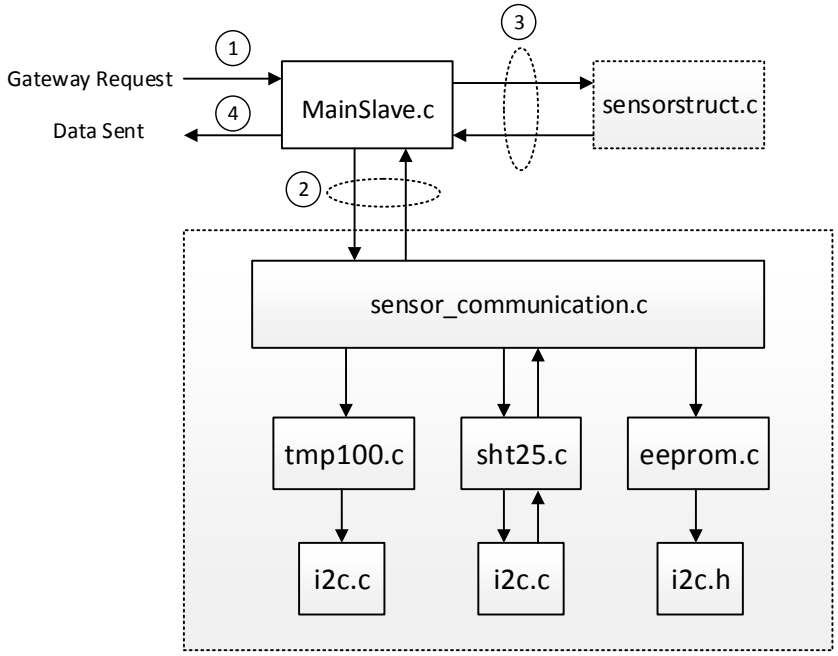


Figure 5.7: Software Modules

The numbered circled markings in the figure denotes the execution process for getting the current measurements of the interfaced sensors.

First Step: The Gateway sends the request to the Sensor Node.

Second Step: The Main Slave module receives the gateway request and accesses the Sensor communication module in case of matched device id. Sensor communication module can access all the interfaced devices such as Tmp100, Sht25 and EEPROM. Tmp100 module configures the Tmp100 sensor in 10 bit resolution and reads the temperature values with the help of I2C module while the SHT25 module reads the temperature and humidity values from the Sht25 sensor using the same I2C module. Current measurement values of the Tmp100 and Sht25 sensors are then forwarded to the Sensor communication module with a size of 6 bytes, which further adds RSSI of Gateway and Restart node status data with the current sensor measurement values, making it a total data of 8 bytes. The same 8 bytes of data is sent to the Main Slave module and also stored in a 16 byte buffer in the Sensor communication module. After one minute, when this buffer is completely filled then 16 byte

data is written in the EEPROM through EEPROM module. In the software program, data is always stored in the EEPROM with a size of 16 bytes after every two minutes. EEPROM module is used for setting the number of bytes to be written in the EEPROM or read from the EEPROM. Before writing any data in the EEPROM, EEPROM module also reads the first two locations in the EEPROM to find the next starting address to write data in it. All this process takes place by making use of I2C module.

Third Step: Sensor struct module in the software adds some more data to the sensors data such as sensor node id, block counter, memory read status and Node Rssi. The same was also described in figure5.3.

Fourth Step: Main Slave module sends all the data to the gateway.

5.3.3 Software Modules for Getting EEPROM values

Figure 5.8 describes the software modules implemented to read the saved values of the sensors in the EEPROM when the sensor node is reset.

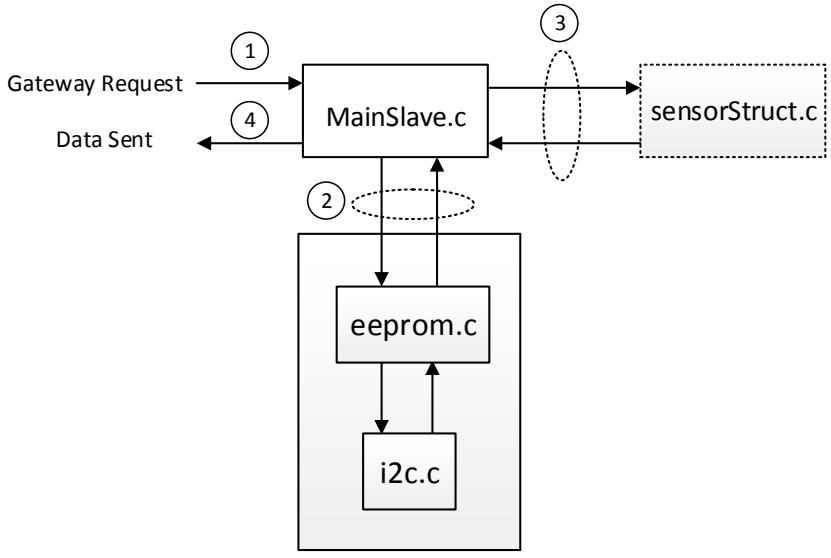


Figure 5.8: Software Modules

First Step: The gateway sends the request to the sensor node.

Second Step: If the device id is matched, the Main Slave module reads the first two cells in the memory. The EEPROM is considered to be empty when the values on both cells make

0xFFFF (raw values). On the contrary if the combined value of both locations is other than 0xFFFF, gives an indication of some values already written in the EEPROM before. The values are read from the EEPROM in a block size of 16 bytes.

Third Step: Sensor struct module appends four more bytes to the 16 byte received from the EEPROM.

Fourth Step: Main Slave module sends the data to the Gateway. This cycle will continue until the values from the EEPROM are completely read.

5.4 Offline Mode

Offline mode has also been developed also for Sensor nodes. In the offline mode, sensor nodes take reading from the interfaced sensors after 1 minute without any request from the gateway. Data is saved in the EEPROM also after every two minutes of readings. This mode does not make use of the Dash7 module. Low power mode 3 of the CC430F5137 has been programmed in this mode. The saved EEPROM readings of sensor nodes in the offline mode can be read later by the gateway after programming each node again with the Dash7 module. The structure of the values saved in the EEPROM is shown in figure 5.9.

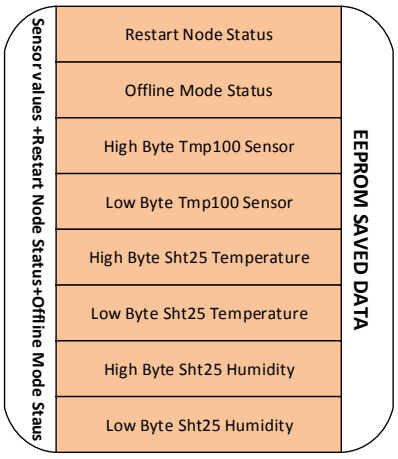


Figure 5.9: Offline Mode Data in EEPROM

The data structure stored in the EEPROM remains same as described in figure5.3 except the second byte. The second byte of EEPROM data in the offline mode is the Offline Mode Status which gives an indication that the data stored in the EEPROM belongs to the offline mode. A constant number of 88 has been assigned for this status byte in the software for the

easy identification in the log text file generated, when the gateway accesses the node in real time.

5.5 Java Software Modification

Java software was used in PC to get the serial data from the UART of microcontroller and to save the data in a text file after converting the raw values of the sensors in readable form. Microcontroller sends data on UART with a length of 20 bytes. Some modifications were needed in the class named as SensorData, which is shown in figure 5.10.

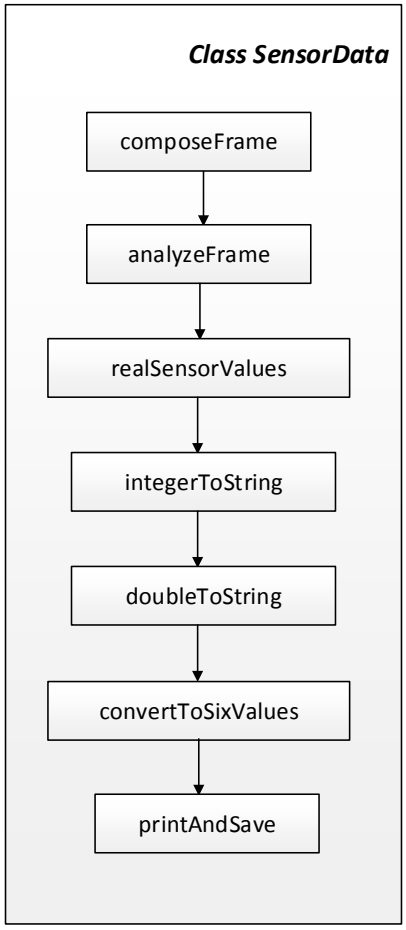


Figure 5.10: Java Class Modification

composeFrame: It is a function which saves the 20 bytes data received in the buffer of InputStream class in a byte array of 20 bytes.

analyseFrame: This function is used to sort the 20 bytes data according to its type, such as merging the two bytes data of Tmp100 temperature in an Integer data type.

realSensorValues: This function converts the raw values of the Tmp100 and Sht25 sensors into real values by the specified formula mentioned in the sensors' data sheet.

integerToString: This function converts all individual integers data to string so that the data can be stored in a text file.

doubleToString: It is used to convert the double values of Tmp100 and Sht25 into string for saving it in a text value.

convertToSixValue: This function limits the values of Tmp100 and Sht25 sensors to maximum six digit.
printandSave: Printing all the values on console and saving them in a log text file is possible by this function.

5.5.1 Log File Data Presentation

The pattern of the data saved in the log text file generated in a Java program is shown in figure 5.11.

1	2	3	4	5	6	7	8	9
20	14	0	-64	0	-65	3.0	3.3220	49.931
21	15	0	-74	0	-75	2.75	2.8286	51.838
15	17	0	-71	0	-74	2.5	3.0324	33.520
43	15	0	-75	0	-76	2.75	3.0002	52.044
10	15	0	-75	0	-76	3.0	2.9037	78.121

Figure 5.11: Log File Data

- 1: Device Id
- 2: Block Counter

- 3: Memory Read Status ("0"= Real Data Measurement and "1"= Data from EEPROM)
- 4: Node RSSI (RSSI value received at the Gateway from the Node)
- 5: Restart Node Status ("0"= Node not reset and "1"= Node reset)
- 6: Gateway RSSI (RSSI value received at the Node from the Gateway)
- 7: Tmp100 Temperaure (in °C)
- 8: Sht25 Temperaure (in °C)
- 9: Sht25 Humidity (% relative humidity)

5.6 IDE for the Software

Code Composer Studio IDE with version number 6.0.1 has been used in the project for the software to develop and debug. This IDE is provided by the Texas insrtument and it supports all Texas Instrument microcontrollers. This software can be downloaded free of cost from website http://processors.wiki.ti.com/index.php/Download_ CCS

5.7 Code at SVN

All the code developed and modified in the project is available on the SVN repositiry. The destination paths of the software modules implemented in the Code Composer Studio and Eclipse are mentioned below.

https://icia.de/svn/students/faisal/RealDataSoftware
https://icia.de/svn/students/faisal/Offlinemode
https://icia.de/svn/students/faisal/java

6 Testing and Measurements

This chapter deals with the testing and measurements of the sensor nodes with the gateway and the analysis of their obtained results. Testings were carried out in the building as well as outside in an open area field.

6.1 Testing in Building

The first test was carried out in the building by placing all five sensor nodes and the gateway in the rooms. The ground floor lab rooms of IMSAS department (University of Bremen) were used for this testing. Figure 6.1 shows the sensor nodes and gateway placement in different rooms. All the four rooms are in one row. The thickness of the room wall connecting to other room is approximately 0.16 m.

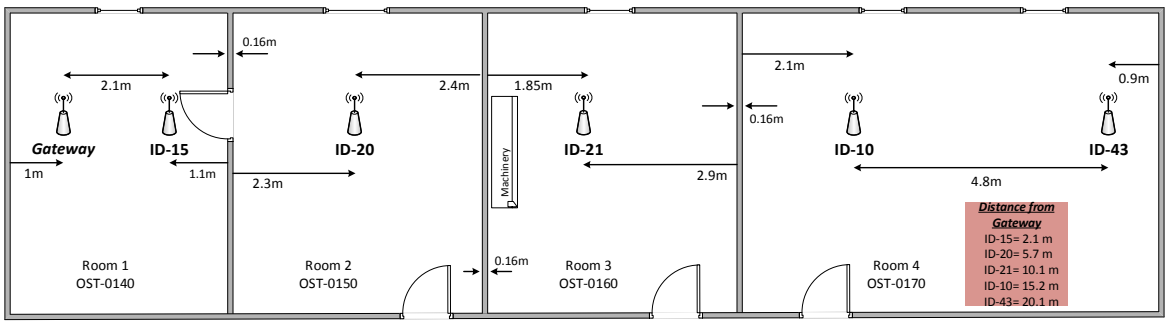


Figure 6.1: Testing in Building

Room 1: It is named as OST-0140, gateway and Node Id 15 were placed in this room. Node Id 15 has a distance of 2.1 m from the base station.

Room2: The room 2 is OST-0150 and Node Id 20 was positioned in it at a distance of 5.7 m from the gateway.

Room3: The room 3 is OST-0160 and Node Id 21 was placed behind a heavy metal machine in this room at a distance of 10.1 m from the gateway.

Room4: This room is OST-0170 and two sensor nodes having Ids 10 and 43 were positioned in the room at a distance of 15.2 m and 20 m respectively from the gateway.

6.1.1 Test One

In the first test of seven minutes, the radio of the gateway was kept in receiving mode for 300 ms after sending the request to each sensor node, which implies that the gateway has only 300 ms to get the required data from the requested sensor node. Five sensor nodes used in the testing so each node gets the next request from the gateway after every 1.5 seconds (5x300ms). The values obtained per sensor node is shown in figure 6.2.

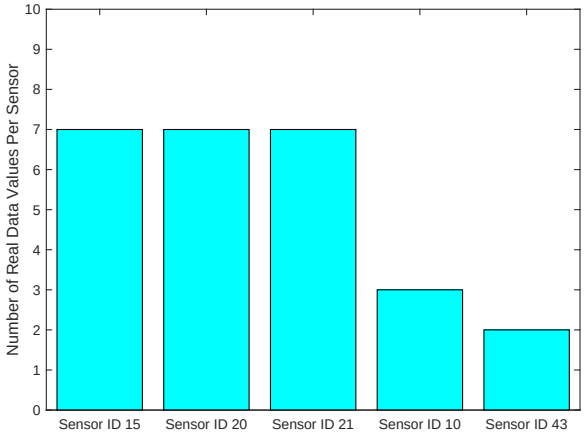


Figure 6.2: Number of Packets Received (Test 1: Setting One)

As shown in figure 6.2, two packets were received from the sensor Node Id 43 and three from Id 10 while the complete seven packets were received from other nodes. Sensor Node with Id 10 was positioned at a metal surface which is also a factor of receiving less packets from this node. The gateway has a very short time to receive the values from the sensor nodes because after addressing each Sensor node it switches to receive mode for a duration of 300 ms to get the response from node, while the node has to take measurements from its interfaced sensor, store the measurement in the EEPROM and respond the gateway by sending the measurements within 300 ms.
Due to less receiving packets at the gateway, Node Id 43 was relocated in the same room at a distance of 17.2 m from the gateway and the metal surface was removed from the base of the Node Id 10. Readings were observed for 21 minutes and the values received per sensor node is shown in figure 6.3.

Significant improvement is visible from figure 6.3 regarding with the packets received at the gateway terminal. Only 2 packets were lost from Node Id 10 and one from Node Id 21.

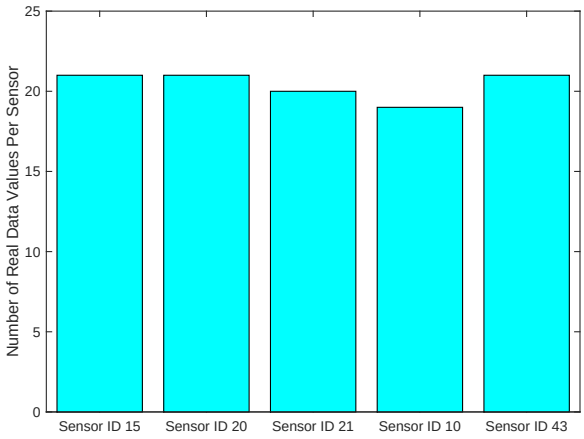
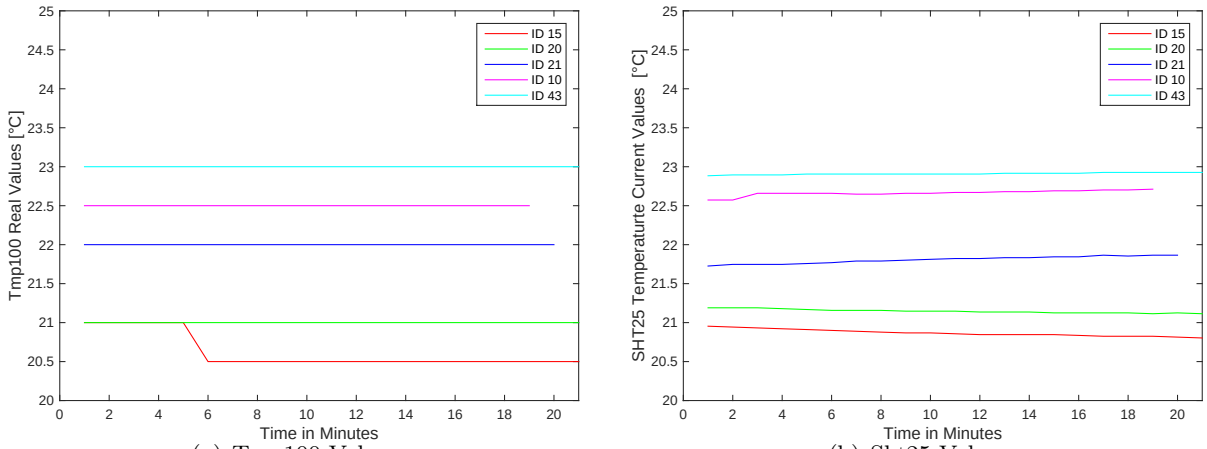


Figure 6.3: Number of Packets Received (Test 1: Setting Two)

6.1.1.1 Temperature values from Tmp100 and Sht25

The temperature values of Tmp100 inside the housing and outside temperature values recorded from Sht25 of each sensor node is shown in figure 6.4.



(a) Tmp100 Values (b) Sht25 Values

Figure 6.4: Temperature Values (Test 1: Setting Two)

The plot depicts slightly different temperature readings, inside the housing and outside the housing, in each room. Tmp100 values inside the housing of each Sensor node remain constant because the resolution of this sensor was set to 0.5°C. On the other side values from the Sht25 sensors, with resolution of 0.01°C, show very smooth changes of the outside temperature.

6.1.1.2 Humidity Values

Humidity readings obtained from each sensor node can be seen in figure 6.5. Node Id 15, in room 1, has the highest humidity values among other sensor nodes.

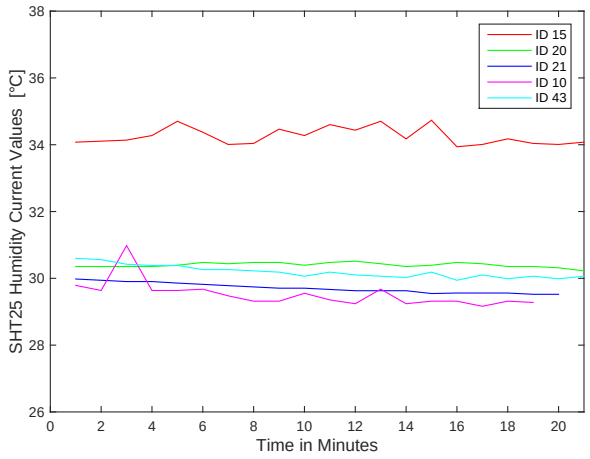


Figure 6.5: Humidity Values (Test 1: Setting Two)

6.1.1.3 RSSI values

The RSSI value of each node with respect to their distances from the gateway is displayed in figure 6.6.

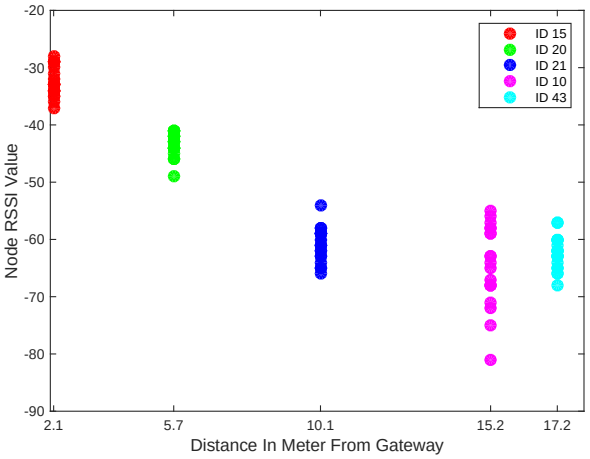


Figure 6.6: Node RSSI Values (Test 1: Setting Two)

The pattern of RSSI values illustrate that the increasing distance of sensor nodes from gateway cause the value of RSSI to decrease. From the above figure, it can be seen that the lower RSSI values were received from Node Ids 21, 10 and 43. Node Id 10 has more decreasing value of RSSI than Node Id 43, although Node Id 43 has the largest distance of 17.2 m from Gateway in comparison to Id 10.

6.1.1.4 Correspondence Between Gateway and Node RSSI

For every node measurements, the node and gateway RSSI values are nearly symmetrical. Figure 6.7 shows the similarity of the Gateway and Node RSSI values.

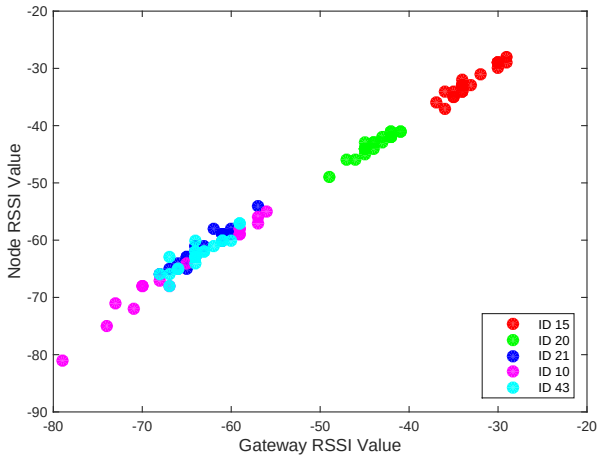


Figure 6.7: Correspondence between Gateway and Node RSSI (Test 1: Setting Two)

6.1.1.5 Mean and Standard Deviation Of RSSI

The mean and standard deviation of RSSI values of each node can be seen in figure 6.8. Node Id 10 has the largest deviation of 7.04 from its mean. There can be several factors for getting such a behaviour from Node Id 10 such as its placement and more reflections of signal in the lab room.

Node Id	Mean	Deviation
15	-32.43	2.58
20	-43.33	2.01
21	-60.90	2.94
10	-64.58	7.04
43	-62.33	2.82

Figure 6.8: Mean and Standard Deviation (Test 1: Setting Two)

6.1.2 Test Two

In this test, some modifications were made such as Tmp100 resolution was changed to 0.25°C in every sensor node, Node Id 43 again placed at a location of 20.1 m from the gateway and the radio of the gateway set to 600 ms in receive mode after sending the request to the Node. The receive time of the gateway was increased due to increased resolution of Tmp100 and to check Node 43 response at a distance of 20.1 m from the gateway. Due to this increased time, each sensor node gets the query request from the gateway after every 3 seconds (5x600ms). The testing was performed for 20 minutes as shown in figure 6.9.

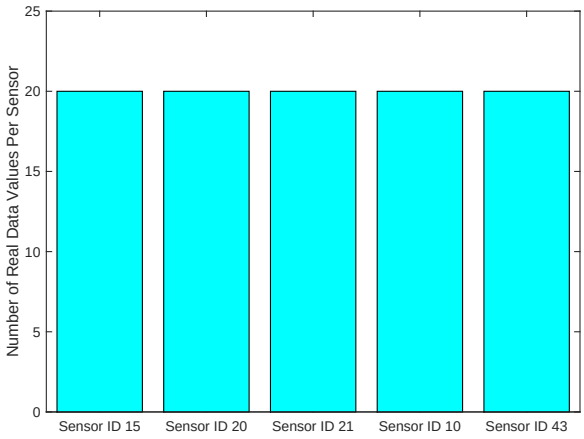
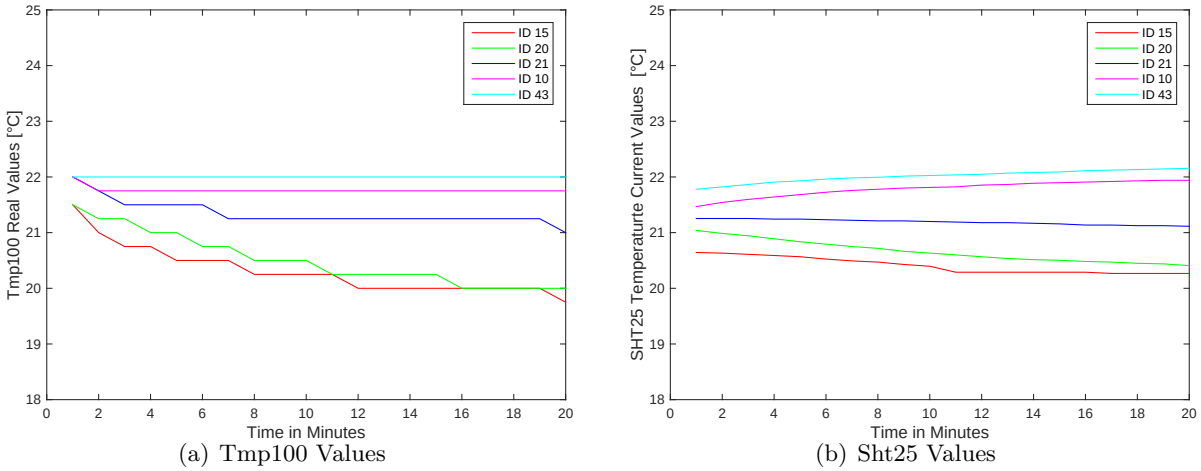


Figure 6.9: Number of Packets Received (Test 2)

In this mode, no packets were lost from any Sensor Node and the Gateway was able to receive the packets from Node Id 43 too at a distance of 20.1 m from the Gateway due to increased radio receive time. Stored data from the EEPROM was also read in this test without any loss of packet.

6.1.2.1 Temperature Values from Tmp100 and Sht25

Figure 6.10 displays the temperature chart of Tmp100 and Sht25 sensors of sensor nodes.



(a) Tmp100 Values

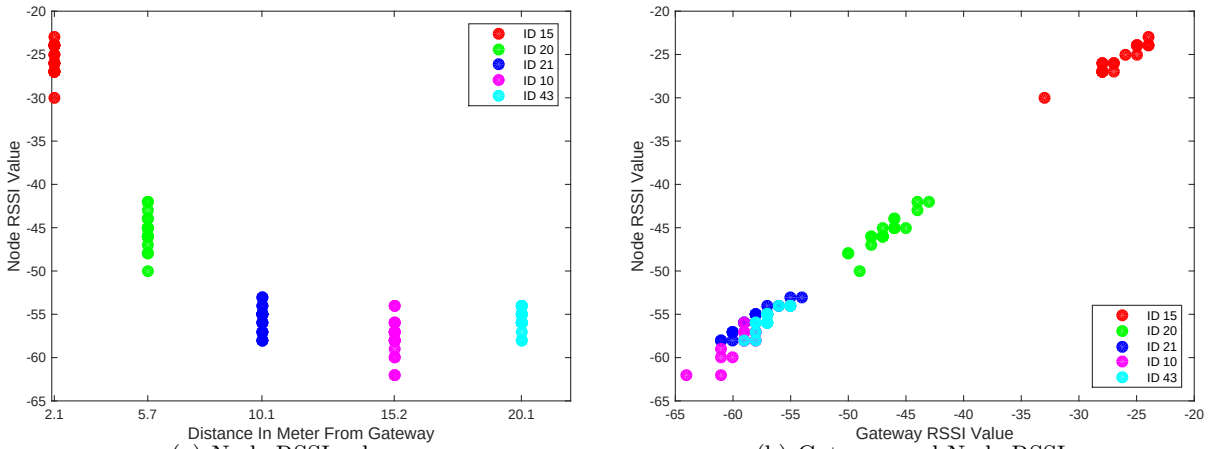
(b) Sht25 Values

Figure 6.10: Temperature Values (Test 2)

The increased resolution of Tmp100 can be clearly seen from the figure 6.10(a). Small variations of values are visible from the temperature graph of inside and outside the housing.

6.1.2.2 RSSI values

The RSSI values of the nodes with their respective distances and its correspondence with the gateway is shown in figure 6.11



(a) Node RSSI values

(b) Gateway and Node RSSI

Figure 6.11: RSSI Values (Test 2)

In this test also, majority of RSSI values of Node Id 21, 10 and 43 lie in the same region and Node Id 10 has the lowest RSSI value. The symmetrical values of the gateway and nodes RSSI were also found in this test.

6.1.2.2.1 Mean and Standard Deviation Of RSSI

The mean and standard deviation of RSSI values can be seen in figure 6.12

Node Id	Mean	Deviation
15	-25.80	1.67
20	-45.45	1.96
21	-55.60	1.54
10	-57.70	2.15
43	-55.60	1.19

Figure 6.12: Mean and Standrad Deviation (Test 2)

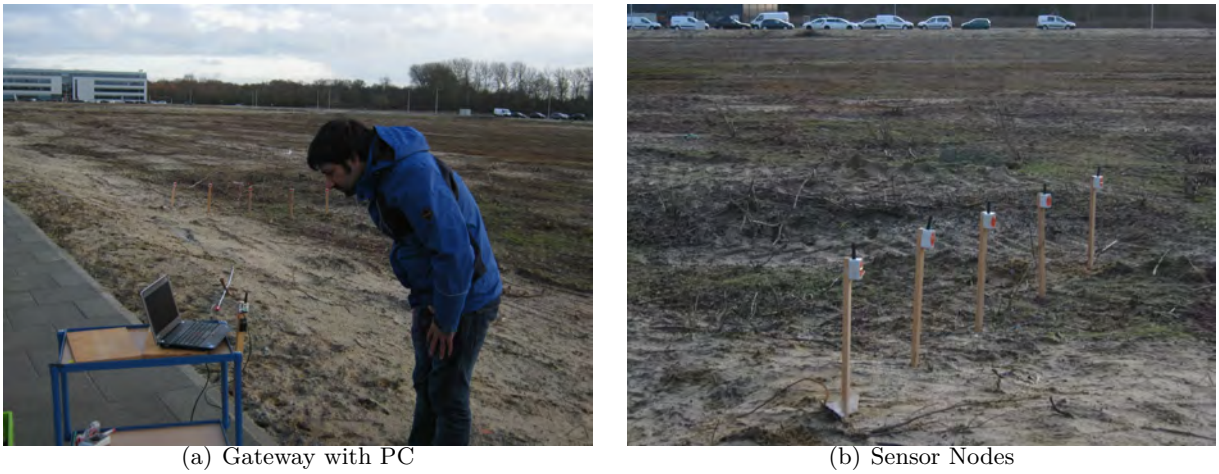
Better RSSI values are obtained in this test as compared to test 2. The overall trend of RSSI is decreasing with the distance except the Node 10 which has the highest RSSI mean value and standard deviation.

6.2 Testing In an Open Field

In order to examine RSSI values and the range of sensor nodes, testing was also performed in an open field to get a larger area for the test. All sensor nodes and gateway were mounted 50 cm above on wooden sticks to avoid possible interface from the ground.

6.2.1 First Test

Setting One: In this test, gateway was positioned at a fixed point in an open field and all sensor nodes were positioned at a distance of 10 m from the Gateway as shown in figure 6.13



(a) Gateway with PC (b) Sensor Nodes

Figure 6.13: Complete Setup

Figure6.14 shows the number of packets received in a testing of 13 minutes. No dropping of packets from any Sensor node is clearly visible from the figure.

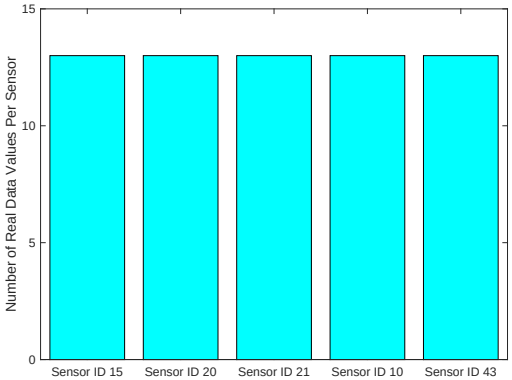


Figure 6.14: Number of Packets Received (Outside Test 1: Setting One)

6.2.1.1 Mean and Standard Deviation Of RSSI

There were no major differences of the RSSI values obtained from each node at a distance of 10 m from the gateway. The mean and standard deviation is displayed in figure 6.15

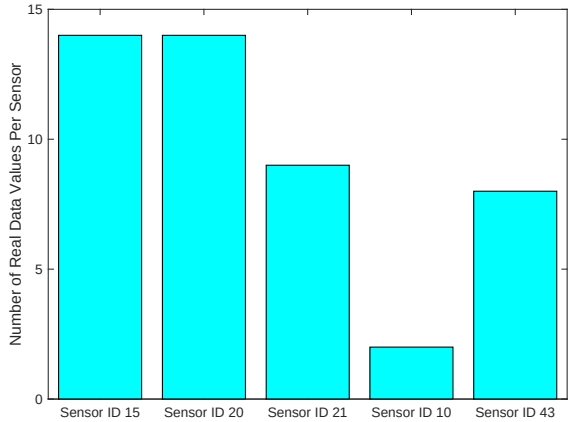
Node Id	Mean	Deviation
15	-57.85	2.41
20	-57.54	1.27
21	-55.85	1.62
10	-55.38	1.76
43	-55.31	1.44

Figure 6.15: Mean and Standard Deviation (Outside Test 1: Setting One)

Setting Two: The purpose of this test is to relocate each sensor node away from the base station by every 10 meter steps in a row to check its range. This test was conducted for 14 minutes. The sensor nodes positioning with respect to the gateway and their readings received is shown in figure 6.16.

Node Id	Distance from Gateway
15	10 meter
20	20 meter
21	30 meter
10	40 meter
43	50 meter

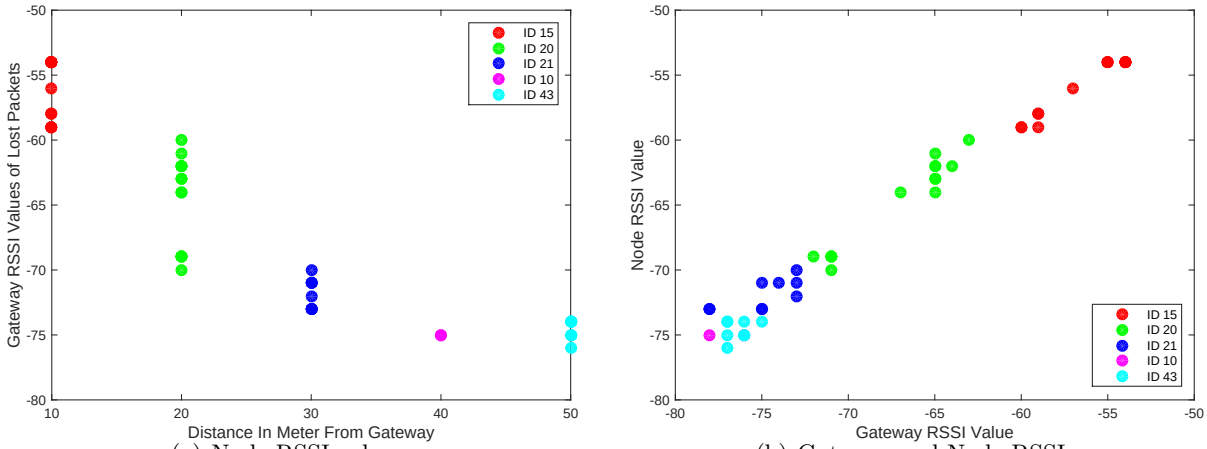
(a) Sensor Node Position



(b) Number of Packets Received

Figure 6.16: Sensor Positions and Received Readings (Outside Test 1: Setting Two)

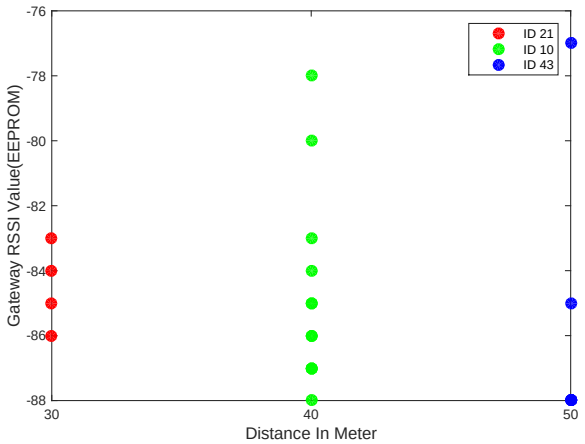
Packets were lost from the Node Ids 21, 10 and 43. Only 2, 8 and 9 packets were received from Node Id 10, 21 and 43. The behaviour of their RSSI values is shown in figure 6.17. Plot depicts the decrease in RSSI value with their increasing distance from the gateway. The lowest node RSSI value received is -75 dB of Id 10 whereas the least gateway RSSI value received at Node Id 10 is -78 dB. There can be several factors for receiving less packets from other nodes such as the disturbance and noise from the environment. Signal latency can also be a factor for receiving less packets.



(a) Node RSSI values (b) Gateway and Node RSSI

Figure 6.17: RSSI Values (Outside Test 1: Setting Two)

To get the better understanding about the loss of packets from the nodes. EEPROM data was read from the Node Ids 10, 21 and 43 after the test. It was found out that the total readings of 14 minutes were saved in the EEPROM of all sensor nodes. The Gateway RSSI values stored in the EEPROM of the lost packets from the sensor Nodes 10,21 and 43 and their mean values are displayed in figure 6.18. It can be seen that the gateway RSSI values received at the nodes were in the range of -78 to -88 dB verified from the EEPROM stored values of the lost packets.



(a) Gateway RSSI of lost packets

Node Id	Distances	Mean	Deviation
21	30 m	-84.50	1.29
10	40 m	-84.67	3.02
43	50 m	-85.67	4.41

(b) Mean and Standrad Deviation

Figure 6.18: Gateway RSSI Values Of Lost Packets (Outside Test 1: Setting Two)

Setting Three: To examine the RSSI values further more, positions of Node Ids 15 and 20 were changed while the position of all other nodes kept same in ten minutes reading as shown in figure 6.19.

Node Id	Distance from Gateway
15	60 meter
20	70 meter
21	30 meter
10	40 meter
43	50 meter

Figure 6.19: Sensor Node Positions (Outside Test 1: Setting Three)

No packets were received from Node Id 20 at 70 meter but all real time measurements observed from Node 15 even at 60 m. The reading pattern from each node is shown in figure6.20.

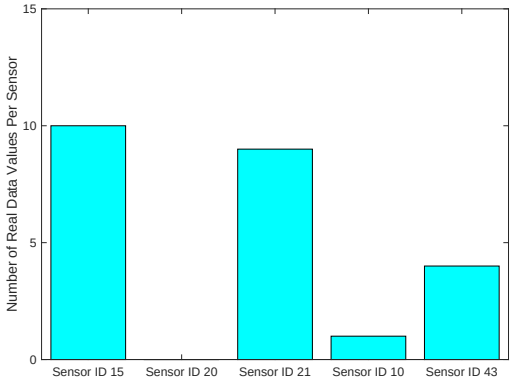


Figure 6.20: Number of Packets Received (Outside Test 1: Setting Three)

The readings were strange according to their distances from the Gateway. Figure 6.21 illustrates the mean RSSI values of nodes. For the lost packets of all nodes, EEPROM values were again read and most of their RSSI values were found in the same region (-78 to -88 dB. The lowest Gateway RSSI value was found in Node Id 10 with a value of -91 dB. From the stored readings in the EEPROM of all Sensor Nodes, it was confirmed that each Sensor node was able to receive the Gateway request irrespective of their distance from it.
Such a behaviour of the RSSI values and loss of packets could be due to noise or disturbance from the environment which can cause the signal latency at gateway, different transmission path can also lead to different RSSI values. Near the test field area, building construction site was also found and short range devices such as hand held radio for voice communication operating at 433 Mhz are also used in such places. Remote keys for cars often also use the same frequency band in Europe.

Node Id	Distances	Mean	Deviation
15	60 m	-72.20	1.13
21	30 m	-73.00	1.66
43	50 m	-74.50	0.58

Figure 6.21: Mean and Standard Deviation (Outside Test 1: Setting Three)

6.2.2 Second Test

In order to improve the reading range of real time measurement from the sensor nodes, some modification was necessary in the software. In this test, the radio of the gateway was set in receiving mode for a duration of 1.2 seconds after sending the request to get the response from the node without any loss of packet if they are positioned at a large distance from the gateway. The test was performed in snowy weather conditions. Sensor Node Id 20 was excluded from this experiment due to its soldering connection break with the Sht25 board inside the housing.

Setting One: The test was started with the same sequencce by positioning all nodes at a distance of 10 m from the gateway and measurement values were recorded for 10 minutes. No loss of packet was observed from any node.

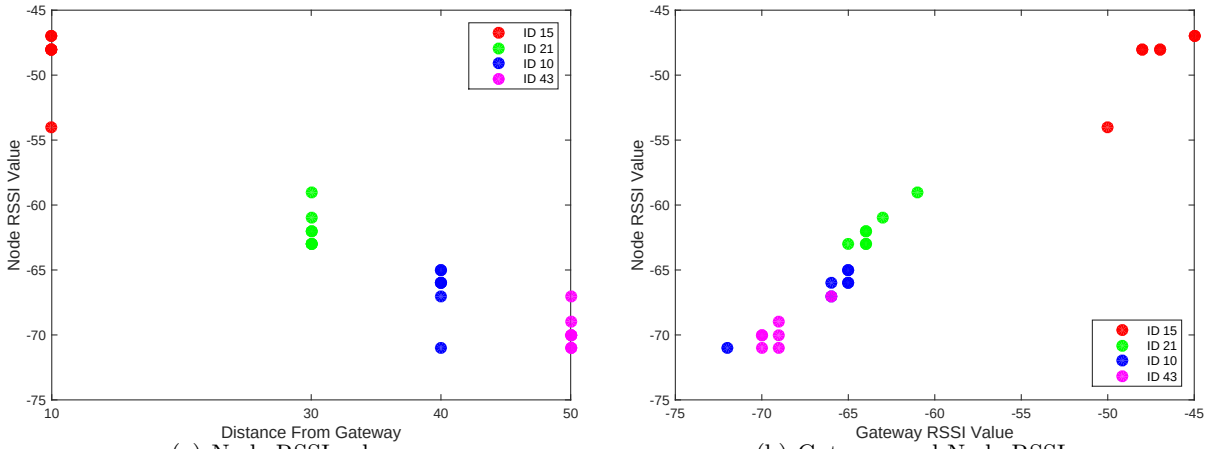
Setting Two: The second arrangement of sensor nodes is shown in figure 6.22. The experiment was carried out for 7 minutes and no packets were lost.

Node Id	Distance from Gateway
15	10 meter
21	30 meter
10	40 meter
43	50 meter

Figure 6.22: Sensor Nodes Placements (Outside Test 2: Setting Two)

Figure 6.23 depicts the Node RSSI values and its symmetrical values with the Gateway RSSI. RSSI values of nodes show a very good decreasing behaviour with their increasing distance from the gateway. No strange readings were observed in this set up of sensors placement. The lowest node RSSI value received at the gateway is -71 dB of Node Id 10. The range of 50 m has been covered in this test without any packet loss at the gateway end.

Setting Three: The sensor nodes arrangement and their readings observed for 14 minutes can be seen in figure 6.24. No packets received from the Node Id 43, mounted at a distance



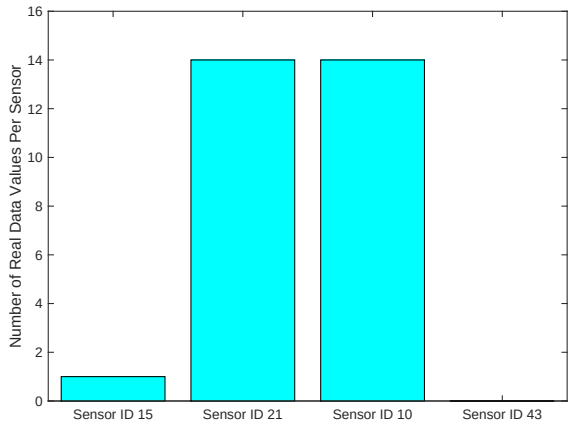
(a) Node RSSI values (b) Gateway and Node RSSI

Figure 6.23: RSSI Values (Outside Test 2: Setting Two)

of 80 m. Whereas only one packet was received from Node Id 15 which was positioned at distance of 70 m. Complete packets of Node Ids 10 and 21 were observed at the gateway.

Node Id	Distance from Gateway
15	70 meter
21	60 meter
10	20 meter
43	80 meter

(a) Sensor Node Position



(b) Number of Packets Received

Figure 6.24: Sensor Positions and Received Readings (Outside Test 2: Setting Three)

Maximum reading range of 60 m was achieved from Node Id 21 in the setting three of the Nodes placement. It was necessary to analyse the gateway RSSI values received at the Node Ids 15 and 43 for the lost packets. In order to accomplish this, EEPROM stored readings were read from both Sensor Nodes. It was found out that both the nodes received the request query from the gateway for the lost packets. The mean value of the gateway RSSI values stored in the EEPROM for the lost packets of both nodes are shown in figure 6.25.
The Gateway RSSI mean values of the lost packets shown in figure 6.25 are higher as compared to the mean values of the lost packets found in figure 6.18, even the nodes were much more closer to the gateway in that configuration. This shows some noise or disturbance effect from

Node Id	Distances	Mean	Deviation
15	70 m	-76.50	1.41
43	80 m	-79.33	2.27

Figure 6.25: Mean and Standard Deviation (Outside Test 2: Setting Three)

the environment which can lead to unrealistic behaviour of RSSI values and can also lead to the loss of packets at the gateway end.

6.2.3 Third test

In the third test, the radio communication data rate was changed from 55.5 kBaud to 27.8 kBaud to see the behaviour of RSSI values by placing the Nodes at different location. Smaller baud values account to increase the radio range but noise channel is also affected on it due to its lower value. The Radio frequency analyser, Spectran HF-4040 V3, used in this experiment to detect the devices operating in the frequency band of 433 Mhz other than sensor nodes. This analyser HF-4040 V3 can be seen in figure 6.26.

Figure 6.26: Spectrum Analyser[21]

The analyser can detect the frequency from 100 MHz to 4 GHz and its signal strength measuring range is -90 dBm to 0 dBm[21].

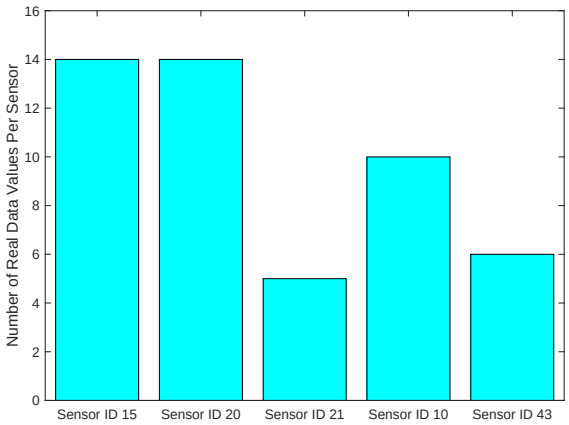
Setting One: Test started in the same sequence by placing all of them at a distance of 10 m except Node Id 20, which was positioned at 2 m. Node Id 20 was again functional after soldering the connection with the Sht25 board in its housing. Previous EERPOM data was erased from Node Id 20 due to raw values stored from the Second Test. The Rf data rate

of this Node was set to 13 Kb. Test was executed for 8 minutes and all the packets were received at the Gateway without loss.During the 8 minutes test, Rf analyser was also turned on and it was continuously detecting noise with a peak level between -50 dBm to -60 dBm. This signal from Rf analyser was continuously observable even at a far distance from the Gateway. The signal strength of the Gateway was also measured at a distance of 1 meter from the Rf analyser and the value found to be -50 dB.

Setting Two: Sensors placement in this experiment and their 14 minutes of recorded reading received at the gateway can be seen in figure 6.27.

Node Id	Distance from Gateway
15	20 meter
20	30 meter
21	40 meter
10	50 meter
43	60 meter

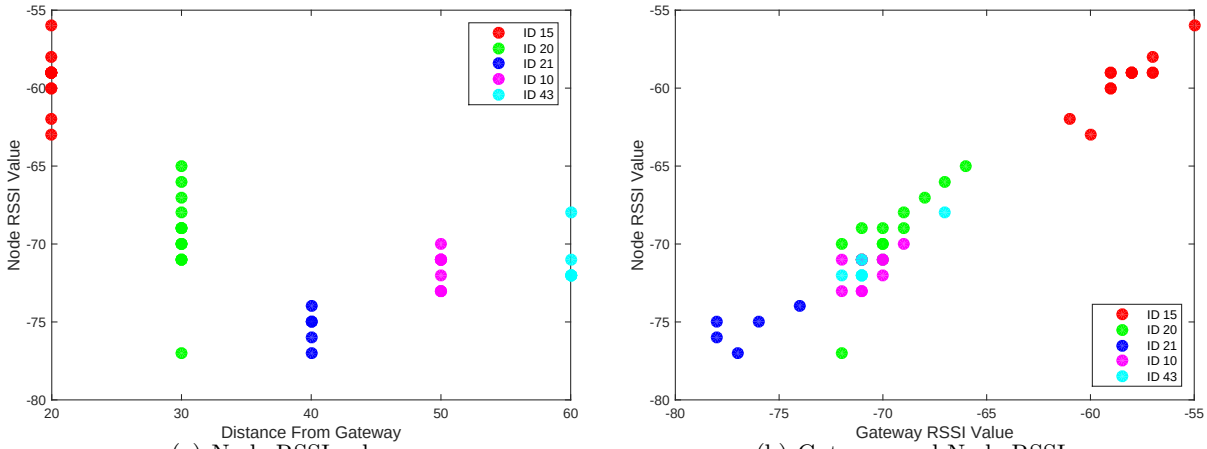
(a) Sensor Node Position



(b) Number of Packets Received

Figure 6.27: Sensor Positions and Received Readings (Outside Test 3: Setting Two)

In this configuration of the nodes placement, the peak noise measured from the Rf analyser was -45 dBm. Only 5 and 6 packets were received from Node Id 21 and 43 respectively. The region of lowest RSSI values belongs to the Node Id 21 as shown in figure 6.28, despite its more closeness from the Gateway in comparison to Node Ids 10 and 43. RSSI values of Node and Gateway are found symmetrical from the figure. Several Node placements were performed in this test with the increasing distance from the Gateway but noise in the environment was completely present, detected by the Rf analyser.

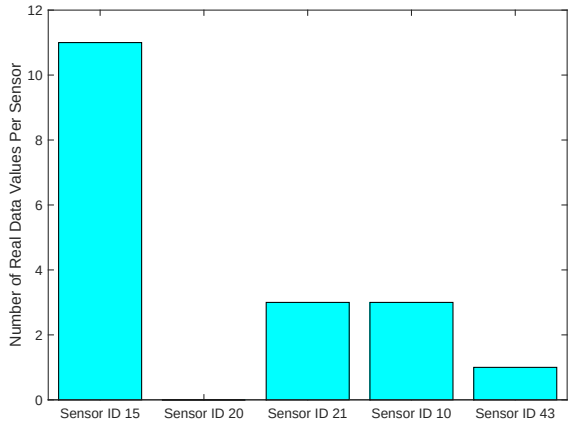


(a) Node RSSI values (b) Gateway and Node RSSI

Figure 6.28: RSSI Values (Outside Test 3: Setting Two)

Setting Three: To examine the long communication range, Node Id 43 was repositioned at a 100 m distance from the gateway. This testing was performed for 11 minutes. All the nodes positioning and the packets received at the gateway from the nodes can be seen in figure 6.29.

Node Id	Distance from Gateway
15	10 meter
20	60 meter
21	80 meter
10	90 meter
43	100 meter



(a) Sensor Node Position (b) Number of Packets Received

Figure 6.29: Sensor Positions and Received Readings (Outside Test 3: Setting Three)

All the packets were received at the gateway end from Node Id 15 because of its very close distance to gateway while no packets observed from Node Id 20, placed at 60 meter. Only one packet was received from Node Id 43 even at a distance of 100 meter. Three packets were received from Node Ids 21 and 10. Interfaced EEPROM was again read to verify the nodes data in it, which showed that all the nodes received request query from the gateway. To examine this sort of behaviour from the nodes, the measured RSSI values of nodes and gateway for the received packets and the saved gateway RSSI values in the EEPROM of the Nodes were compared with the model of free space path loss.

6.2.3.1 Free Space Path Loss

Free space path loss model is used to determine the loss in signal strength of radio wave in a line of sight communication. Free space path loss model is not valid in case of effects from the ground or any obstacle in the communication path. It is used to estimate the path loss in low range communication[22]. Free space path loss can be approximated by the below mentioned formula[6].

$$FSPL(dB) = P_0 - 10 \cdot \log_{10}\left(\frac{4 \cdot \pi \cdot d \cdot f}{c}\right)^2 \tag{6.1}$$

Where:
FSPL is the free space path loss in dB
$\boldsymbol{P_0}$ is a constant number such as antenna gain, antenna output power and receive power
d is the distance between transmitter and receiver
f is the operating frequency of transmitter and receiver
c is the speed of light

For the setting three of the third test, the free space path loss model was compared with the Node and Gateway RSSI values of real and saved measurements in the EEPROM.

6.2.3.1.1 Comparing with Node and Gateway RSSI values

Figure 6.30 depicts the free space path loss model, compared with the mean RSSI values of nodes and gateway of the received packets.

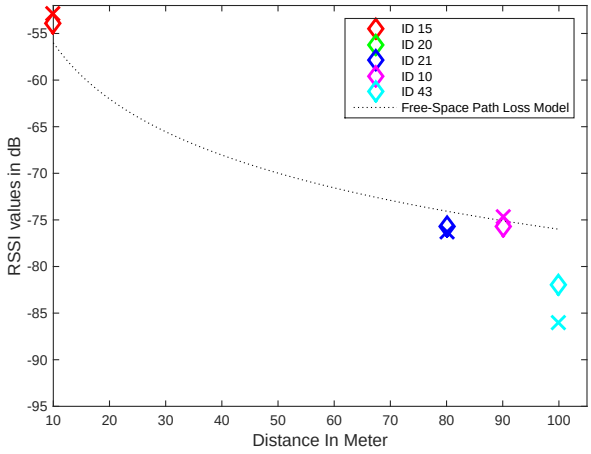


Figure 6.30: Free Space Path Loss Model Comparison

The "diamond" sign shows the mean value of all Node RSSI values and the "cross" sign shows the gateway RSSI values in figure 6.30. The values of node and gateway signal strength of Node Id 43, at 100 meter, are deviating from the path loss model. Values of Node Id 10

completely matches with the model. Nodes Id 15 and 21 are in very close proximity to the model. No RSSI value of Node Id 20 is shown in the plot due to not receiving of any packet at the gateway.

6.2.3.1.2 Comparing with saved Gateway RSSI values in EEPROM

In order to analyse the stored signal strength values of gateway in nodes EEPROM, it was also compared with the path loss model as shown in figure 6.31.

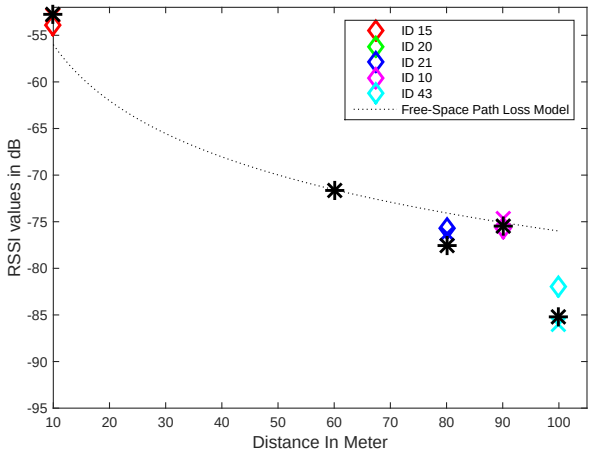


Figure 6.31: Comparison with saved Gateway RSSI in EEPROM

The black sign marking in the plot shows the EEPROM mean Gateway RSSI values received at the nodes. The plot clearly verifies that the Node Id 20 also received the Gateway request in the testing and its saved Gateway signal strength in the EEPROM completely matches with the path loss model. In this case also, Node Id 43 kept its same response for deviating from the path loss model.
There can be several reasons for the deviation from the path loss model such as ground noise due to sensors positioning just at 50 cm above from the ground level, uneven surface of the ground and also the noise measured from the Rf analyzer.

6.3 Current Measurements

Low current energy consumption is an important factor in making wireless sensor nodes. The most of the current is consumed by the radio chip CC1101 core inside the CC430F5137 microcontroller. At 433 MHz with the power output of antenna 10 dB, the radio current in receive mode is 17 mA while the radio current in transmit mode is 29 mA[12]. Interfaced sensors such as Sht25 and Tmp100 draw significant low current with their maximum consumption of 330 μA[17] and 70 μA[16] respectively. The maximum write current in EEPROM is 3 mA and maximum read current is 1 mA[18]. The CC430F5137 microcontroller consume 160

μA/MHz[10]. The current of Node was measured by the voltage drop over a 2.5 Ω resistor after connecting it with the battery. It was found out that the current consumption in sleep mode is 200 μA and it draws 17.6 mA when the radio is active in receive mode.

6.3.1 Current Consumption of Node when Sending EEPROM Data

Current consumption of Node was seen in the Oscilloscope when sending EEPROM data to the Gateway. In case of five nodes, every node gets the request from the gateway after every three seconds (600 ms x 5 = 3 sec) when the gateway radio receive time is selected to 600 ms. The radio of the node is turned on after every three seconds to receive the request from the gateway and the node responds by sending the data to the gateway. The voltage drop measured across the 2.5 Ω resistor when sending EEPROM data to the gateway is shown in figure 6.32.

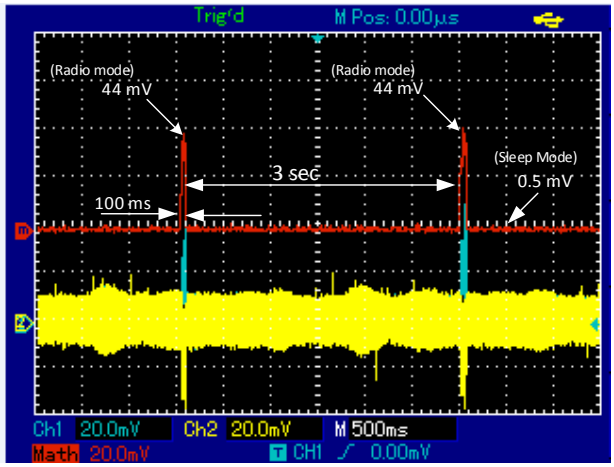


Figure 6.32: Node Current When Sending EEPROM Data

As shown in the plot, the node remains in low power mode for 3 seconds and the current measured across resistor was 200 μA(0.5 mV/2.5 Ω) in this duration. After 3 seconds, the radio turns on for a duartion of 100 ms and the current measured across resistor was 17.6 mA (44 mV/2.5 Ω). Finally it can be said that significant amount of current is reduced in sleep mode when sending EERPOM data to the gateway.

6.3.2 Current Consumption of Node when Sending Real Measurement Data

In this case also, the sleep mode current is same as of 200 μA for the duration of 1 minute. After 1 minute, the radio is turned on in receive mode to listen the gateway query. It listens, measures the current values and send it back to the Gateway. For this whole process the

radio remains in receive mode for a period of 1 second as shown in figure 6.33. The majority of the time is consumed for waiting gateway request.

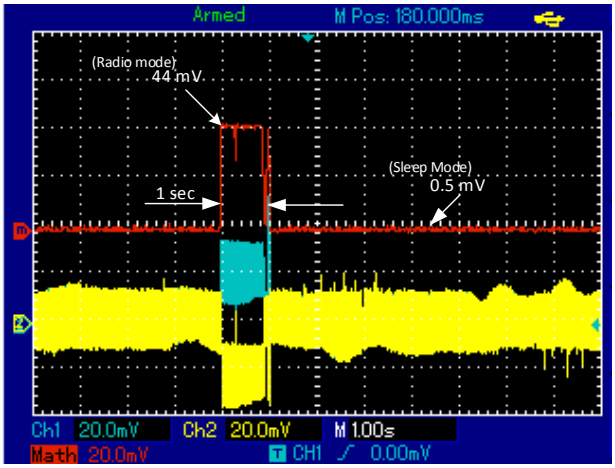


Figure 6.33: Node Current When Sending Real Measurement Data

Hence the node draws current of 17.6 mA for a duration of 1 second and then again goes in sleep mode for 1 minute.

6.3.3 Current Consumption in Offline Mode

In offline mode, Sensor nodes draw very low current in the rating of 40 μA measured by the multimeter. Current consumption is reduced in offline mode because of the no radio implementation in this mode as well as usage of low power mode 3 in the microcontroller.

7 Conclusion

The overall goals achieved in the thesis are the successfully interfacing of sensors (Sht25 and Tmp100) and the EEPROM with the Wizzimote board by I2C interface, interfaced sensors integration with the Dash7 module for the communication between nodes and gateway and high reading range capabilities of gateway to read sensor values from nodes in presence of noise.

Transmission output power of antenna or receiver sensitivity and low data rate transmission is the important factor to increase the range of Rf radio CC1101. In all the tests performed inside or outside the building, the transmission output power was set at the maximum value of 10 dBm and could not be increased further. The utmost requirement in the test was to obtain real time measurements from the interfaced sensors of sensor nodes. In the building test, sensor nodes were not positioned beyond 20 meter so the maximum reading range tested in the building observed to be 20 meter without any loss of packets at the base station.
Open field tests were completely under the influence of noise and despite of this factor the maximum reading range achieved in the test without any loss of packets at the gateway was found to be 60 m in the first and second test with their setting three arrangement of nodes. The highest range of 100 meter was tested in an open field trials with loss of packets, when the data rate of radio set to 27.8 kBaud.

The number of saved readings in the EEPROM of all sensor nodes were found to be equal which verifies that all the sensor nodes received gateway query in every test performed either inside or outside the building. The packets loss of nodes data at the gateway, programmed to specific radio receive time, becomes obvious due to the noise factor detected by the Rf analyzer which can become the source of signal latency.

If the tests are to be performed in a noise free environment, then the range of these nodes will be more promising with less packet loss at the Gateway end. The test was not carried out in a banana container but the above discussions and findings show promising signs of better results if it is tested in the container. In the future, the number of other sensors such as air flow sensor or other parameter sensing devices can also be interfaced with the sensor nodes by the I2C interface for the real time monitoring of other parameters too in Intelligent Container.

List of Figures

Bibliography

[1] The intelligent container. http://www.intelligentcontainer.com/.

[2] Reiner Jedermann, Ulrike Praeger, Martin Geyer, and Walter Lang. Remote quality monitoring in the banana chain. *Transactions of the Royal Society*, Vol.372, June 2014.

[3] Cantwell MI. In postharvest technology of horticultural crops (ed. a. kader). *2002 Summary table of optimal handling conditions for fresh produce.*

[4] Reiner Jedermann, Markus Becker, Carmelita Goerg, and Walter Lang. Testing network protocols and signal attenuation in packed food transports. *Int. J. of Sensor Networks*, Vol.09(Nos. 3/4):pp.170–181, 2011.

[5] Walter Lang, Reiner Jedermann, Damian Mrugala, Amir Jabbari, Bernd Krieg-Brueckner, and Kerstin Schill. The "Intelligent Container" A Cognitive Sensor Network for Transport Management. *IEEE SENSORS*, Vol.11(No.3):pp.688–698, March 2011.

[6] Reiner Jedermann, Thomas Poetsch, and Chanaka Lloyd. Communication techniques and challenges for wireless food quality monitoring. *Transactions of the Royal Society*, Vol.372, June 2014.

[7] Dash7 alliance. http://www.dash7-alliance.org/.

[8] WizziLab. Wizzikit the smallest and most versatile dash7 development kit, 2012. http://www.wizzilab.com/wp-content/uploads/2012/12/WizziKit-ProductBrief.pdf.

[9] WizziLab. Getting started with the wizzikit 2, 2012-2013. http://www.wizzilab.com/wp-content/uploads/2013/03/WizziKit2-Datasheet.pdf.

[10] Texas Instrument. Msp430 soc with rf core, September 2013. http://www.ti.com/lit/ds/symlink/cc430f5137.pdf.

[11] Aldo Briano. Msp430 launchpad low power mode, August 2010. http://processors.wiki.ti.com/index.php/MSP430_LaunchPad_Low_Power_Mode.

[12] Texas Instrument. *CC430 Family User's Guide*, May 2009. http://www.ti.com/lit/ug/slau259e/slau259e.pdf.

[13] *I2C Interface or TWI (Two Wire Interface).* http://www.engineersgarage.com/tutorials/twi-i2c-interface.

[14] Embedded Lab. *Inter-Integrated Circuit (I2C) communication*, May 2009. http://www.ti.com/lit/ug/slau259e/slau259e.pdf.

[15] John H. Davies. *MSP430 Microcontroller Basics.* Newnes, 2008.

[16] Texas Instrument. *Digital Temperature Sensor with I2C Interface*, 2007. http://www.ti.com/lit/ds/symlink/tmp100.pdf.

[17] Sensirion. *Humidity and Temperature Sensor IC*, 2014. http://www.sensirion.com/fileadmin/user_upload/customers/sensirion/Dokumente/Humidity/Sensirion_Humidity_SHT25_Datasheet_V3.pdf.

[18] Microchip. *64K I2C Serial EEPROM*, 2002. http://ww1.microchip.com/downloads/en/DeviceDoc/21189f.pdf.

[19] JP Norair. Introduction to dash7 technologies, March 2009. https://dash7.memberclicks.net/assets/PDF/dash7%20wp%20ed1.pdf.

[20] Maarten Weyn, Glenn Ergeerts, Luc Wante, Charles Vercauteren, and Peter Hellinckx. Survey of the Dash7 Alliance Protocol for 433MHz Wireless Sensor Communication. *International Journal of Distributed Sensor Networks*, 2013.

[21] AARONIA AG. http://www.aaronia.com/Datasheets/Spectrum_Analyzer/Handheld_Spectrum_Analyser_Spectran_HF-4000.pdf.

[22] Free space path loss: Details, formula, calculator. http://www.radio-electronics.com/info/propagation/path-loss/free-space-formula-equation.php.