FIELD ESTIMATION IN A WIRELESS SENSOR NETWORK USING LOW POWER AND LOW COST DESIGNED SENSORS

by

Veysel Yaman Akgün

Submitted to Graduate School of Natural and Applied Sciences
in Partial Fulfillment of the Requirements
for the Degree of Master of Science in
Electrical and Electronics Engineering

Yeditepe University

2017

FIELD ESTIMATION IN A WIRELESS SENSOR NETWORK USING LOW POWER AND LOW COST DESIGNED SENSORS

APPROVED BY:

Assist. Prof. Dr. Engin Maşazade .....
(Thesis Supervisor)

Prof. Dr. Duygun Erol Barkana .....

Prof. Dr. Lütfiye Durak Ata .....

DATE OF APPROVAL: .... /.... /2017

ACKNOWLEDGEMENTS

Primarily and foremost, I would like to thank to my parents for their endless support, help and encouragement throughout my education life.

Also, I wish to explain my sincere gratitude to Assist. Prof. Dr. Engin Maşazade who offers me the necessary help and support for the preparation of this thesis which is prepared with the help of his guide, extensive knowledge in communication systems, motivation and patience. Moreover, I would like to thank to committee members Prof. Duygun Erol Barkana who contributes my education and also thesis during critical times and Prof. Lütfiye Durak Ata for dealing with the thesis, accepting to be a jury member and feedbacks that promote and improve this research.

Furthermore, I would like to thank to my friends Abdulkadir Köse and Volkan Talha Doğukan for their helps. In addition, I would especially thank very much to academic staff of the Department of Electrical and Electronics Engineering of Yeditepe University.

Finally, in short, I would like to thank everybody who have contributed to my thesis in the process of this project preparation, provide and support me for my success and achievements during my education life.

Finally, I am grateful to The Scientific and Technological Research Council of Turkey (TUBITAK) which supports this project under Grant 113E220.

ABSTRACT

FIELD ESTIMATION IN A WIRELESS SENSOR NETWORK USING LOW POWER AND LOW COST DESIGNED SENSORS

A Wireless Sensor Network (WSN) observes desired quantities from a given region of interest using numerous cheap and simple sensors. Sensors transmit their measurements to a central unit called fusion center (FC) for final inference. In this thesis, we first design a sensor using low power and low cost components which are MSP430G2553 microcontroller, nRF24L01+ communication unit, and a Light Dependent Resistor (LDR). Using such sensors, we form a WSN. In this thesis, sensors measure light intensities at their locations and send their measurements to the FC. For the transmissions between sensors and FC, we define the communication protocol in a Carrier Sense Multiple Access (CSMA) manner. In this protocol, we explicitly define data request and data reply operations, data and control packet formats, acknowledgement (ACK), collision handling and collision avoidance procedures. In this work, we select the task of the WSN as field estimation. The FC forwards the gathered WSN data to a computationally powerful computer. Then, using computer, we first determine the spatial dependence between the sensor measurements called variogram. Then using the variogram, we perform field estimation using Ordinary Kriging where the light intensity at an unobserved location is estimated as a weighted sum of received LDR measurements. Our test results show that the predicted light measurement using Ordinary Kriging at a specific location is quite close to the actual LDR measurement at that location.

ÖZET

DÜŞÜK GÜÇLÜ VE DÜŞÜK MALİYETLİ TASARLANAN DUYARGALAR KULLANILARAK BİR TELSİZ DUYARGA AĞINDA ALAN KESTİRİMİ

Bir Telsiz Duyarga Ağı (TDA), belirli bir alandaki incelenmek istenen nicelikleri ucuz ve basit duyargalar kullanarak gözlemlemektedir. Duyargalar ölçümlerini merkezi birim, Tümleştirme Merkezi'ne (TM), son çıkarım için göndermektedirler. Bu tezde, ilk olarak düşük güç tüketimli ve düşük maliyetli birimler, MSP430G2553 mikrodenetleyicisi, nRF24L01+ haberleşme birimi, ve ışığa bağlı değişen direnç (LDR), kullanılarak bir duyarga tasarlanmaktadır. Bu duyargalar kullanılarak bir TDA oluşturulmaktadır. Bu tezde, duyargalar bulundukları noktalardaki ışık şiddetlerini ölçmekte ve ölçümlerini TM'ye göndermektedirler. Duyargalar ve TM arasındaki haberleşme için Taşıyıcı Tabanlı Çoklu Erişime (TTÇE) dayalı bir haberleşme protokolü tanımlanmaktadır. Bu protokolde, veri istek ve veri yanıtlama işlemleri, veri ve kontrol paket yapıları, onay, çarpışmayı ele alma, çarpışmadan kaçınma işlemleri açıkça tanımlanmaktadır. Bu çalışmada, TDA'nın görevi alan kestirimi olarak seçilmektedir. TM'de toplanan TDA verisi işlem gücü açısından kuvvetli bir bilgisayara aktarılmaktadır. Bilgisayar vasıtasıyla, ilk olarak yarı-ilinti olarak ifade edilen duyarga ölçümleri arasındaki uzaysal ilişki elde edilmektedir. Yarı-ilinti modeli kullanılarak, alan kestirimi Sıradan Kriging yöntemi ile gerçekleştirilmektedir. Sıradan Kriging yönteminde duyarga bulunmayan bir noktadaki ışık şiddeti, duyargalardan toplanan LDR ölçümlerinin ağırlıklı ortalamasıyla hesaplanmaktadır. Yapılan test sonuçları, Sıradan Kriging kullanılarak bir noktadaki tahmini alan şiddeti ile aynı noktadaki gerçek duyarga ölçümünün birbirlerine oldukça yakın olduğunu göstermektedir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS/ABBREVIATIONS

Ω	Ohm
A	Ampere
ACK	Acknowledgement
ADC	Analog-to-Digital Converter
AM	Active Mode
ARD	Auto Retransmit Delay
ART	Auto Retransmission
CCS	Code Composer Studio
CE	Chip Enable
cm	Centimeter
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
CSMA	Carrier Sense Multiple Access
CSN	Chip Select NOT
dBm	Decibel-milliwatt
DCO	Digitally Controlled Oscillator
DSP	Digital Signal Processing
ESB	Enhanced ShockBurst
FC	Fusion center
GFSK	Gaussian Frequency Shift Keying
GND	Ground
GPIO	General-Purpose Input/Output
HW	Hardware
I^2C	Inter-Integrated Circuit
IC	Integrated Circuit
IRQ	Interrupt Request
ISM	Industrial, Scientific and Medical

Kbps	Kilobit per Second
LDR	Light Dependent Resistor
LED	Light Emitting Diode
LPM	Low Power Mode
LSByte	Least Significant Byte
m	Meter
mA	Milliampere
MAC	Medium Access Control
mAh	Milliampere Hour
Mbps	Megabit Per Second
MCU	Microcontroller Unit
MISO	Master-In-Slave-Out
MOSI	Master-Out-Slave-In
MSByte	Most Significant Byte
MSE	Mean Squared Error
μA	Microampere
nA	Nanoampere
PC	Personal Computer
PID	Packet Identity
PRX	Primary Receiver (RX)
PTX	Primary Transmitter (TX)
RF	Radio Frequency
RPD	Received Power Detector
RSSI	Received Signal Strength Indicator
RX	Receive
SCK	Shift Clock
SMCLK	Sub-Main Clock
SPI	Serial Peripheral Interface
SW	Software
TI	Texas Instruments
TX	Transmit

UART	Universal Asynchronous Receiver/Transmitter
ULP	Ultra-Low Power
USB	Universal Serial Bus
USCI	Universal Serial Communication Interface
V	Voltage
VCC	IC Power-Supply Pin
WDT	Watchdog Timer
WSN	Wireless Sensor Network

1. INTRODUCTION

With the help of technological developments, systems which work without any human interaction have been rapidly spreading, due to the fact that they are able to perform lots of operations perfectly than humans without any errors or defects. In addition, human-less systems provide further advantages such as uninterrupted runtime, data storage, remote access controlling and so on. Therefore, they are preferred instead of the manual systems since it costs more money and the system owner also needs to consider the human related problems. Although there exists investment costs of human-less systems, they are advantageous in the long run to produce systematic and robust solutions. With the help of rapid technological developments in communication systems, the systems without human interaction are able to be processed and controlled over further distances. Communication systems originate from simple methods such as communication with smoke, communication with dove, communication with a messenger, to telegraph, telephone, cell phones and the internet. Recently used communication systems are developed from the discovery of Graham Bell and Thomas Watson's invention of voice communication over wire in 1875. Nowadays, communication systems are so developed, fast and also they will continue to improve. Nowadays, they are able to keep in contact with remote locations which can be as far away as the intercontinental places or the objects at the edge of the solar systems [4].

Wireless Sensor Networks (WSNs) are established in order to observe some quantities of interest over a certain region using numerous cheap and simple nodes called sensors. The sensor measurements can be then transferred to a far away location to deduce inference from the network. WSNs are becoming very useful and have widespread usage to measure the physical or environmental conditions of a region like its temperature, light intensity, pressure, and moisture. The usage of WSNs can be extended to military applications such as in the battlefield surveillance; or body area networks applications such as health monitoring. The WSN shown in Fig. 1.1 consists of distributed sensors and a fusion center (FC). Sensors measure quantities such as temperature, pressure, humidity, light etc. and they transmit their measurements to the FC for the final deduction. Since sensors are envisioned as simple, cheap and battery powered devices [5], sensors forming the WSN are required to be both energy

efficient and low cost to maximize the lifetime and minimize the deployment cost of the WSN. Note that the lifetime of each sensor is limited by the energy capacity stored in its battery. Therefore, sensors with low power consumption prolong the lifetime of the WSN. Additionally, monitoring the interested area with a large number of sensors improves the final inference performance while heavy sensor deployment raises the installation cost of the WSN.

In this thesis, we consider that the task of the WSN is field estimation. In field estimation problem, based on the received sensor measurements observed from known locations, the field intensity at unobserved locations in the field are predicted [6, 7]. Field Estimation problem perfectly overlaps with the usage of WSNs. In order to minimize the cost, the region of interest needs to be covered with limited number of sensors. Therefore, the unknown measurements at unobserved locations should be estimated from the observed data. Field estimation using WSNs has founded itself in many application areas, such as in precision agriculture to monitor the soil moisture [8], or soil organic matter [9], in weather activity monitoring such as weather temperature prediction [7] or wind field intensity estimation [10], or in mines to predict the coal amount [6]. Specifically nowadays, precision agriculture evolves as a key application in the ongoing Internet of Things researches [11].

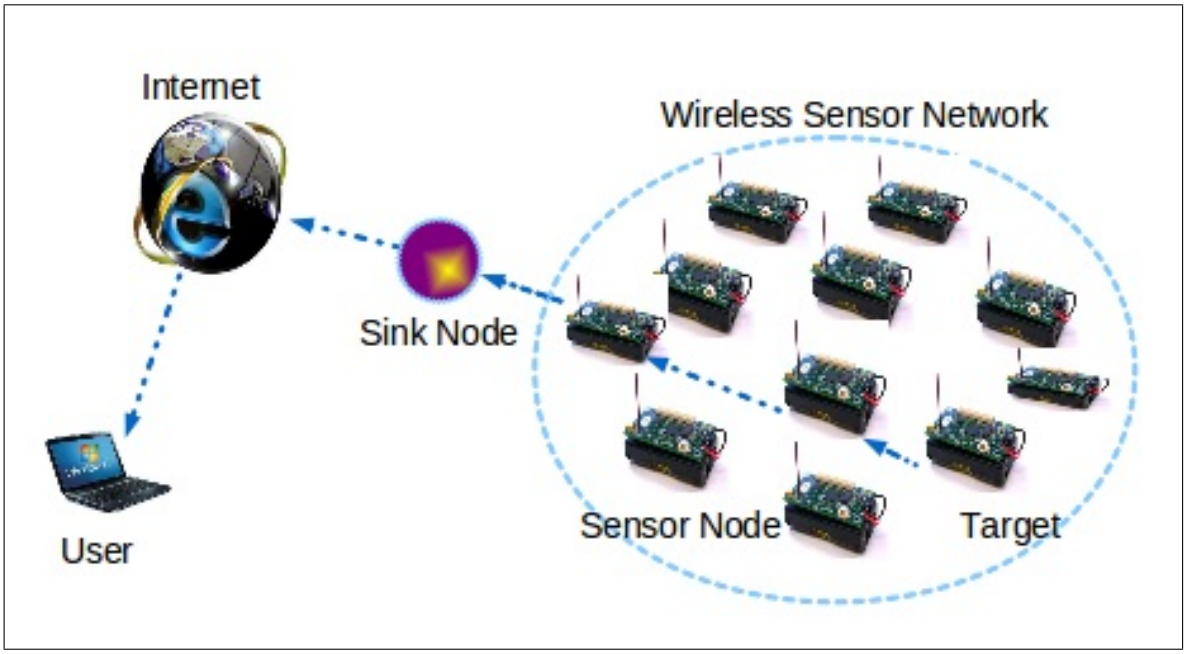


Figure 1.1. A Typical WSN Operation

In this thesis, we first design a sensor node using low cost and low power components which are microcontroller, MSP430G2553, radio transceiver unit, nRF24L01+, and sensory device, light-dependent resistor (LDR). Then using such custom designed sensors which have limited signal processing capabilities, we form a WSN by defining a communication protocol between sensors and the FC in a Carrier Sense Multiple Access (CSMA) manner. Finally, the sensor measurements collected at the fusion center are transferred to a PC which executes field estimation using MATLAB in real time.

1.1. LITERATURE SURVEY

It is possible to develop various WSN applications with the commercially available sensors such as MICAz, Telos, iMote2, Xbee [12–15]. As an example an iMote2 sensor is shown in Fig. 1.2. In [12], a vehicle tracking system was implemented by using the Arduino One microcontroller and the Xbee communication unit. In [13], ferromagnetic targets were identified and located by using MICAz units. In [14], the lifetime of sensors studies using the Telos units was examined. When a WSN is formed with commercial products, the programming burden required for the wireless communication unit and internal sensor operations are significantly lifted from the network designer. On the other hand, the formation of a WSN with such units may create serious costs for a limited budget study. As an example, the price of a single TelosB module is around 90 Euros [16].

Fortunately, it is possible to create a sensor node with low energy consumption and low cost by carefully choosing its fundamental elements such as microcontroller, communication unit, and the sensing unit [17]. Texas Instruments (TI) MSP430 family microcontrollers are popularly used in commercially available sensors due to their low power consumption and cost. As an example, the TelosB module uses TI MSP430F2618 as its microcontroller [16]. Additionally, MSP430F series microcontrollers have been also used to design sensors as in [18] and [19]. As compared to MSP430F series, MSP430G series microcontrollers are simpler, cheaper and also provide ultra lower power consumption. In the literature, there are few designs for uni-directional communication [20] or bi-directional communication [21] where both transmitter and receiver employ MSP430G2553 microcontroller.

Figure 1.2. A View of Imote2 with Battery Board and External Antenna

There are various alternatives for the wireless transceiver module to be connected to the microcontroller. These transceivers commonly operate at the free 2.4 GHz Industrial, Scientific and Medical (ISM) radio bands. Despite Texas Instruments CC1101, CC24XX, CC25XX wireless modules have been preferred in sensor node designs, Nordic Semiconductor's nRF24L01 wireless module has also been preferred over such products. nRF24L01 provides lower cost and lower power consumption as compared to CC1101 and Wi-Fi modules [22], CC24XX [23, 24] and CC25XX [24]. nRF24L01 wireless module has been previously used with Atmel ARM based microcontrollers [25–32]. Due to its ultra low power consumption, MSP430F series microcontrollers have also been used with nRF24L01 in [18, 19]. Most of these work consider a transmission between a transmitter and receiver pair [18,25,26,28,30–32]. On the other hand, there are also works which form a basic network and a communication protocol among such custom design nodes [19,27,29,33]. Furthermore,

in [34] various transmitters and receivers using nRF24L01 have been deployed and the trade-offs between data rate, package length versus the number of lost packets have been observed under different frequencies, packet size and the number of transmitters. Besides, models for simulating nRF24L01 device have also been presented in [23, 35]. Best of our knowledge there is no WSN application whose sensors are formed with low cost, simple, ultra low power MSP430G series microcontroller and the nRF24L01 wireless module.

For the sensing unit of the wireless sensor node, temperature, humidity, light, etc. sensory devices can be used for environmental monitoring [28, 29, 31, 32, 36] or pulse oximetry, electrocardiogram etc. sensory devices can be used for health monitoring [19, 33]. In order to measure the light intensity in a given region of interest, Light Dependent Resistor (LDR) can be easily included to the wireless sensor node design since it is easy to calibrate LDRs and set up a test area under different light conditions.

Upon completing the design of a wireless sensor node, a suitable communication protocol should be defined for the communication between sensors and the fusion center. Since sensors share the wireless transmission medium while communicating with the fusion center, an appropriate medium access control (MAC) policy needs to be defined. Different from the MAC protocols designed for the wireless networks which usually focus on efficient data delivery, the MAC protocol of a WSN should also consider the energy efficiency of the network by also seeking other objectives such as the individual sensor lifetime, the entire sensor network lifetime, or minimizing the number of transmissions [37–39]. Similar to the MAC policies for wireless networks, a WSN MAC policy can be defined on a contention based Carrier Sense Multiple Access (CSMA) policy [40] where the sensors first sense the medium, if the channel is idle, they then transmit their measurements to the fusion center. Since simultaneous sensor transmissions may cause collisions, collision avoidance and negotiation protocols between sensors and the fusion center need to be well defined.

The collected sensor measurements at the FC can be then transferred to a computationally powerful computer. For field estimation, based on the received sensor measurements observed at known locations, the spatial relation between the sensor measurements, i.e., the spatial covariance (or the spatial variogram) between the sensor measurements needs to be first

estimated. Upon modelling the variogram of the field, field estimation can be performed by using Ordinary Kriging [6, 7]. In Ordinary Kriging, the field is estimated at an unknown location according to the weighted linear sum of the received sensor measurements based on a Best Linear Unbiased Estimator [41]. In [36], using MICAz sensors, field estimation was performed using temperature and light measurements using energy efficient communication methods. In [42], using iMote2 sensors, temperature measurements were taken from the walls of a cold air storage, and the temperature distribution of the area was obtained. In [43], upon field reconstruction using Ordinary Kriging, a temperature control system was established by integrating the WSN into a smart home unit, where depending on the field estimation result, heating or cooling can be further performed in the area of interest. In [44], field estimation was performed under random packet losses. In [45], rather than executing field estimation using entire sensor data, field estimation was performed with sensor selection where the objectives were both minimizing the Ordinary Kriging error variance and the number of selected sensors. Finally, in [46], different spatial covariance functions to model the sensor measurements were compared in a multi-sensor system and the temperature and moisture distribution of the field were obtained.

1.2. LIST OF CONTRIBUTIONS

The main contributions of this thesis are briefly listed as follows:

- We first form a simple wireless sensor node using ultra low power, low cost components, MSP430G2553 as the microcontroller, nRF24L01+ as the wireless communication module and an LDR as the sensory device.
- Such custom design sensors are connected with a fusion center using a CSMA based communication protocol. For the protocol, we explicitly define data request and data reply procedures, data and control packet formats, acknowledgement and collision handling and avoidance procedures.
- Upon the reception of sensor measurements, we first obtain the experimental similarity, i.e., experimental variogram, of the field. Then, we determine the best mathematical variogram model fitting to the experimental variogram. Simulation results show

that, under different light conditions, field estimation is performed successfully at an unobserved location based on received sensor measurements.

1.3. THESIS ORGANISATION

The rest of the thesis is organized as follows,

In Chapter 2, we explain the sensor design and its main components. We state the general properties of the MSP430G2553 microcontroller and the nRF24L01+ communication module. Then, we further explain nRF24L01+'s communication procedure. Finally in this section, we present the calibration process of the LDR's.

In Chapter 3, we present the protocol written for the communications between sensors and the fusion center. We explicitly define data request and data reply operations, data and control packet formats, acknowledgement, collision handling and collision avoidance procedures.

In Chapter 4, we explain the mathematical foundations of Field Estimation problem. We first define experimental variogram which is a measure of the similarity between sensor measurements. We then define Ordinary Kriging which estimates the field intensity as a weighted summation of the received sensor measurements using the variogram.

In Chapter 5, we present our test results, and show the efficiency of the proposed work.

Finally, we devote Chapter 6 to our conclusions and address future research directions.

2. DESIGN OF SENSOR NODES

In this chapter, we examine the fundamental elements of a sensor node in detail. A sensor node has a micro-controller in order to perform basic operations, a transceiver unit for double sided duplex communication, a sensing unit which measures the desired attributes of the environment, a power supply for energizing the sensor node and a boarding unit for making appropriate connections between the components. The sensor node which is designed for this thesis is demonstrated in Fig. 2.1. In this work, we use MSP430G2553 as the microcontroller, nRF24L01+ as the communication module, and Light Dependent Resistor (LDR) as the sensing unit.

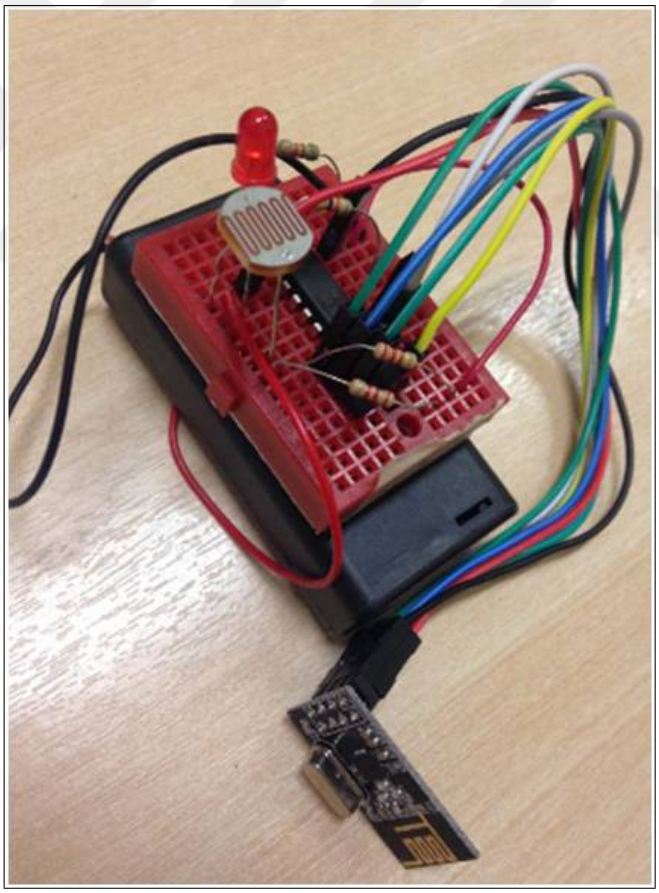


Figure 2.1. A View of Simple Sensor Node

These fundamental elements are handled and explained below in depth.

2.1. MICROCONTROLLER

The family of MSP430 which is a family of ultra-low power (ULP) microcontrollers of Texas Instruments is appropriate in terms of its low power consumption and low cost as compared to other microcontrollers such as ATmega328 [47], ATmega2560 [48], ATmega32u4 [49], etc. Although MSP430F series Microcontroller Units (MCUs) are widely used in sensor design [18, 19], MSP430G series MCUs also provide low power consumption but they are simpler and cheaper as compared to MSP430F series MCUs. As an example, the prices of MSP430G and MSP430F series MCUs are compared in Table 2.2 where the price of a single MSP430G2553 MCU is around 2.65 American Dollar [2]. From the table, it is easily seen that the unit price decreases as the number of microcontrollers bought increases. In addition, the technical specifications of MSP430G and MSP430F microcontrollers are compared in Table 2.1. Thus, MSP430G2553 16-bit microcontroller shown in Fig. 2.2 is used to realize the project.

MSP430G2553 has a Central Processing Unit (CPU) with operating frequency up to 16MHz [50]. Moreover, it supports communication protocols to connect to other microcontrollers and peripheral devices using Inter-Integrated Circuit (I^2C), Serial Peripheral Interface (SPI), and Universal Asynchronous Receiver/Transmitter (UART). Besides, it has 8 Analog to Digital Converter (ADC) Channels to measure the quantities of interest. The ADC operations are done with 10 bits. In addition to 8 Channels ADC, there are also internal temperature and voltage sensors to measure the chip surface temperature and the supply voltage of the CPU. It has a wide supplying voltage range from 1.8V to 3.6V so that it can be fed by numerous sources when the voltage supplies are different. Also, it has a 20 pin Integrated Circuit (IC) package.

Beyond, MSP430G2553 has two 16 Bit Timer Modules inside called Timer A and Timer B so, the basic clock operations can be done by using them separately. Also, it is designed to work with low power modes (LPMs) as in the modern microcontrollers since power management is a critical thing which extends the life of battery-controlled devices. The MSP430 has totally six power modes including one active and five low-power modes as shown in Table 2.3. For

Table 2.1. Some Technical Properties of MSP430G2553 and MSP430F2618 [2]

Product Info & Specifications	**MSP430F2618TZQWR**	**MSP430G2553IN20**
Height:	0.74 mm	4.57 mm
Length:	7 mm	24.33 mm
Package/Case:	BGA-113	PDIP-20
Packaging:	Reel	Tube
Program Memory Type:	Flash	Flash
Series:	MSP430F2618	MSP430G2553
Width:	7 mm	6.35 mm
Brand:	Texas Instruments	Texas Instruments
Data RAM Size:	8 kB	512 B
Interface Type:	I2C, IrDA, JTAG, LIN, SPI, UART, USCI	I2C, IrDA, SPI, UART
Mounting Style:	SMD/SMT	Through Hole
Number of I/Os:	64 I/O	16 I/O
Program Memory Size:	116 KB	16 KB
ADC Resolution:	12 bit	10 bit
Core:	MSP430	MSP430
Data Bus Width:	16 bit	16 bit
Maximum Clock Frequency:	16 MHz	16 MHz
Maximum Operating Temperature:	+ 105 C	+ 85 C
Minimum Operating Temperature:	- 40 C	- 40 C
Number of ADC Channels:	8 Channel	8 Channel
Number of Timers/Counters:	2 Timer	2 Timer
Operating Supply Voltage:	1.8 V to 3.6 V	1.8 V to 3.6 V
Processor Series:	2 Series	2 Series
Standard Pack Qty:	2500	20
Tradename:	MSP430	MSP430

Table 2.2. Price Comparison of MSP430G2553IN20 and MSP430F2618TZQWR [2]

Number of Units	MSP430G2553 Unit Price (in $)	MSP430F2618TZQWR Unit Price (in $)
1	$2.65	$12.21
10	$2.39	$11.22
25	$2.27	$10.92
100	$1.92	$9.49
250	$1.80	$8.72
500	$1.58	$8.43
1000	$1.18	$7.57

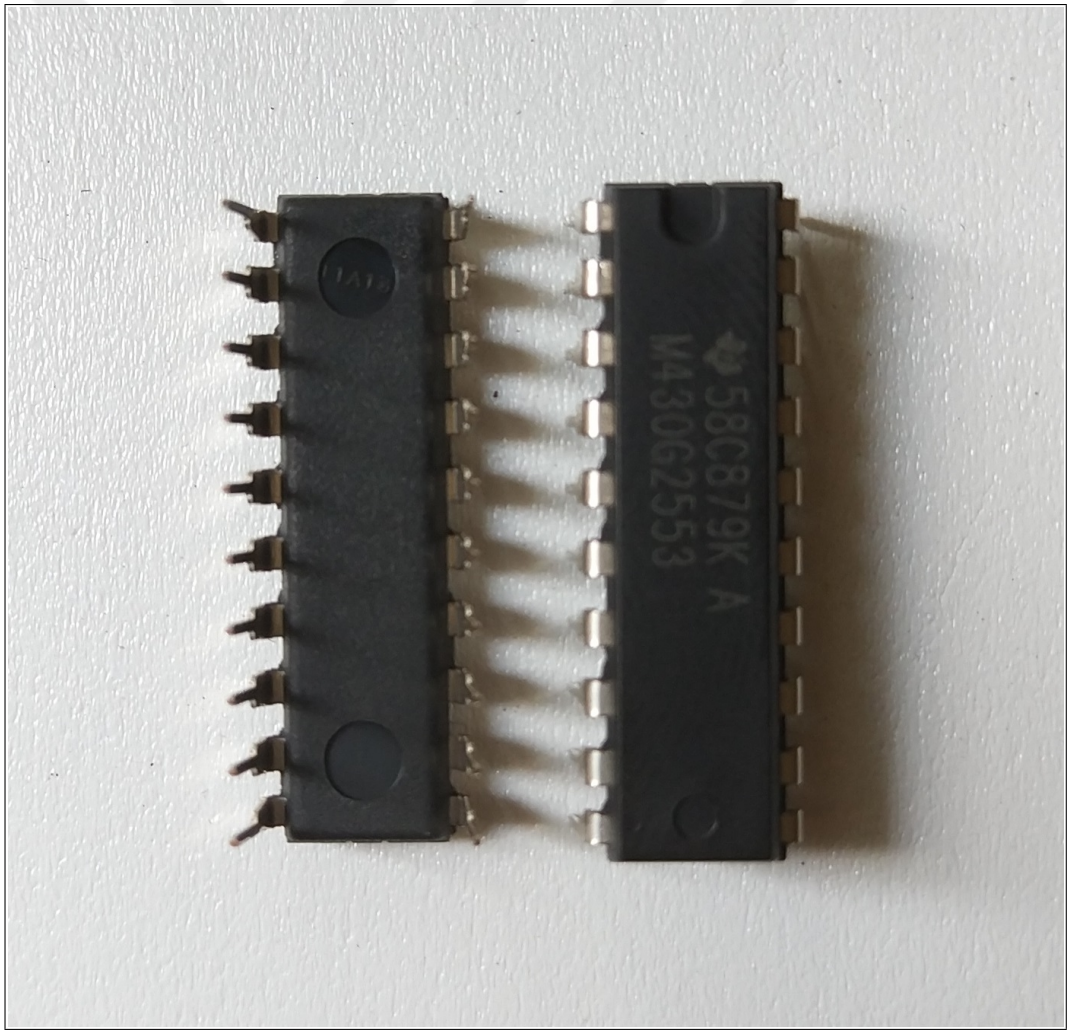


Figure 2.2. MSP430G2553 Microcontroller (TI)

instance, it consumes 330µA/MHz in active power and 0.7µA in standby Power for Low Power Mode 3 (LPM3). The wake-up time is about 1.5µs while the microprocessor is in LPMs. Furthermore, there is a watchdog timer (WDT) which detects the trouble if fault or error occurs in the system, and recover the CPU from infinite loop by resetting the whole system. In the designed system, LPM1 is used to optimize the power management.

Table 2.3. The Low Power Modes (LPMs) of MSP430G2553 Microcontroller [3]

	Functionality	
Mode Type	**Active**	**Deactive**
Active Mode (AM) - Draws about 230 µA current.	CPU, all clocks and enabled peripheral modules	-
Low-Power Mode 0 (LPM0) - Draws about 56-µA current.	SMCLK and ACLK	CPU and MCLK
Low-Power Mode 1 (LPM1) -	SMCLK and ACLK	CPU, MCLK, and DCO(if it is not used)
Low-Power Mode 2 (LPM2) - Draws about 22-µA current.	ACLK and DCO	CPU, MCLK, and SMCLK
Low-Power Mode 3 (LPM3) - Draws about 0.5-µA current. (*Standby* mode)	ACLK	CPU, MCLK, SMCLK, and DCO
Low-Power Mode 4 (LPM4) - Draws about 0.1-µA current. (*Off* mode)	Only RAM is retained.	CPU, all clocks, and the crystal oscillator

The MSP430G2553 has four calibrated Digitally Controlled Oscillator (DCO) frequencies which are 1 MHz, 8 MHz, 12 MHz, and 16 MHz. Also, operating frequency division can be selected as 1, 2, 4, and 8. In addition, Universal Serial Communication Interface (USCI) module is used for serial communication. USCI supports asynchronous and synchronous communication protocols. The asynchronous communication protocols are UART, enhanced UART with automatic baud rate detection (LIN), and IrDA. Further, synchronous communication protocols consist of SPI (3 or 4 pin) and I^2C. USCI consists of two different modules called USCI-A0 and USCI-B0. SPI (3 or 4 pin), UART, enhanced UART, and IrDA are supported by USCI_A0 while SPI (3 or 4 pin) and I^2C are supported by

USCI_B0. The microcontroller may be give a report about the communication and its registers via UART protocol. In this work, USCI-A0 module is defined for the UART communication and USCI-B0 module is defined for the SPI communication in order to use both UART and SPI simultaneously since USCI-B0 module is not capable of UART communication.

2.2. TRANSCEIVER UNIT

In this work, we select nRF24L01+ shown in Fig. 2.3 as the transceiver unit since it provides low cost and low power consumption as compared to CC1101, CC24XX, CC25XX wireless modules [22–24]. Under these two criteria, longer battery life can be achieved.

2.2.1. General Properties of nRF24L01+

nRF24L01+ Single Chip Transceiver Antenna operates at 2.4 GHz license-free Industrial, Scientific and Medical (ISM) band which is used in common for most of the devices that are working in licence free worldwide ISM band. It supports 126 channels from 2.400 GHz to 2.525 GHz shown in Fig. 2.4. The nRF24L01+ channel spacing is 1 MHz which gives 126 possible channels for 250 kbps and 1 Mbps air data rate while the channel spacing is 2 MHz for 2Mbps air data rate. Also, it is a ultra low power communication unit which is a highly integrated Radio Frequency (RF) transceiver Integrated Circuit (IC). In addition, the air data rates can be selected as 250 kbps, 1 Mbps, and 2 Mbps [1]. The price of the nRF24L01+ is around 1 American Dollar [51]. Moreover, the communication infrastructure between nRF24L01+ and microcontroller is a little bit more complex than the communication units such as HC05 and HC06 Bluetooth Modules [52] which use UART protocol because nRF24L01+ uses SPI protocol for the communication with a microcontroller. Besides, it uses Gaussian Frequency Shift Keying (GFSK) modulation at the radio front end. In addition, some parameters are user configurable such as frequency channel, output power and air data rate.

Besides, the chip of nRF24L01+ operates in ultra low power as in the microcontroller. It consumes about 11.3mA while transmitting a data packet at 0dBm output power level, and it

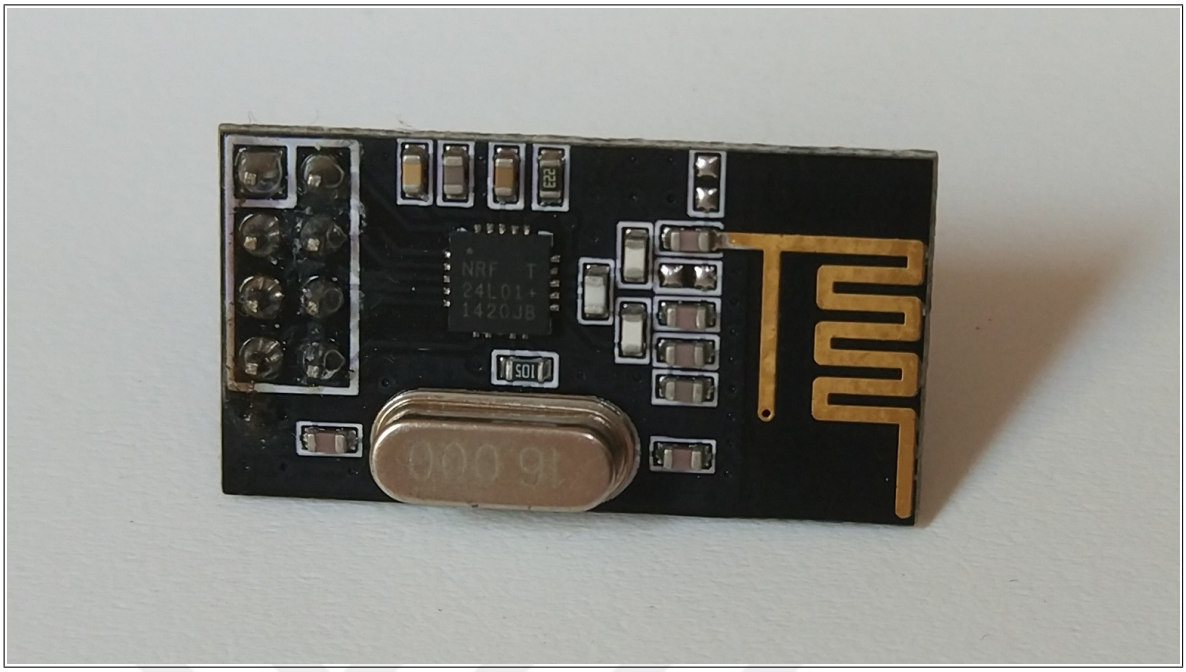


Figure 2.3. nRF24L01+ Transceiver Unit (NS)

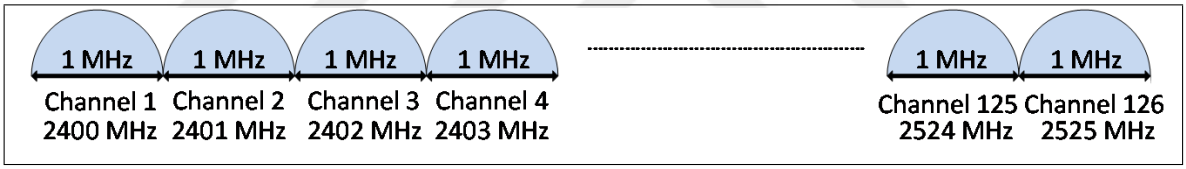


Figure 2.4. nRF24L01+ Channels

consumes about 13.5mA while receiving at 2Mbps air data rate. Furthermore, it consumes 900nA in power down mode and 26µA in standby mode (standby-I) [1]. Also, there is a voltage regulator on the chip and supply range is so wide from 1.9V to 3.6V as enough for common feeding levels. Detailed information about energy consumption at other output power levels including configurable transmitting RF powers and current consumption of the transceiver are shown in Table 2.4.

Additionally, there are totally 8 pins which are Ground (GND) which refers to potentially zero for the circuit, IC Power-Supply Pin (VCC) which is the supply voltage side, Chip Enable (CE) for Receive (RX)/Transmit (TX) enable, Chip Select Not (CSN) for enabling the SPI commands and responds, SPI Shift Clock (SCK) for SPI clock up to 10 MHz,

Table 2.4. RF Output Power Setting for the nRF24L01+ [1]

SPI RF-SETUP (RF_PWR)	RF Output Power	DC Current Consumption
11	0 dBm	11.3 mA
10	-6 dBm	9.0 mA
01	-12 dBm	7.5 mA
00	-18 dBm	7.0 mA

Master-Out-Slave-In (MOSI) for SPI output where data are sent from microcontroller to nRF24L01+, Master-In-Slave-Out (MISO) for SPI input where data are sent from nRF24L01+ to microcontroller, Optional Interrupt Request (IRQ) for interrupt about RX/TX status in case of received or sent packet. In addition, CE, CSN, SCK, MOSI are inputs and MISO and IRQ are outputs of the transceiver antenna.

nRF24L01+ has drop-in compatibility with nRF24L01 which is the older and simple version of nRF24L01+. Also, it is on-air compatible with other versions as nRF2401A, nRF2402, nRF24E1 and nRF24E2 in 250 kbps and 1 Mbps air data rates. Further information can be found in [1].

In addition, the circuit schematic including the pin connections between MSP430G2553 and nRF24L01+ is shown in Fig. 2.5. From the Fig. 2.5, P 1.1 and P 1.2 of MSP430G2553 are reserved for UART operations. Moreover, the ADC measurement is taken by using P 1.4. Besides, P 1.5 is used as SPI CLK, P 1.6 is used as SPI MISO, and P 1.7 is used as SPI MOSI in 3 pin SPI. Furthermore, P 2.0, P 2.1, P 2.2 are used for CE, CSN, IRQ, respectively. Additionally, the system is supplied with 2 AA batteries and there is an on-off switch to turn on or shutdown the device completely. For example, the switch can be used to disconnect the device from energy supplier.

In this work, we select the output power of nRF24L01+ as 0 dbM. In addition, the communication on air is realized in one channel frequency operated at 2.520 GHz which is the 121th channel. Besides, the air data rate is chosen as 1 Mbps. Moreover, the microcontroller MSP430G2553 operates at 8 MHz although it can be operated up to 16 MHz as the maximum

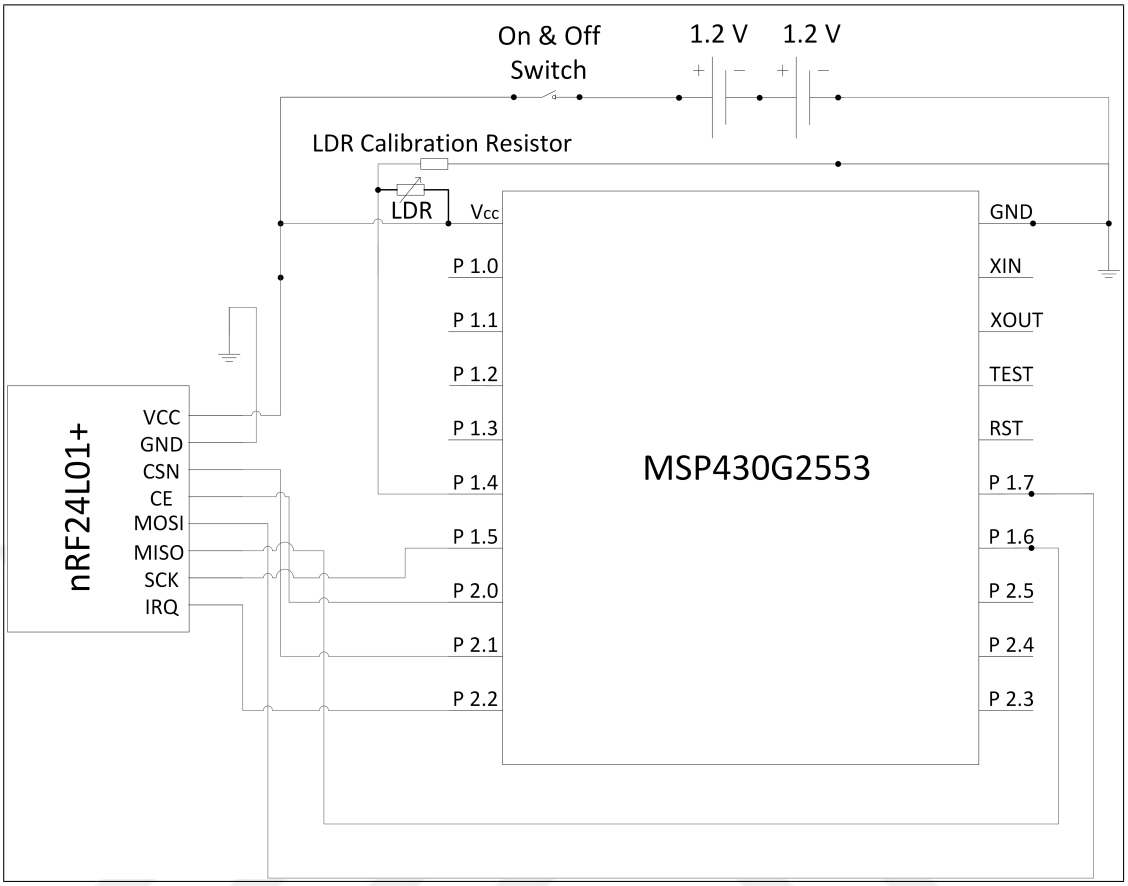


Figure 2.5. Circuit Schematic of Sensor Node

level. This is because the maximum clock cycle limit for communication via SPI between MSP430G2553 and nRF24L01+ is limited with 8 MHz since SPI (USCI) uses Sub-Main Clock (SMCLK) and SPI speed limit for nRF24L01+ is 10 MHz. In other words, SMCLK may prefer less than 10 MHz.

2.2.2. Packet Format of nRF24L01+

nRF24L01+ has Enhanced Shockburst (ESB) specific packet format layer which features automatic package assembly and timing, automatic acknowledgement and retransmission of packets. It provides high communication performance and low power consumption even if it is used with low cost microcontrollers. It has additional features as automatic packet handling and auto packet transaction handling. The Enhanced Shockburst packet format is shown in Table 2.5.

The first part of the packet, the preamble, is used to synchronize the receivers demodulator to the incoming packet as bit stream. It has 1 byte. There are two options in preamble side as 01010101 or 10101010. If the first bit of address register is 1, the preamble is selected automatically starting with 1 which is 10101010. Otherwise, it is automatically set to the preamble starting with 0. Here, bit altering is necessary in order to stabilize the receiver and ensure that the packet is coming.

The second part is about the receiver address which represents the pipe address of the receiver. It makes sure that the packet is received by the intended receiver. This prevents from accidental cross talk among multiple nRF24L01+'s. Address width is configurable as 3, 4 or 5 bytes.

The third part is packet control field. Detailed information about packet control field is shown in Table 2.6. Here, payload length field is used to define payload length up to 32 bytes. Packet Identity (PID) field is used to distinguish a new or retransmitted packet. No Acknowledgment flag is only used for auto acknowledgement feature. When the flag is high, the receiver understands that the packet will not be acknowledged automatically.

The fourth part consists of payload where the content of the payload is defined by the user. The width of the payload can be selected up to 32 bytes. Table 2.7 represents our payload format used in this thesis. Since we need 15 bytes, the size of the packet is selected as 16 bytes under the criteria that the payload can only be the power of 2's. The payload includes a start byte, a receiver no id, a transmitter no id, time information, and measurements. Since the MCU has two different timer clocks which are capable of timing operations, the timers work and count to know what the date and time are. The details of the packet format shown in Table 2.7 will be examined in detail with all of their contents in Chapter 3.

The fifth and last part of Table 2.5 is Cyclic Redundancy Check (CRC) which is responsible for error detection mechanism. The address, Packet Control Field and Payload are used to calculate 1 or 2 bytes CRC value.

Table 2.5. An Enhanced ShockBurst Packet with Payload (0-32 bytes) [1]

Preamble 1 byte	Address 3-5 byte	Packet Control Field 9 bit	Payload 0 - 32 byte	CRC 1-2 byte

Table 2.6. Packet Control Field [1]

Payload length 6 bit	PID 2 bit	NO_ACK 1 bit

2.2.3. Data Pipes of nRF24L01+

nRF24L01+ has six parallel data pipe MultiCeiver in RX mode which enables simultaneous connections up to 6 transmitters as shown in Fig. 2.6 with unique addresses. In other words, the physical address of each data pipe must be different which are decoded in the nRF24L01+. When an nRF24L01+ is configured as the Primary Receiver (PRX), it can receive data which are sent to one of the different data pipes on a certain frequency channel. Also, all data pipes can operate with Enhanced ShockBurst packet format and only a packet can be received from only one of the six data pipes at a time. This means that nRF24L01+ searches each data pipe addresses sequentially where two way communication is done among devices at the same time. Besides, when one nRF24L01+ is adjusted as PRX, six nRF24L01+'s can be configured as Primary Transmitter (PTX) which is the maximum top level that one PRX can communicate. CRC enabled/disabled, CRC encoding scheme, RX address width, frequency channel, air data rate, and LNA gain are the other adjustments which can be set in common for all data pipes.

As shown in Fig. 2.7, up to 5 bytes configurable address can be defined for each data pipe under the following criteria. None of the pipes can be assigned to same address where Pipe 0 and Pipe 1 addresses can be defined arbitrarily. When 5 bytes unique address is defined for data pipe 1, the remaining four pipes must use the four most significant address bytes of data pipe 1. For the rest of the pipes, only the Least Significant Bytes (LSBytes) of 5 bytes can be configurable. In other words, LSBytes can only be assigned which separates the unique addresses of data pipes from each other and 4 Most Significant Bytes (MSBytes) are the same

Table 2.7. General nRF24L01+ Payload Package Byte

	Description
Payload 1	Start Byte
Payload 2	Delivery Address
Payload 3	Sender Address
Payload 4	Year
Payload 5	Month
Payload 6	Day
Payload 7	Hour
Payload 8	Minute
Payload 9	Second
Payload 10	Protocol Type
Payload 11	Threshold Value
Payload 12	Sleeping Time
Payload 13	Adjustable and Reserved For the Package Type
Payload 14	Adjustable and Reserved For the Package Type
Payload 15	Adjustable and Reserved For the Package Type
Payload 16	Unused

for each data pipe address.

PRX receives the packets sent from more than one PTX using MultiCeiver and Enhanced ShockBurst. When data are received from PRX, PRX takes the unique data pipe address of the transmitter and uses it as the TX address of the ACK packet. This helps to send the ACK packet to the correct PTX which transmits data. Fig. 2.8 is a simple example of address definition for the PRX and PTXs. As seen on the figure, RX_ADDR_Pn is derived from the RX_ADDR_P0 which is defined for PRX unique pipe address of the data pipe 0. Also, RX_ADDR_P0 and TX_ADD must be the same for PTX to sense the ACK packet as pipe address for the designated pipe on PRX. For example, PRX receives the PTX1 from pipe1 address (RX_ADDR_P1) 0xB3B4B5B6F1 by using data pipe1 of the PRX. The ACK of PTX1 is sent to pipe0 of the PTX1 as RX_ADDR_P0 0xB3B4B5B6F1. Again, PRX receives the PTX2 from pipe2 address (RX_ADDR_P2) 0xB3B4B5B6CD via data pipe2 of the PRX. The ACK of PTX2 is sent to

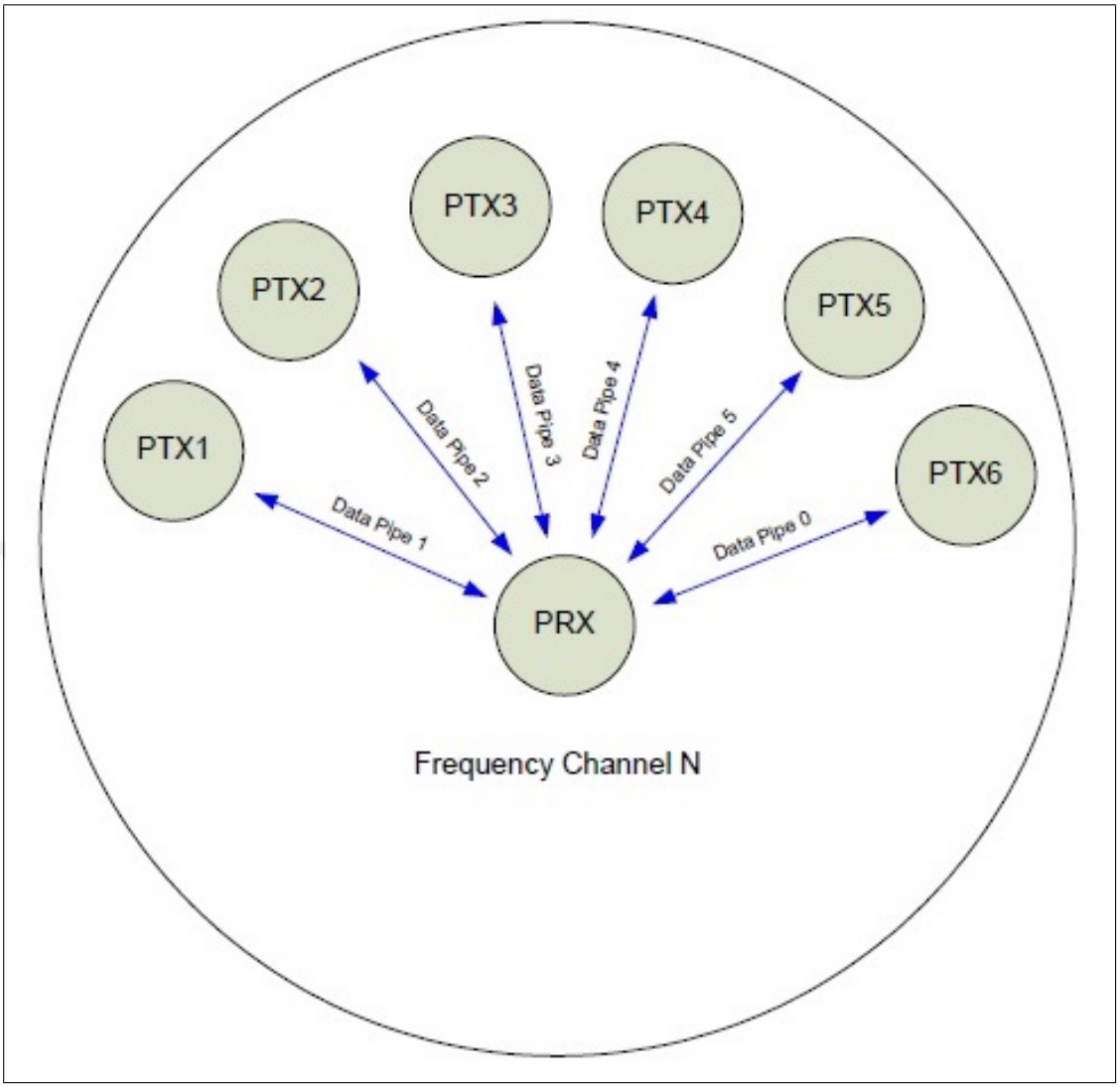


Figure 2.6. PRX using MultiCeiver [1]

pipe0 of the PTX2 as RX_ADDR_P0 0xB3B4B5B6CD. PRX receives the PTX3 from pipe3 address (RX_ADDR_P3) 0xB3B4B5B6A3 via data pipe3 of the PRX. The ACK of PTX3 is sent to pipe0 of the PTX3 as RX_ADDR_P0 0xB3B4B5B6A3. PRX receives the PTX4 from pipe4 address (RX_ADDR_P4) 0xB3B4B5B60F by using data pipe4 of the PRX. The ACK of PTX4 is sent to pipe0 of the PTX4 as RX_ADDR_P0 0xB3B4B5B60F. PRX receives the PTX5 from pipe5 address (RX_ADDR_P5) 0xB3B4B5B605 by using data pipe5 of the PRX. The ACK of PTX5 is sent to pipe0 of the PTX5 as RX_ADDR_P0 0xB3B4B5B605. Finally, PRX receives the PTX6 from pipe0 address (RX_ADDR_P0) 0x7878787878 via data pipe0 of the PRX. The ACK of PTX6 is sent to pipe0 of the PTX6 as RX_ADDR_P0 0x7878787878.

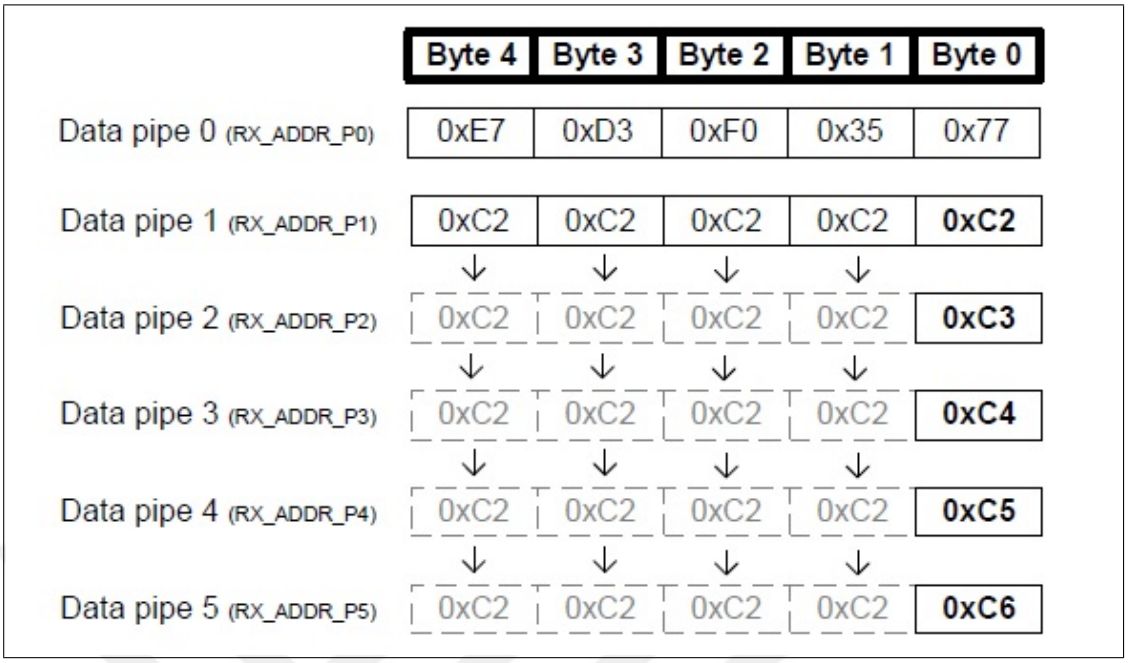


Figure 2.7. Addressing Data Pipes from 0 to 5 [1]

In addition, after one of the data pipes receives a packet completely, other data pipes can begin to receive incoming data. When more than one PTX start to transmit data to one PRX, Auto Retransmit Delay (ARD) can be an option to get rid of the Auto Retransmission (ART) so that PTXs transmitting data only inhibit each other once.

Also, there are three Acknowledgement (ACK) options in nRF24L01+ radio transceiver unit which are no ACK, ACK with empty package and ACK with payload. For this thesis, ACK with payload is selected, where the receiver sends an ACK to the transmitter by replying with the received package. Since each message has a specific and unique information like sensor node number, time information and so on, the transmitter validates the received ACK with the previously transmitted data packet. Moreover, additionally CRC check is done by the nRF24L01+ to secure the packet of message.

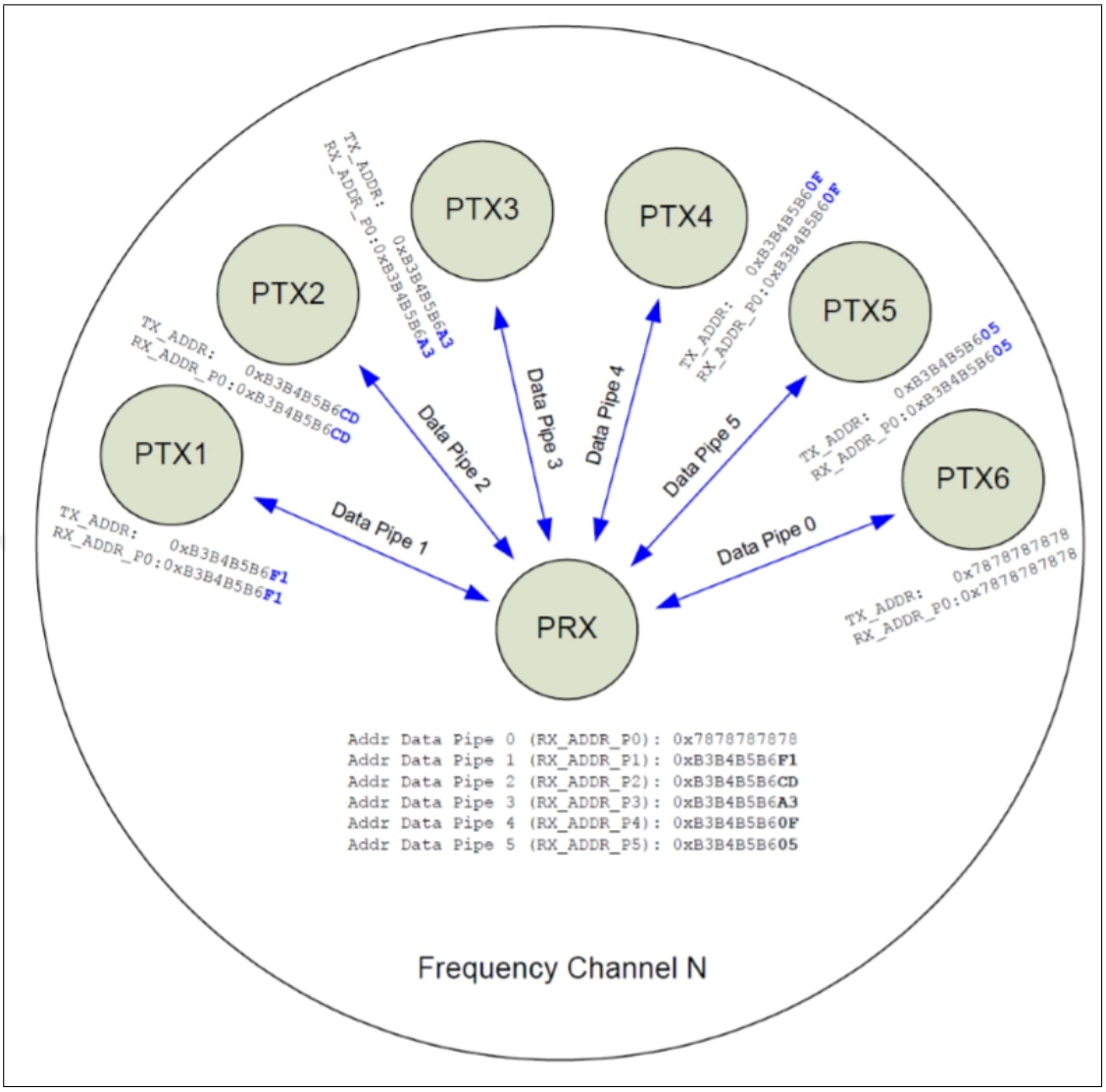


Figure 2.8. Example of Data Pipe Addressing in MultiCeiver [1]

2.3. A SIMPLE TWO WAY COMMUNICATION BETWEEN A TRANSMITTER AND A RECEIVER

An example of a simple communication algorithm shown in Fig. 2.9 is explained below. In Fig. 2.9, the FC acts as the receiver and the sensor acts as the transmitter where other devices may also exist in the same transmission channel. The detailed information about nRF24L01+ functions [53] and header files are given in Appendix A and Appendix B, respectively.

Firstly, when the microcontroller powers up and starts to work, it runs the code which

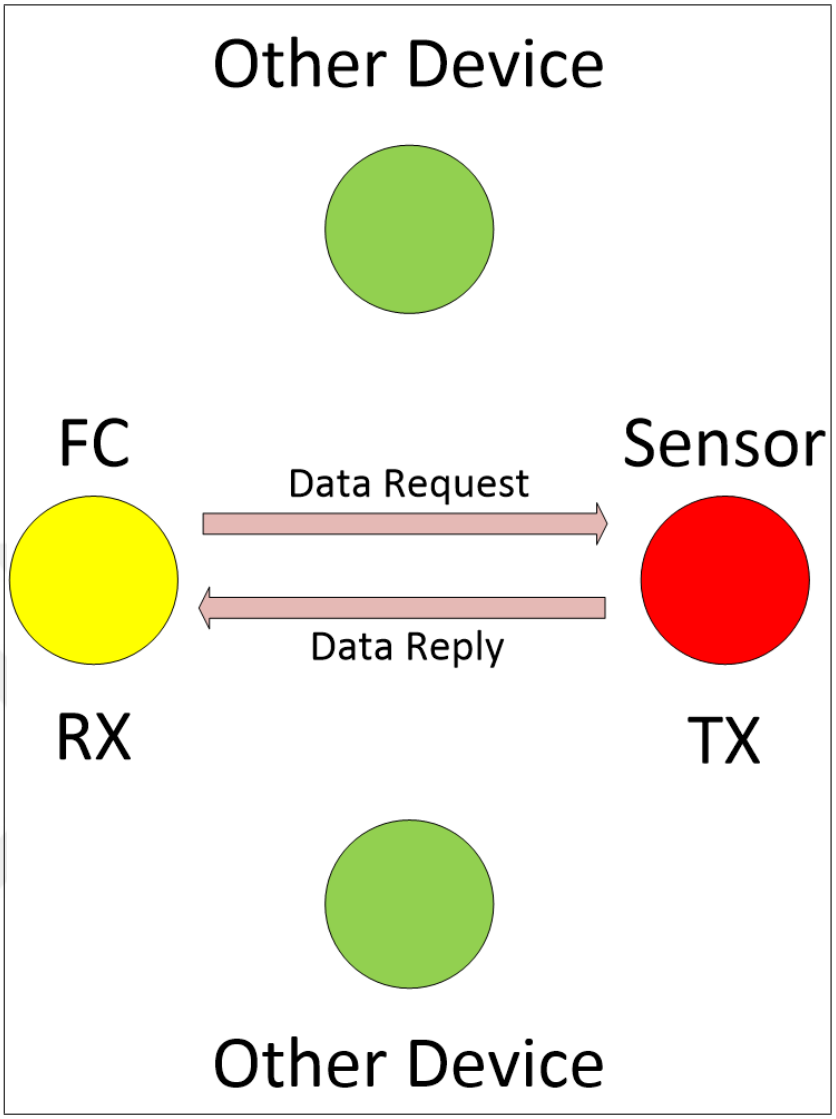


Figure 2.9. Simple Two Way Communication Among Peripheral Devices

adjusts General-Purpose Input/Output (GPIO) pins and other default registers. The communication side with nRF24L01+ transceiver starts with msprf24_init() function to initialize the communication part making ready. After that, it opens the pipes needed by using msprf24_open_pipe function. Then, it defines its own unique receiving and transmitting addresses with w_rx_add(char *addr) and w_tx_add(char *addr) functions. To finalize the initializing part, followings are loaded in an order which are defined at the beginning side msprf24_set_pipe_packetsize, msprf24_set_address_width(), msprf24_set_channel(), msprf24_set_speed_power(), and msprf24_enable_feature(char feature). Now, the transceiver module is ready to communicate with its own microcontroller and other transceivers.

In other words, the necessary settings are loaded and the system makes ready to work. After the configuration side finishes, transceiver starts listening the medium by using msprf24_activate_rx() function.

The reception algorithm of the transceiver unit is shown in Fig. 2.10. Transceiver unit scans the air with all defined pipe addresses simultaneously. When the address of the packet matches with one of the pipe address, it generates an interrupt. By the way, the MCU controls the transceiver unit by using $rf\_irq\&RF24\_IRQ\_FLAGGED$ command at the same time to check that there is any incoming data. If incoming data is available, MCU closes the interrupt flag of the transceiver unit by using $rf\_irq\& =\sim RF24\_IRQ\_FLAGGED$ command. Then, MCU uses $msprf24\_get\_irq\_reason()$ function to learn the interrupt reason. If $(rf\_irq\&RF24\_IRQ\_RX)||msprf24\_rx\_pending()$ command is true for the receiver side, it means that this interrupt is about the receiving operations and there is at least one incoming data pending. MCU uses $r\_rx\_peek\_payload\_size()$ and $r\_rx\_payload(uint8\_tlen, uint8\_t* data)$ functions to learn the data size up to 32 bytes and incoming data, respectively. After reading the incoming data, MCU checks the payload address whether is matched to its own register address or not. If the addresses are matched with each other, it sends an acknowledgement to notify the transmitter MCU that the data are received successfully. Function $w\_ack\_payload(uint8\_tpipe, uint8\_tlen, uint8\_t * data)$ is used to send ACK with received payload to the transceiver unit to notify the transceiver of the transmitter. After ACK is sent, receiver interrupt is cleaned with $msprf24\_irq\_clear(RF24\_IRQ\_RX)$ function. If $(rf\_irq\&RF24\_IRQ\_RX)||msprf24\_rx\_pending()$ command is false or the addresses are not matched, MCU ignores the data. Then, the general interrupt register is cleaned by using $msprf24\_irq\_clear(rf\_irq)$ and MCU goes back to check the transceiver unit by using $rf\_irq\&RF24\_IRQ\_FLAGGED$ command.

The transmitter algorithm of the transceiver unit is shown in Fig. 2.11. When there is a transmission needed, firstly MCU waits for the delay if there is. Then, communication channel is adjusted by using $msprf24\_set\_channel()$ and the contents of TX FIFOs are erased by using $flush\_tx()$. After that, the buffer array is created according to the payload format as in Table 2.7. When the buffer array is done, MCU checks the air by using $r\_reg(RF24\_RPD)$ function. If there is a channel activity, it waits for a delay. Then,

MCU again checks the medium. When the medium is clear to transmit, MCU sends the buffer array to the transceiver unit by using $w\_tx\_payload(uint8\_tlen, uint8\_t * data)$ function. At that point, data are uploaded to the transceiver unit registers and it waits for the transmission. Function $msprf24\_activate\_tx()$ is needed to transmit the data by the MCU. Then, MCU goes to LPM1 to wait for the communication result. After MCU leaves from LPM, it uses $msprf24\_get\_irq\_reason()$ function to learn that if there is a transmission interrupt. This refreshes the rf_irq register. Therefore, if rf_irq register does not contain a transmission interrupt, MCU goes to retransmission. If rf_irq contains a transmission interrupt, the criteria $!((rf\_irq \& RF24\_IRQ\_TX)||(rf\_irq \& RF24\_IRQ\_TXFAILED))$ is checked. If it is true, this means the interrupt is related with transmission. MCU checks whether the $rf\_irq \& RF24\_IRQ\_FLAGGED$ command is true. If it is true, MCU closes the interrupt flag of the transceiver unit by using $rf\_irq \& = \sim RF24\_IRQ\_FLAGGED$ command. Then, MCU uses $msprf24\_get\_irq\_reason()$ function to learn the interrupt reason. After that, if ((rf_irq & RF24_IRQ_TX)||(rf_irq & RF24_IRQ_TXFAILED)) && (msprf24_current_state()!=RF24_STATE_PRX) command is true, this means that there is a transmission interrupt from the transceiver unit about the report of the transmission. If the command $rf\_irq \& RF24\_IRQ\_TX$ is true, this means successful transmission. If the command $rf\_irq \& RF24\_IRQ\_TX$ is false, then MCU checks the other condition which is the last possible case that command $rf\_irq \& RF24\_IRQ\_TXFAILED$ is true or not. If it is true, the general interrupt register is cleaned by using $msprf24\_irq\_clear(rf\_irq)$ and then MCU waits time directly proportional with its node id in order to avoid collisions with other sensors. After that MCU goes to retransmission. If it is false, the general interrupt register is cleaned by using $msprf24\_irq\_clear(rf\_irq)$.

2.4. SENSING UNIT (LDR)

This part is about the sensing element which measures the interested attribute of the medium. The experiments with light are much more easier than the other elements such as gas, molecules, etc, because distribution of light intensity in the medium can be observed easily with human eyes. Therefore, LDR shown in Fig. 2.12 is used for measuring the light intensity and it is more suitable in order to sense the light intensity at the specific sensor node point as

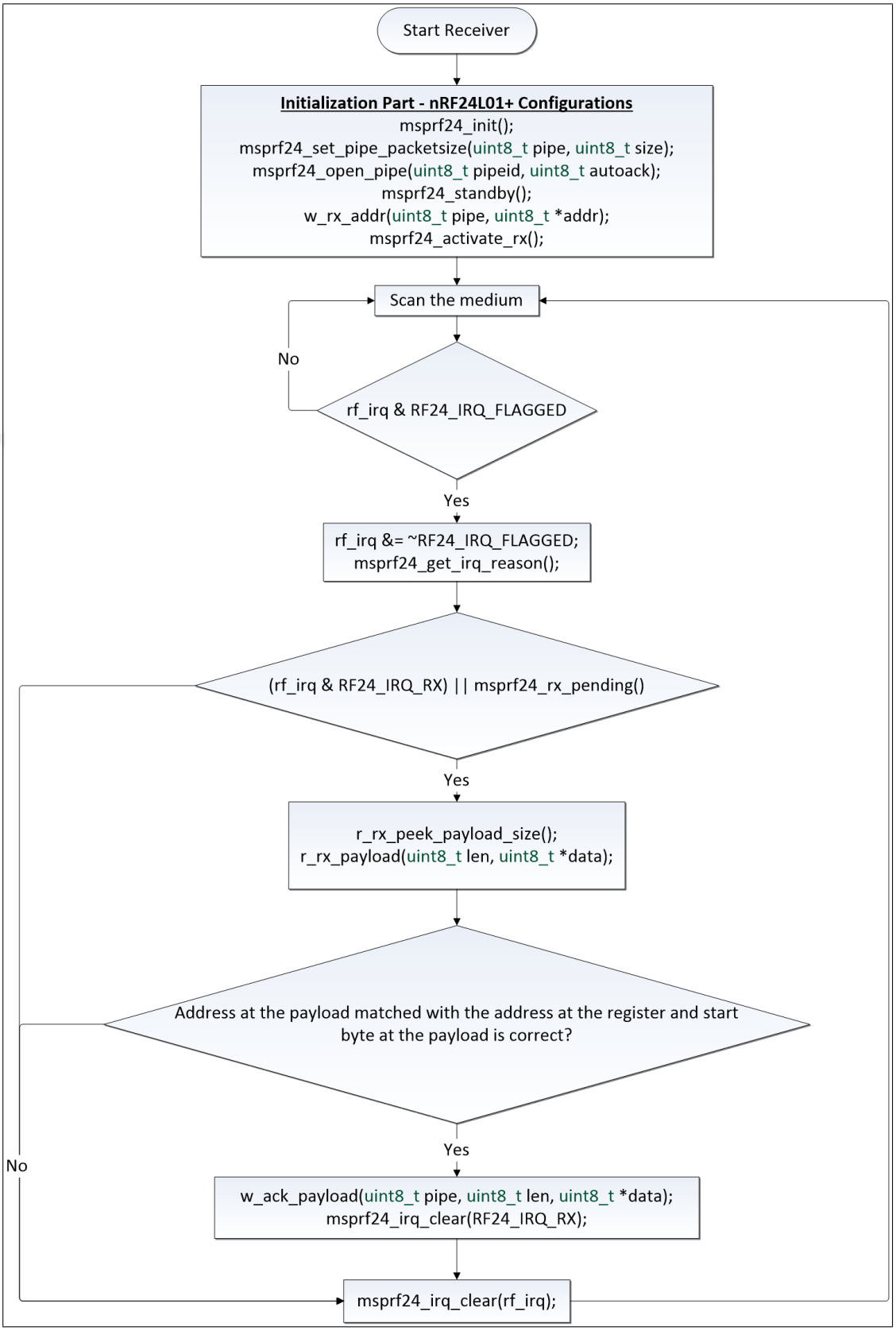


Figure 2.10. Simple Receiving Operation Among Peripheral Devices

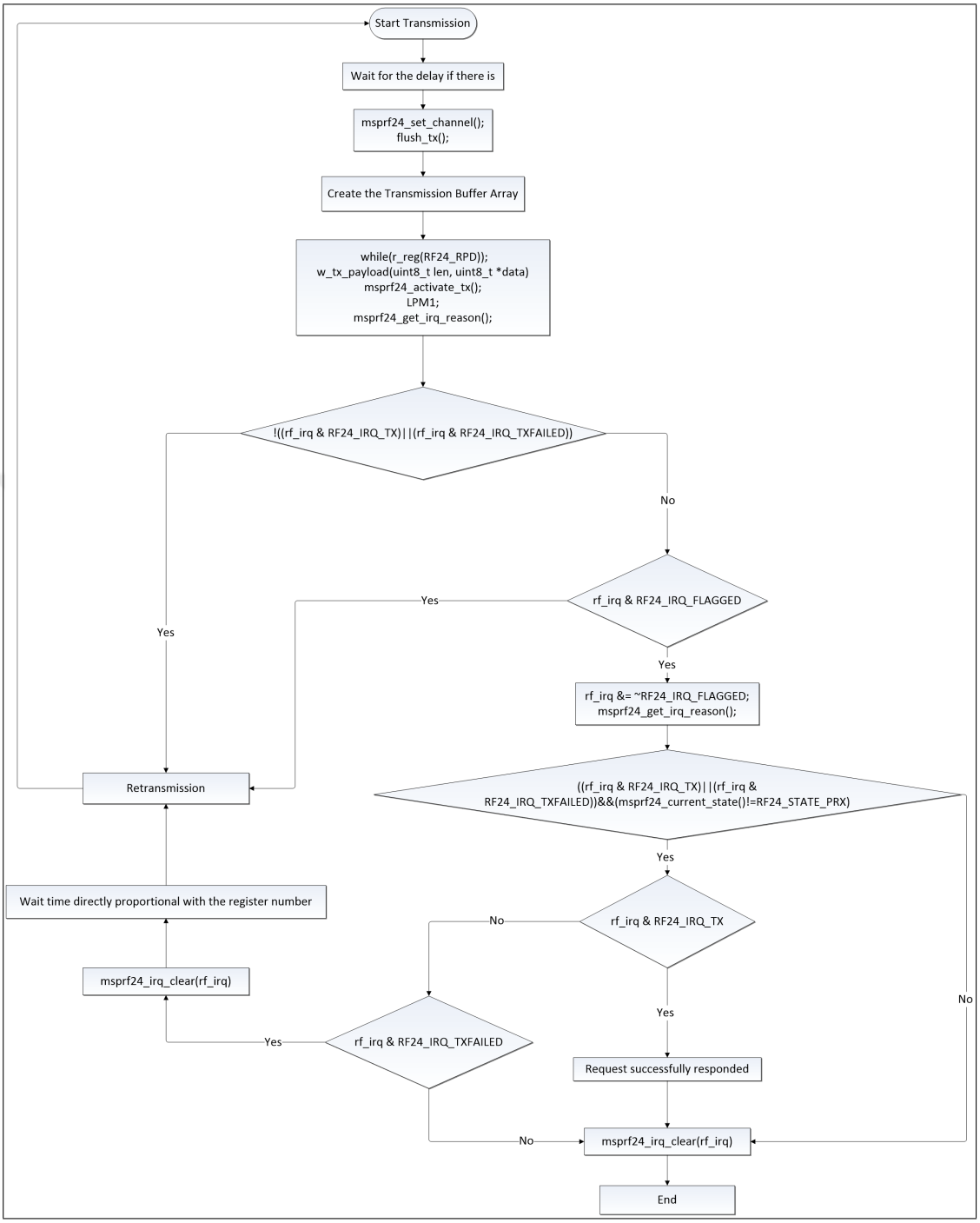


Figure 2.11. Simple Transmission Operation Among Peripheral Devices

compared to other types of sensing devices.

Sensory device LDR is connected to the one of the 8 ADC pins of the MCU. As seen in Fig. 2.5, Pin 1.4 of the MCU is used in this work. Temperature and voltage values are also obtained in 10 bits resolution with the help of ADC. In other words, data are measured with 10 bits by using ADC. The average value of the 20 LDR measurements is used as the final quantity to prevent from instant measurement errors and minimize the effect of outlier measurements. Since payload bytes of the nRF24L01+ consist of 8 bits, in this work, 10 bits ADC is quantized to 8 bits data by removing the least significant 2 bits of the ADC value. LDR and its calibration resistor are connected in series as shown in Fig. 2.5

It is considered that the output light intensity levels may be different with two different sensors even if they are at the same location under the same light conditions. Therefore, a calibration procedure is needed in order to achieve and have good solutions in terms of reliability and accuracy. After the calibration part is done, the outputs of sensory device become pretty close to each other at a certain light intensity where the measurements are taken at the same location.

2.4.1. Calibration

Firstly, an LDR is placed with potentiometer in series under a specific amount of light intensity to determine the reference output in terms of voltage. We form the circuit diagram shown in Fig. 2.13 for the calibration process. It should be noted that as the amount of light on the LDR increases, the output voltage also increases, too. This direct proportionality is related with the connection among LDR, potentiometer and power supply. Actually, there is a basic voltage division principle between LDR and calibration resistor. It is remembered that, LDR and its calibration resistor are connected in series and the positive terminal of the power supply is connected to a terminal of LDR where there is no node connection with LDR and its specific resistor to have direct proportionality. When the Vin and Ground connections are done in reverse, now, there is an inverse relationship between light intensity and output voltage which results in increased light intensity with decreased output level.

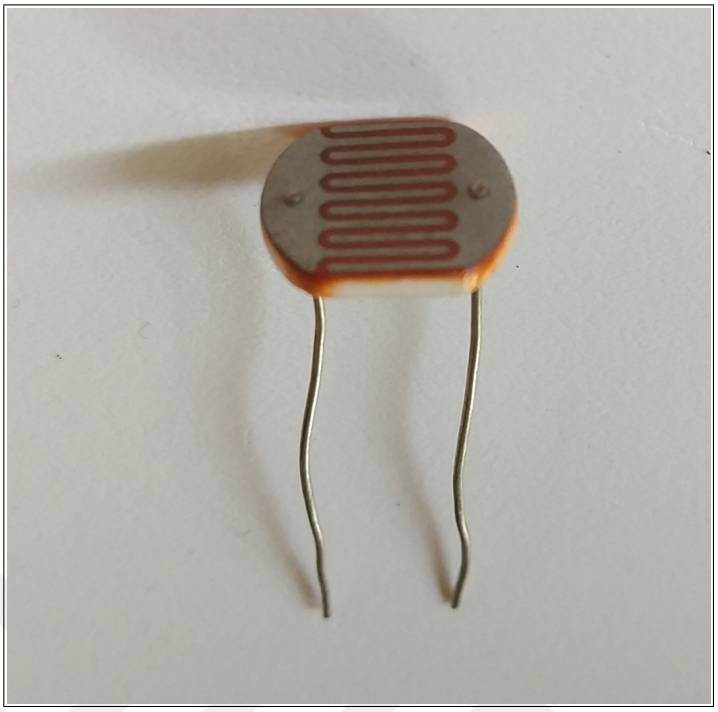


Figure 2.12. LDR (Light Dependent Resistor) Sensing Unit

After calibration circuit is ready to work, the input voltage level Vcc is adjusted to 2.8V as a reference input-supply voltage. Before energizing the circuit, be sure that Vcc and Ground (GND) connections are correct to have directly proportional result with the input light intensity where the positive side of the power supply is connected to one of the LDR pins which has not a node connection between LDR and potentiometer. Afterwards, the remained unconnected pin of the potentiometer is connected to the negative terminal of the power supply that is GND. Now, the circuit is ready to specify the reference output level (Vout) in terms of voltage at a specified light intensity level. The reference output level can be adjusted to the value around 1.5 Volts by spinning the potentiometer adjustment stick. When the desired output level is reached, the resistance of the potentiometer which is in use is measured to know how much calibration resistor is needed for the reference LDR, and the specific calibration resistor of the LDR under test is found at the end of measuring the calibration resistor value.

Secondly, other remained LDRs are located under the same amount of light intensity. After that, the same procedure is done to find the specific calibration resistor for each LDR by adjusting the output to the desired level which is identified in the first step.

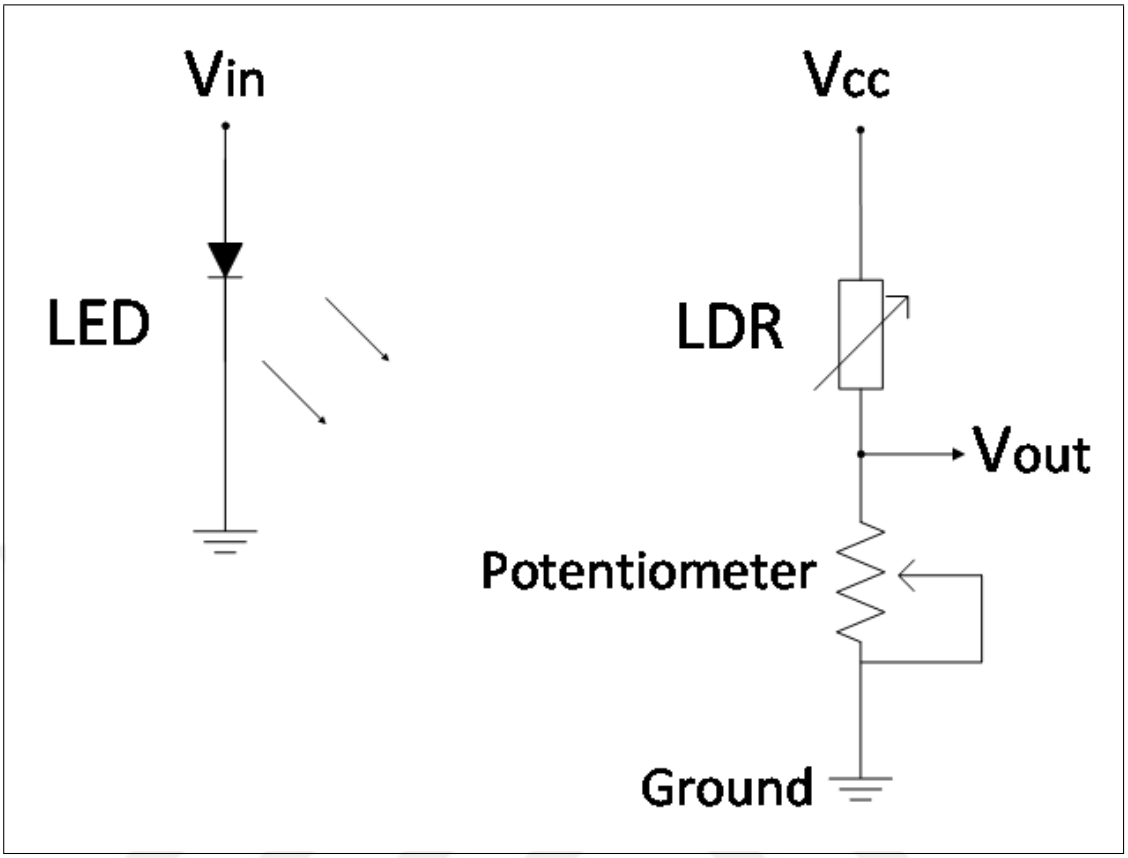


Figure 2.13. Calibration Circuit Diagram

Thirdly, the closest resistors in practice to the experimental calibration resistors is chosen to get and approach the most accurate result. In Table 2.8, required resistor values for each LDR are listed after calibration part is done. The closest resistors can be the combination of two or more resistors which are connected in series or parallel as it can be consisted of one resistor. It is remembered that the exact values of the two same resistors can be different from each other. For instance, when two different 1KΩ is measured with multimeter, the exact resistances are read as about 0.97KΩ and 1.07KΩ,respectively. The variation is directly related with tolerances of the resistors. Therefore, this feature of the resistors must be taken into account. It is important that, in this project, the output levels of the LDRs are set about 0.3V which is enough for hardware (HW) calibration under a specific light intensity. In addition, the exact values of the resistors used are found by measuring with multimeter . In order to approximate the required resistors obtained in the calibration stage, two resistors are used sometimes to minimize the difference between required and used resistors. Then, the exact values of the resistors in result are obtained.

Table 2.8. Calibration Resistors Measured at 26.5^oC

Node Number	**Calibration Resistor Required**	**Exact Value of Resistor 1**	**Exact Value of Resistor 2**	**Connection Type**	**Exact Equivalent Resistor Measured**
1	1.068k	2.195k	2.200k	Parallel	1.103k
2	0.678k	0.690k	-	-	0.690k
3	0.736k	1.485k	1.466k	Parallel	0.742
4	0.643k	0.671k	-	-	0.671k
5	0.699k	0.693k	-	-	0.693k
6	1.414k	0.228k	1.211k	Serial	1.438k
7	1.024k	1.019k	-	-	1.019k
8	0.809k	0.804k	-	-	0.804k
9	0.863k	0.835k	-	-	0.835k
10	1.025k	1.006k	-	-	1.006k
11	0.691k	0.696k	-	-	0.696k

The experimental setup which observes the relationship between Vin and Vout, is shown in Fig. 2.14. The input energizing voltage level (Vcc) is adjusted to the 2.8V and then a 1W power Light Emitting Diode (LED) light source is located at the same axis and direction with the LDR. Therefore, LED and LDR are face with each other. After that, the distance between LDR and LED is set as 10 cm and another power supply is connected to the 1W power LED light source. When the system is ready to collect data from the LDR, the supplied voltage of the LED (Vin) is increased from 2V to 4V gradually. Gradual increment is necessary in order not to miss the output voltage corresponding the input voltage at that interval. While the input voltage of the 1W power LED is increasing, that means brightness of the LED increases, input voltage and the output voltage level of the LDR is captured and recorded with the help of ”Cassy Lab” Software (SW) Product at the same time.

It is considered that, input voltage (Vin) is started from 2V since LEDs behaviour are so similar as diodes. In other words, it works after 2V and gives light to the ambient. In addition, while the input voltage of LED increases which means that the brightness of the ambient light

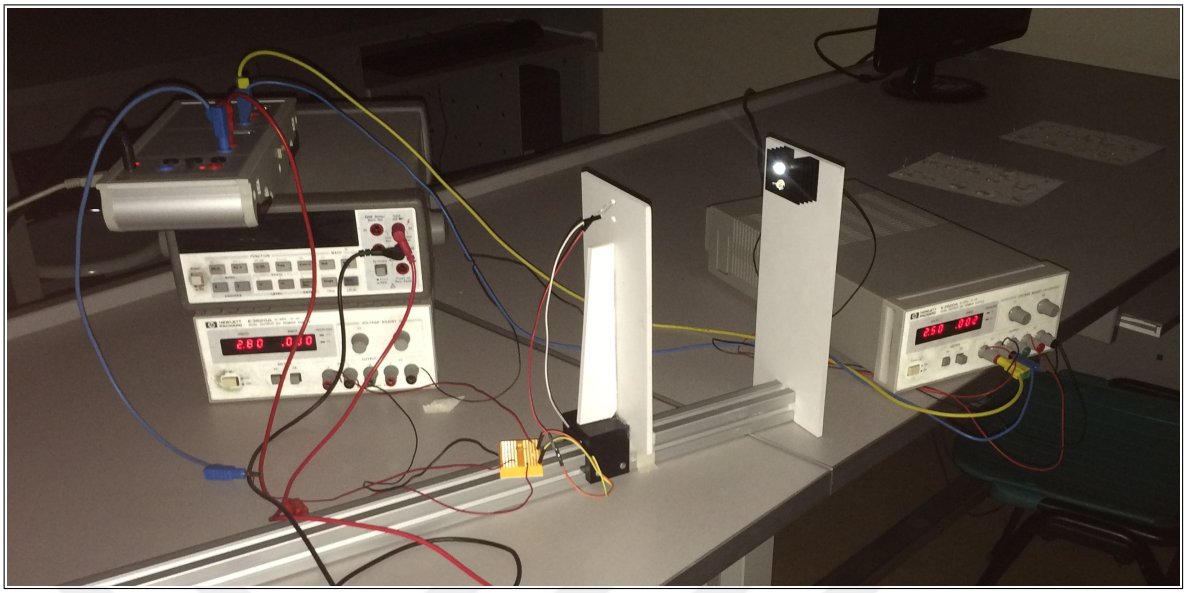


Figure 2.14. An Image From Calibration Using Cassy Lab

will also increase, the output voltage at the sensor as LDR also increases although the power supply voltage of the sensor LDR is still constant as 2.8V.

The graph about the characteristics of the LDRs (the output voltages of LDRs vs the input voltages of LDRs) after calibration process is shown in Fig. 2.15. As seen on the graph, the responses of the LDRs from dark to light ambient are so close to each other after the calibration done. Therefore, all LDRs give close output voltages to each other until around 3 Volts (Vin) under the same light intensity. In our work, the output voltage (Vout) varies from 0V to 1.6V for the test area conditions. It is easy to say that, all sensory device characteristics are similar to each other at the operating output range.

2.5. ENERGY CONSUMPTION OF THE COMPONENTS

The power spent at each component is roughly shown in Table 2.9. In Table 2.9, the average consumptions of each unit is given based on the supply voltage at 2.75V. It is easy to see that significant energy is consumed by the transceiver unit while the microcontroller energy consumption is the least. Also, LDR consumes power depending on the light intensity of the

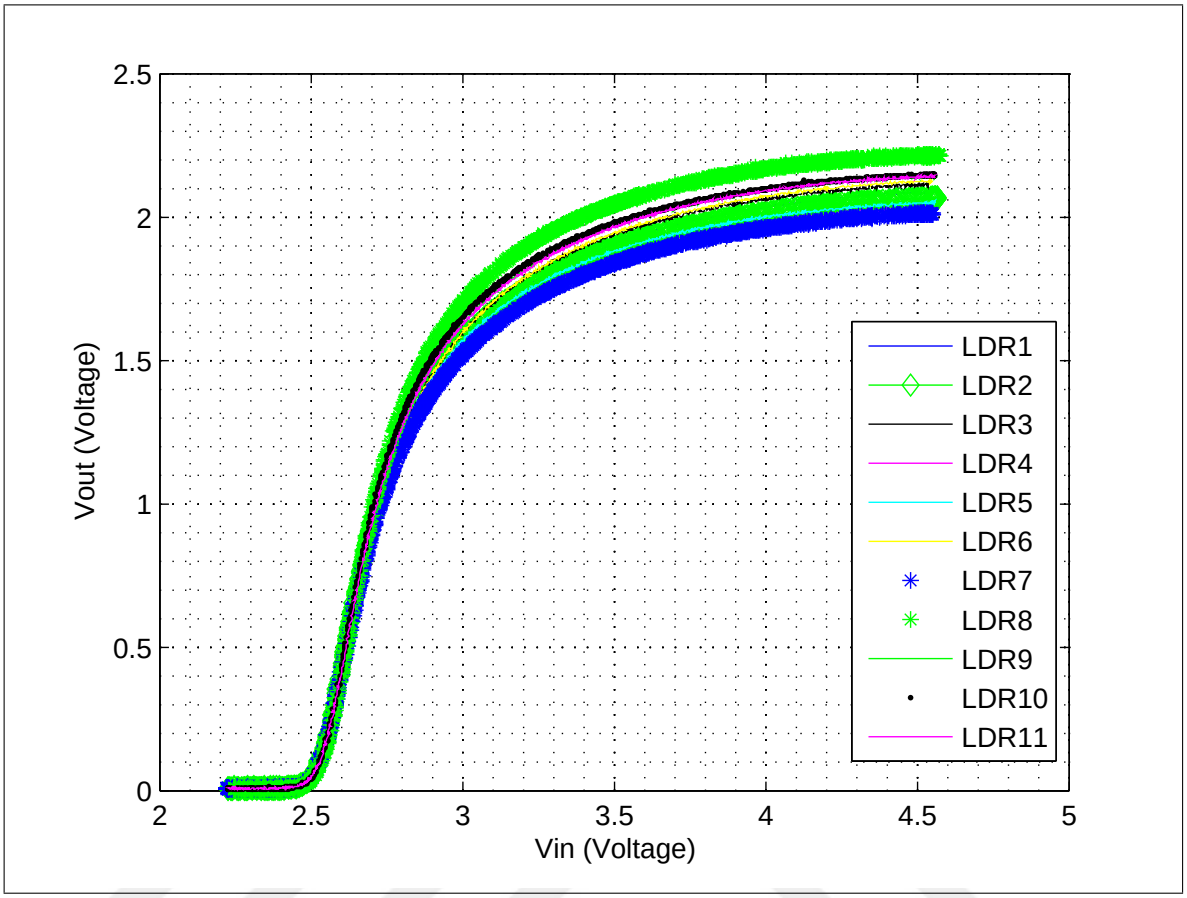


Figure 2.15. Characteristics of the LDRs

medium so, the energy spent by LDR is directly proportional with the light intensity.

2.6. COST OF A SENSOR NODE

The total cost of a sensor node including the price of each component is roughly shown in Table 2.10 below. A sensor node roughly costs around 12 USD's.

Table 2.9. Energy Consumption of Each Component at the Sensor Node

	Components				
	MSP430G2553	nRF24L01+	LDR	Overall System	
Description	Active Mode(AM)			Under Low Light	Under High Light
Current Consumption @2.75V	230μA	$\sim$14mA	0,15mA$\sim$2mA	**14,6 mA**	**16,4 mA**

Table 2.10. The Cost of Each Sensor Node

Number	**Description**	**Price (in $)**
1	MSP430G2553	2.8
2	nRF24L01+	0.99
3	Sensor (LDR)	0.35
4	Batteries (2 x AA 2100 mAh)	4.75
5	Mini Breadboard	0.71
6	Other Components (Cable, Resistor, Battery Box, etc.)	2
7	**General Total**	**11.60 $**

3. COMMUNICATION ALGORITHM OF THE WSN

In this chapter, we present the communication protocol used to connect sensors with the FC. As seen in Fig. 3.1, let sensor S_i is located at x_i where its measurement is represented as $Z(x_i)$. N sensors positioned around a FC to form the WSN. FC sends the sensor measurements collected from the WSN to PC. Here, FC consists of MSP430G2553, and nRF24L01+ which is directly connected to PC. There is a duplex communication via wire between FC and PC by using COM Port interface. There is a hardware UART converter at the MSP430G2553 LaunchPad. Therefore, microcontroller may communicate with PC by using the UART protocol with its launchpad. Remember that FC is connected to PC by UART and connected to nRF24L01+ via SPI. The data sent from FC by using UART is converted to the serial port (COM Port) data on the PC or vice versa. Thus, the communication interface between each other is provided. Then, MATLAB is used for statistical inference on the computer. Also, the locations of all sensor nodes are known in MATLAB in advance.

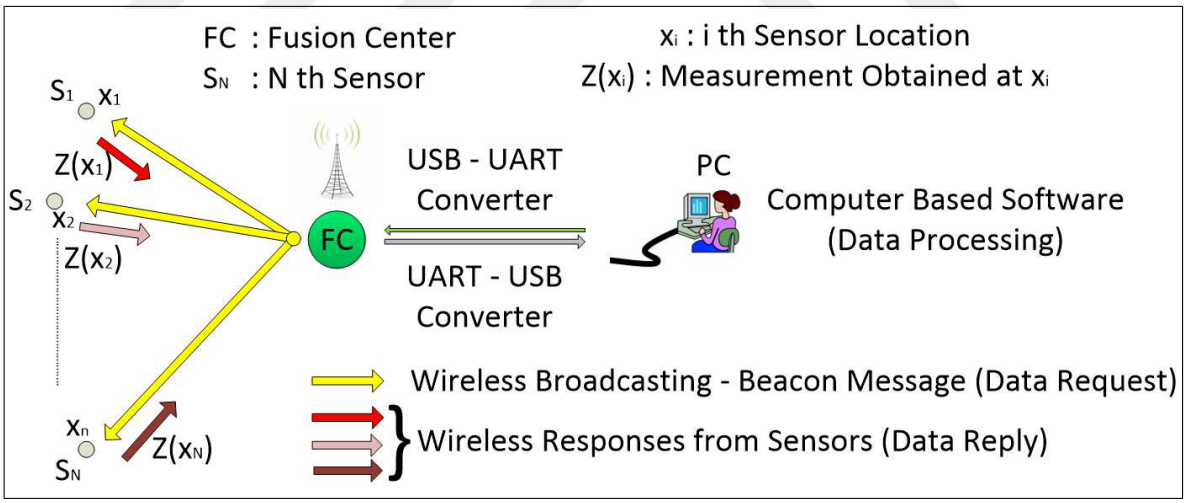


Figure 3.1. General Communication Chart

3.1. DATA REQUEST

The flowchart of the operations carried out at the FC microcontroller are shown in Fig. 3.2. In the recommended system, the collection of sensor measurements is started by the user on the

PC. Therefore, FC waits an initial request from MATLAB. When the FC receives a request from MATLAB, it gathers the sensors data by using the broadcasting protocol noticed by the PC. Generally, beacon message called as broadcast message is used in sensor networks. Here, FC makes a data request to the all sensor nodes by using a beacon message and waits the individual replies from all sensor nodes. When FC receives the sensor measurements or sensor data, it will send an acknowledgement to each sensor to notify that the sent packet is received successfully by FC. When fusion center takes all the data from all of the sensors or waits enough time for the request called timeout period, it sends the whole collected sensor data as a report to the MATLAB. Then, it clears the sensor data on its register to make available to another request reporting. After that, FC waits an another request from the MATLAB. Moreover, FC understands the packet owner that is the sensor node number by the information inside the received payload.

Let's begin with the packet which is sent by FC to the all sensor nodes. This packet says to all sensor nodes that FC requests data so, after receiving this packet, take a measurement at your location using your sensory device LDR and send them to the FC with the packet format shown in Table 3.1. Table 3.1 contains necessary information based on the FC registers. In other words, a packet is sent to all sensor nodes at a time and it includes fusion center register information. In addition, the packet information covers a start byte to check that this packet belongs to the system, if not, the request will not be taken into account and unanswered, a delivery address which indicates where the packet goes to, a sender address which indicates where this packet comes from, the time basics which are year, month, day, hour, minute, and second for syncing and be synchronous with fusion center.

After that, there is a protocol type knowledge which indicates that this is a broadcasting protocol communication between fusion center and sensor node, a threshold value to indicate that there is a data about sensory device LDR reading which is large enough for making a measurement or very small and little which may be ignorable, and a sleeping time in second to consume less energy at the sensor nodes. It is known that, in order to maximize the life of sensor node, the low power modes would be used to obtain a better battery life. Therefore, when there is no need to take data very often, microcontroller and other unnecessary equipments may be gone to sleep or shut down to save energy and increase the life. Actually,

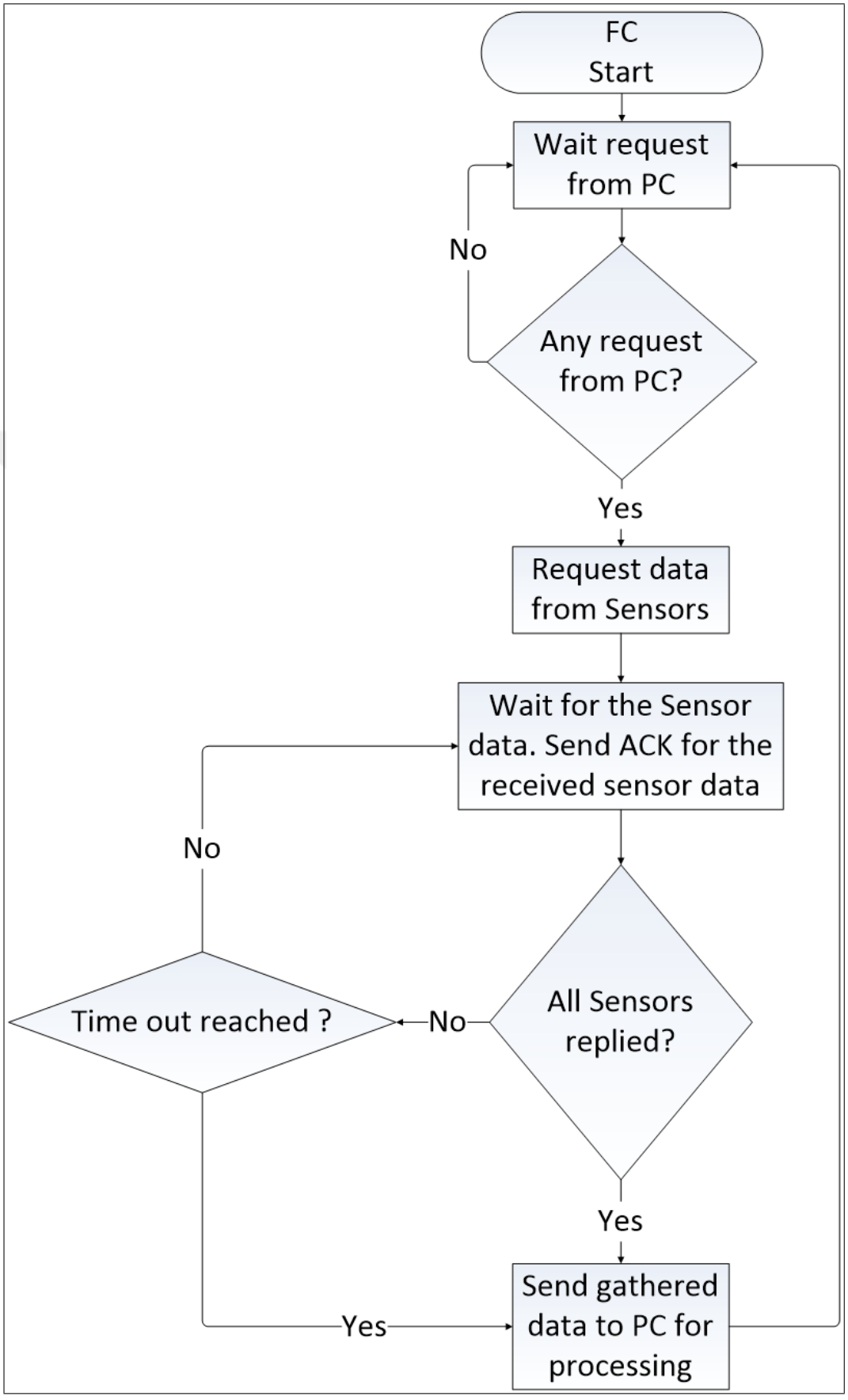


Figure 3.2. Process Flow Charts at FC

protocol type knowledge, sleeping time, and battery level are not used in this thesis, but they are the infrastructures for the future protocols.

Table 3.1. Payload Packet Format From FC To Sensor - Beacon Packet

	Description
Payload 1	Start Byte As 25 For This Network
Payload 2	Delivery Address As 250 For General Broadcasting Address
Payload 3	Sender Address As 0 FC Address
Payload 4	Year Taken From MATLAB Currently
Payload 5	Month Taken From MATLAB Currently
Payload 6	Day Taken From MATLAB Currently
Payload 7	Hour Taken From MATLAB Currently
Payload 8	Minute Taken From MATLAB Currently
Payload 9	Second Taken From MATLAB Currently
Payload 10	Protocol Type Taken From MATLAB As 1 For Parallel Communication
Payload 11	Threshold Value As 0 For This Network
Payload 12	Sleeping Time As 0 For This Network
Payload 13	Unused
Payload 14	Unused
Payload 15	Unused
Payload 16	Unused

By using only 6 pipes assigned to each sensor as described in Section 2.2.3, we are unable to implement a WSN which has more than 6 sensors. In order to connect numerous sensors with the FC, in this thesis we only utilize two of the pipes. Pipe 0 addresses of each sensor and the FC are used for data transmissions between sensors and FC, and Pipe 1 addresses of sensors and the FC are used for initial data requests from FC. In Fig. 3.3, we form $N = 11$ sensors and the FC where initial Pipe 0, Pipe1 and transmit addresses are shown. Pipe 0 addresses of devices are defined unique while Pipe 1 addresses are selected the same for all devices in the WSN. As an example, when the FC transmits a data request packet, the 5-bytes transmit address section of Fig. 2.7 is filled with the common Pipe 1 address 0xDEADBEEFC8 and the payload section is filled with Table 3.1. When a sensor receives the data request, it compares the transmit address with the addresses defined in its pipes. Since the transmit address matches

with its Pipe 1 address, the sensor enters its data reply routine, i.e., obtaining measurement and preparing its data packet according to Table 3.2.

As previously shown in Fig. 2.6, in order to receive ACK from FC, the TX address and the RX address at a given pipe should be identical. In our setting, the Pipe 0 addresses of sensors and the FC are selected as different. Upon transmitting the data packet to the FC, a sensor immediately updates its Pipe 0 address as the Pipe 0 address of the FC. As an example, Sensor 1 has initial Pipe 0 address 0xDEADBEEF01. Upon transmitting its data packet to the FC, it updates it Pipe 0 address as the Pipe 0 address of the FC, that is the Pipe 0 address of Sensor 1 is changed to 0xDEADBEEF00. By doing so, now Sensor 1 is able to capture the ACK message corresponding to its transmission. Since the rest of the sensors will have different Pipe 0 addresses at that time instant, the ACK of Sensor 1's transmission will be received only by Sensor 1. In this work, we use ACK with payload, that is each sensor receives an identical copy of its previously transmitted packet. Upon the reception of the ACK, the sensor updates its Pipe 0 address to its initial value and waits for another data request from the FC. As an example, upon receiving the ACK, Sensor 1 sets its Pipe 0 address back to 0xDEADBEEF01.

nRF24L01+ receives data when incoming packet address in transmission is matched with one of its own pipe addresses, then it notifies this situation to its microcontroller. After that, the MCU checks the second byte (delivery id) given in Table 3.1. If the delivery node id matches with the FC's own id located at its registers, MCU sends an ACK with payload command to the nRF24L01+. If not, MCU ignores the received package and continues to listen the air for future packages.

It is also possible that two sensors send their packet to the Pipe0 address of the FC at the same time. In this case, both sensors change their Pipe0 address as the Pipe0 address of the FC 0xDEADBEEF00 and wait ACK. If there is a collision, and they both do not receive any ACKs, they both retransmit their packets as described in Section 3.2. On the other hand, when FC receives one of the packets from its Pipe 0, it sends an ACK with the contents of the received payload data. In this case, both sensors receive the ACK, on the other hand, the sensor whose the transmitted payload data matches with the content of the received ACK, understands that it successfully transmitted its packet. However, the other sensor whose

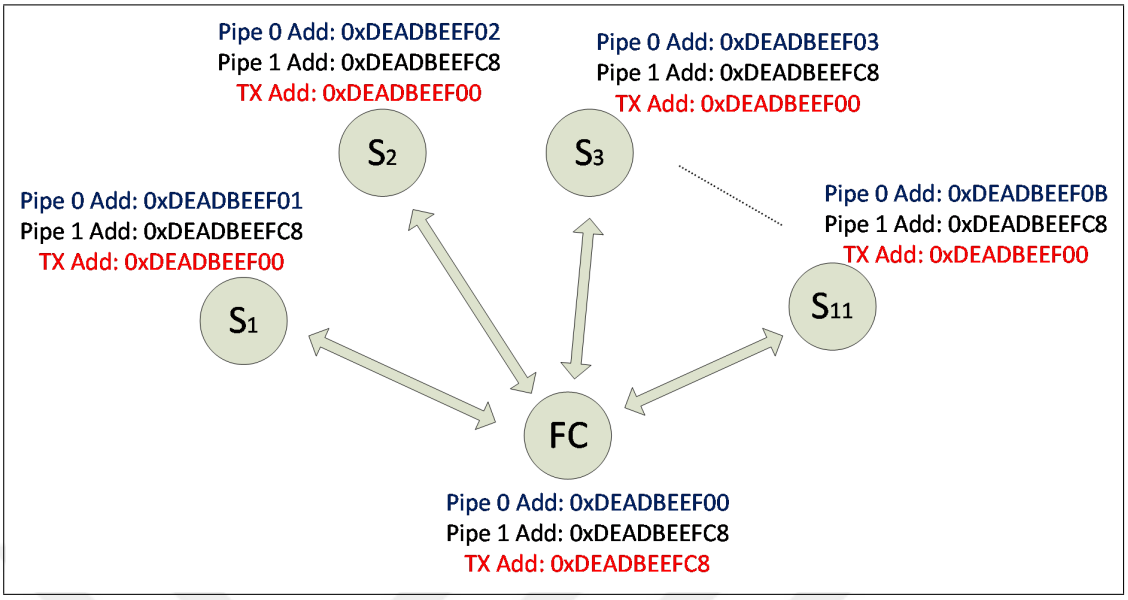


Figure 3.3. Transceiver Unit Addressing For Each Sensor

transmitted payload data does not match with the payload received in the ACK, understands a failure in transmission and performs a retransmission.

3.2. DATA REPLY

Working chart of the sensor microcontroller is shown in Fig. 3.4. A sensor listens the medium until it receives a request from its Pipe 1. When the sensor node notices this request, it takes measurements using its LDR and prepares the payload presented in Table 3.2. The data packet is sent to Pipe 0 address of FC. The packet format which sends from sensor nodes to the fusion center includes start byte, delivery address, sender address, time basics, protocol type, threshold value, sleeping time, LDR sensor measurement, temperature sensor measurement, and information of battery level measurement. In other words, the sensor node sends its own registers first which are received from the FC in the requesting message. These are sent back to the fusion center because they are all important and say that the packet is prepared based on the information received. Then, the sensor node adds its own sensor measurements to the package. Finally, the whole packet is sent to the fusion center.

Here, we develop the medium access control algorithm based on CSMA. In other words, sensors first check the activity in the air. If they conclude an absence of activity, they transmit. Received Power Detector (RPD) is an equivalent function of Received Signal Strength Indicator (RSSI) and used in the RX mode as a property of the nRF24L01+ in order to conclude whether there is any channel activity on the air. It should be remembered that, the register in nRF24L01+ is not very developed so, it can only detect that whether the received signal power is higher than -64dBm or not at the operating frequency channel. If the signal power is greater that -64dBm, RPD register is set high with 40μs delay. Therefore, the nRF24L01+ of each sensor listens the medium first and if there is no channel activity, it sends its package. However, if it detects an activity, it waits for two maximum packet durations, where a packet duration is computed based on the packet format presented in Table 2.5 with full payload size 32 bytes for the worst case scenario. Upon the end of this period, the sensor again checks the medium, if there is no channel activity, it sends its packet. Otherwise, it keeps waiting for another two maximum packet durations.

Upon the transmission of the packet to the FC, the nRF24L01+ of the sensor waits at most 4000 μs for the reception of the respective ACK. If no ACK is received within this period, the packet is automatically retransmitted. In this work, each data packet from sensors to FC is transmitted at most 10 times. After the completion of the 10th transmission, if the sensor has not received an ACK from the FC yet, it concludes a transmission failure. Failures may occur as a result of the collisions which occurs as a result of the simultaneous transmissions of several sensors to the FC. If a sensor realizes such a transmission failure, it first waits for a duration proportional to its node ID. This aims to prevent simultaneous transmissions of the previously failed packets. As an example, Sensor 1 waits less time and Sensor 11 waits more time before retransmission. At the end of this period, each sensor updates their packet content with new measurements and then checks the medium. If the medium is clear, the sensor transmits its packet to the FC. If the medium is occupied, sensor waits for two maximum packet durations as described above.

It is also possible that the FC may receive the packet of particular sensor, but the corresponding ACK might not be received by the sensor. In this case, the sensor concludes a transmission failure and executes the process described above. If the FC receives multiple packets from

the same sensor, it overwrites the packets received from this sensor. In other words, the last successfully received packet coming from the sensor will be stored in the FC's respective register.

Finally, if a sensor sends its packet successfully to the FC, i.e., it receives the ACK of the transmitted packet, it restarts its hardware by using watchdog timer (WDT) in order to prevent an infinite loop at the sensor. Afterwards, the sensor goes to the listening mode and keeps checking its Pipe 1 to detect another request from the FC.

Table 3.2. Payload Packet Format From Sensor To FC

	Description
Payload 1	Start Byte As 25 For This Network
Payload 2	Delivery Address As 0 For FC
Payload 3	Sender Address As Specific Address Number For Each S
Payload 4	Year Taken From Sensor Register Currently
Payload 5	Month Taken From Sensor Register Currently
Payload 6	Day Taken From Sensor Register Currently
Payload 7	Hour Taken From Sensor Register Currently
Payload 8	Minute Taken From Sensor Register Currently
Payload 9	Second Taken From Sensor Register Currently
Payload 10	Protocol Type Taken From FC As 1 For Parallel Communication
Payload 11	Threshold Value As 0 For This Network
Payload 12	Sleeping Time As 0 For This Network
Payload 13	Sensory Device LDR Measurement
Payload 14	Internal Temperature Sensor Measurement
Payload 15	Battery Measurement
Payload 16	Unused

3.3. DATA EXCHANGE BETWEEN FC AND PC

There is also a communication protocol between PC and the FC. PC manages the all system using MATLAB so, the user enter the protocol type, time difference between two data gathering requests, and total number of data gatherings at the beginning. Upon executing

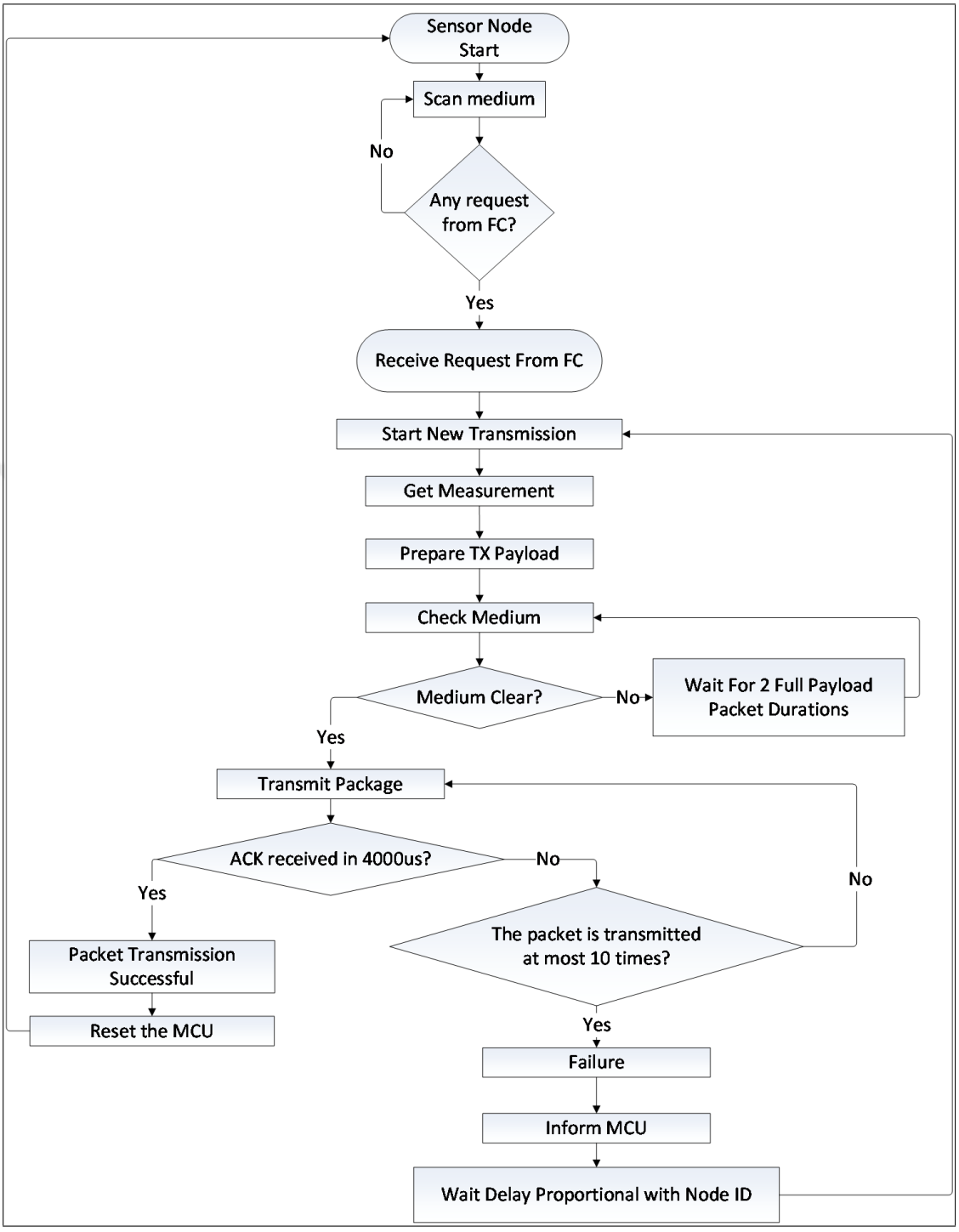


Figure 3.4. Process Flow Charts at Sensors

the script on MATLAB, MATLAB sends the data gathering request to the FC by sending a the packet defined in Table 3.3. The packet includes start byte for MATLAB side, year, month, day, hour, minute, second, protocol type, threshold value, and sleeping second. It is set that the start byte for MATLAB side is different from start byte with NRF24L01+ data transmission communication. The time basics which include year, month, day, hour, minute, and second are kept up to date since MATLAB takes the time from real time clock of the PC. Therefore, all of the system including the FC and the sensors are in sync the time of PC when they receive new packets presented in Table 3.3 and Table 3.1, respectively.

Then, MATLAB waits a reply from the FC. After all sensor data are collected at the FC, FC delivers the available WSN data with a packet format defined in Table 3.4 which is the last and final packet type. The packet includes the followings for each sensor node separately; sensor location number, year, month, day, hour, minute, second, protocol type, threshold value, sleeping time, LDR sensor measurement, temperature sensor measurement, information of battery level measurement, respectively. Note that the time contents of the received packets are updated by the timers of each sensor.

After the FC collects and gathers the sensor packets, it sends the all WSN data to PC via COM Port of the computer by using UART protocol. Then, on the PC, MATLAB executes the field estimation which will be explained in Chapter 4 in detail. When the results are ready, MATLAB saves them along with the available WSN measurements. Then, the MATLAB script waits for a certain duration before initializing another request from the FC. The MATLAB script continues until it reaches the maximum number of data gatherings. After that, the script finishes and results are saved on the PC.

Table 3.3. Packet Byte From PC To FC

	Description
Payload 1	Start Byte As 23 For This Network
Payload 2	Year Taken From Computer Currently
Payload 3	Month Taken From Computer Currently
Payload 4	Day Taken From Computer Currently
Payload 5	Hour Taken From Computer Currently
Payload 6	Minute Taken From Computer Currently
Payload 7	Second Taken From Computer Currently
Payload 8	Protocol Type Given To FC As 1 For Parallel Communication
Payload 9	Threshold Value As 0 For This Network
Payload 10	Sleeping Time As 0 For This Network

Table 3.4. Packet Byte From FC To PC

	Description
	(do i=1:Total Number Of Sensor Nodes)
Payload 1	i th Sensor Node Number
Payload 2	i th Year Taken From S Register
Payload 3	i th Month Taken From S Register
Payload 4	i th Day Taken From S Register
Payload 5	i th Hour Taken From S Register
Payload 6	i th Minute Taken From S Register
Payload 7	i th Second Taken From S Register
Payload 8	i th Protocol Type Given To FC As 1 For Parallel Communication
Payload 9	i th Threshold Value As 0 For This Network
Payload 10	i th Sleeping Time As 0 For This Network
Payload 11	i th LDR Sensor Measurement
Payload 12	i th Internal Temperature Sensory Device LDR Measurement
Payload 13	i th Battery Measurement

4. FIELD ESTIMATION

After the FC gathers data from all the sensors in the WSN, it forwards the collected data to the PC where field estimation is performed on MATLAB. In this section, we briefly review the fundamental properties of the function called variogram and the method of ordinary kriging which are both used to estimate the field intensity at an unobserved location using the data collected from known sensor locations [6, 7].

4.1. VARIOGRAM

The variogram function measures the spatial relation of the random process $Z(\mathbf{x})$ over the field of interest. In other words, it measures how two different measurements, $Z(\mathbf{x}_i)$ and $Z(\mathbf{x}_j)$ observed at locations $\mathbf{x}_i$ and $\mathbf{x}_j$ influence each other. Before introducing the formal definition of the variogram, we assume that the field of interest is second order stationary, which means

- The mean of the field is constant $E[Z(\mathbf{x})] = \mu$.
- The covariance function between the observations collected from two locations in the field, depends only on the distance between these two locations as,

$$Cov(\mathbf{x}_i, \mathbf{x}_j) = Cov(\mathbf{h}) \tag{4.1}$$

  where $\mathbf{h} = |\mathbf{x}_i - \mathbf{x}_j|$ is the distance between $\mathbf{x}_i$ and $\mathbf{x}_j$.

Variogram, $\gamma(\mathbf{h})$ then measures the variance of the increments between the measurements of two locations. Let $\mathbf{x}_i = \mathbf{x} + \mathbf{h}$ and $\mathbf{x}_j = \mathbf{x}$, then

$$\gamma(\mathbf{h}) = \frac{Var(Z(\mathbf{x} + \mathbf{h}) - Z(\mathbf{x}))}{2} \tag{4.2}$$

which represents the average dissimilarity between the observations obtained from two locations separated by distance $\mathbf{h}$. Variogram can also be written in terms of the covariance of the random field as,

$$\gamma(\mathbf{h}) = \frac{Var(Z(\mathbf{x}+\mathbf{h}) - Z(\mathbf{x}))}{2} \tag{4.3}$$

$$= \frac{E[(Z(\mathbf{x}+\mathbf{h}) - Z(\mathbf{x}))^2] - (E[(Z(\mathbf{x}+\mathbf{h}) - Z(\mathbf{x}))])^2}{2}$$

$$= \frac{E[(Z(\mathbf{x}+\mathbf{h}))^2] - 2E[Z(\mathbf{x}+\mathbf{h})Z(\mathbf{x})] + E[(Z(\mathbf{x}))^2]}{2}$$

$$= E[(Z(\mathbf{x}))^2] - E[Z(\mathbf{x}+\mathbf{h})Z(\mathbf{x})]$$

$$= Cov(0) - Cov(\mathbf{h})$$

Then, variogram satisfies the following properties,

- $\gamma(0) = 0$
- $\gamma(\mathbf{h}) \geq 0$
- $\gamma(-\mathbf{h}) = \gamma(\mathbf{h})$

4.1.1. Experimental Variogram

In practice the real variogram may be unknown and needs to be estimated prior to the field estimation. The variogram can be estimated from a sample of data collected from certain locations of the field. Let a pair of sensor measurements obtained from the field are represented with $z(\mathbf{x}_i), \ldots, z(\mathbf{x}_j)$. Then the squared difference between the observed values is defined as,

$$\gamma^*_{i,j} = \frac{(z(\mathbf{x}_i) - z(\mathbf{x}_j))^2}{2} \tag{4.4}$$

For the entire field, the experimental variogram then yields a measure of dependence as,

$$\gamma^*(\mathbf{h}) = \frac{1}{2N(\mathbf{h})} \sum_{N(\mathbf{h})} (z(\mathbf{x}_i) - z(\mathbf{x}_j))^2 \tag{4.5}$$

where $N(\mathbf{h})$ represents the number of sensor pairs whose distances are $\mathbf{h}$, i.e, $N(\mathbf{h}) = \{\mathbf{h} = \mathbf{x}_i - \mathbf{x}_j, i, j \in 1, \ldots, N\}$.

4.1.2. Variogram Models

The variogram models introduced below are used for fitting to the experimental variogram. Depending on the application, one of the models may best reflect the nature of the field.

- Bounded Linear Model:

$$\gamma(\mathbf{h}) = \begin{cases} b\left(\frac{|\mathbf{h}|}{a}\right) & 0 \leq |\mathbf{h}| \leq a \\ b & elsewhere \end{cases} \tag{4.6}$$

- Spherical Model

$$\gamma(\mathbf{h}) = \begin{cases} b\left[\frac{3}{2}\frac{|\mathbf{h}|}{a} - \frac{1}{2}\left(\frac{|\mathbf{h}|}{a}\right)^3\right] & 0 \leq |\mathbf{h}| \leq a \\ b & elsewhere \end{cases} \tag{4.7}$$

- Exponential Model

$$\gamma(\mathbf{h}) = \; b\left[1 - \exp\left(-\frac{|\mathbf{h}|}{a}\right)\right] \quad |\mathbf{h}| \geq 0 \tag{4.8}$$

- Gaussian Model

$$\gamma(\mathbf{h}) = \; b\left[1 - \exp\left(-\frac{|\mathbf{h}|^2}{a^2}\right)\right] \quad |\mathbf{h}| \geq 0 \tag{4.9}$$

- Matérn Model

$$\gamma(\mathbf{h}) = \; b\left[1 - \frac{1}{2^{v-1}\Gamma(v)}\left(\frac{|\mathbf{h}|}{a}\right)^{v} I_v\left(\frac{|\mathbf{h}|}{a}\right)\right] \quad |\mathbf{h}| \geq 0 \tag{4.10}$$

where v is the smoothness parameter, $v \in [0, \infty)$, $\Gamma(.)$ is the gamma function and $I_v(.)$ is the modified Bessel function of order v.

Fig. 4.1 illustrates the parametric variogram models listed above for $a = b = 1$ as a function of distance between two observation locations, $\mathbf{h}$. Additional variogram models can also be found in [6]. In this thesis, we first obtain the experimental variogram using the collected sensor data at the FC. Then, we fit the best variogram model to our experimental variogram and obtain the set of parameters a, b, and v (For the Matérn Model only) which minimize the error between the experimental variogram $\gamma^*(\mathbf{h})$ and the selected variogram model $\gamma(\mathbf{h})$ as,

$$\min_{a,b,v} \sum_{\mathbf{h}} \left(\gamma^*(\mathbf{h}) - \gamma(\mathbf{h})\right)^2 \tag{4.11}$$

4.2. ORDINARY KRIGING

Kriging is a specific name which originates from the Geostatics terminology where Kriging is an interpolation technique where the aim is to perform surface mapping and estimate the field intensity at any location using limited data. In this thesis, we use the Ordinary Kriging (OK) method in order to estimate the field intensity at an interested location [6, 7].

Assume that the field is sampled by N sensors at known locations $\mathbf{x}_i$ where $i \in \{1, 2, \ldots, N\}$. Let each sensor measurement observed at location $\mathbf{x}_i$ be represented as $Z(\mathbf{x}_i)$ Then the OK estimate of the field intensity at a certain location $\mathbf{x}_0$, $\hat{Z}(\mathbf{x}_0)$ is obtained by taking the linear combination of sensor measurements as,

$$\hat{Z}(\mathbf{x}_0) = \sum_{i=1}^{N} \lambda_i Z(\mathbf{x}_i) \tag{4.12}$$

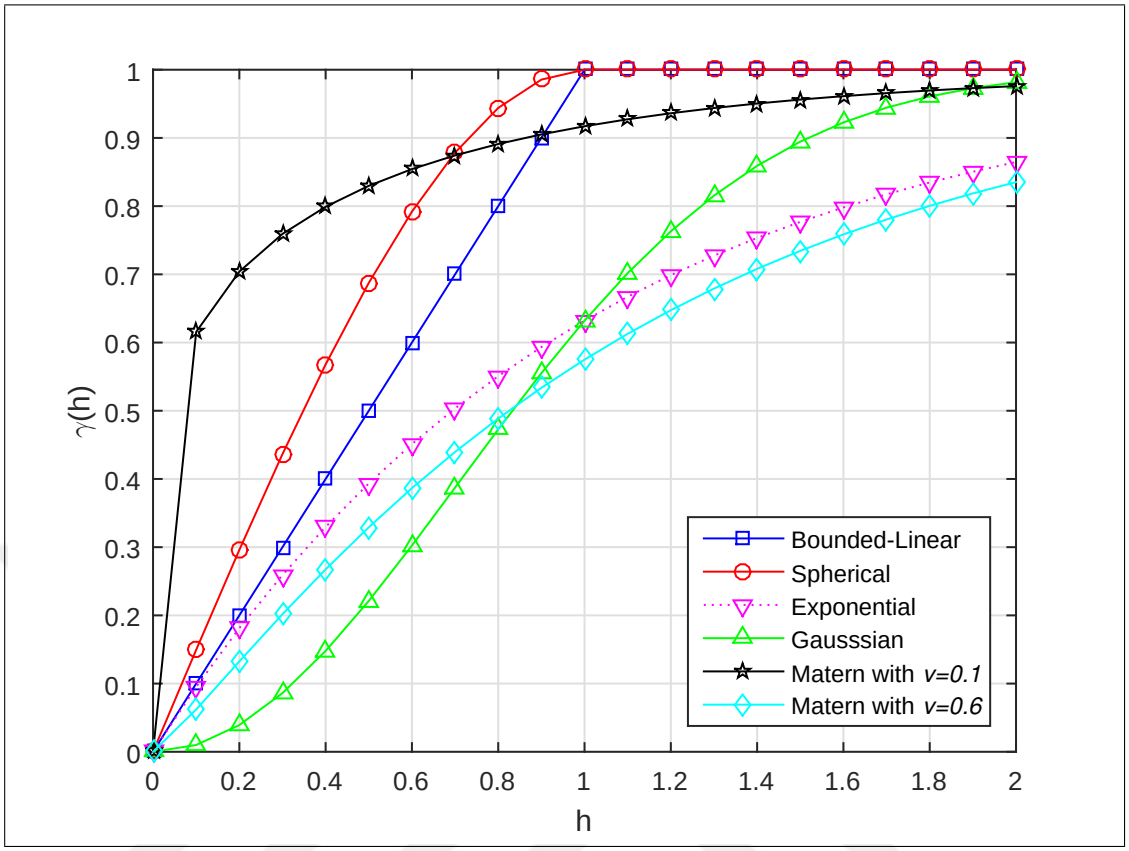


Figure 4.1. Variogram models under $a = b = 1$.

where λ_i represents the unknown weights.

In this section, we follow the formulations and notations presented in [7]. In order to ensure that the OK estimator is unbiased, i.e.,

$$E[\hat{Z}(\mathbf{x}_0) - Z(\mathbf{x}_0)] = 0$$

The sum of weights needs to be set to one as,

$$\sum_{i=1}^{N} \lambda_i = 1$$

Then, the expected error in estimation becomes zero as,

$$E[\hat{Z}(\mathbf{x}_0) - Z(\mathbf{x}_0)] \quad = E\left[\sum_{i=1}^{N} \lambda_i Z(\mathbf{x}_i) - Z(\mathbf{x}_0) \sum_{i=1}^{N} \lambda_i\right]$$

$$= \sum_{i=1}^{N} \lambda_i E[Z(\mathbf{x}_i) - Z(\mathbf{x}_0)] = 0 \tag{4.13}$$

The variance of the estimation error at location $\mathbf{x}_0$ using the OK weight vector $\boldsymbol{\lambda} = [\lambda_1, \ldots, \lambda_N]^T$, $\sigma^2_{\boldsymbol{\lambda}}(\mathbf{x}_0)$ is expressed as,

$$\sigma^2_{\boldsymbol{\lambda}}(\mathbf{x}_0) = E\left[\left(\hat{Z}(\mathbf{x}_0) - Z(\mathbf{x}_0)\right)^2\right] - E^2\left[\left(\hat{Z}(\mathbf{x}_0) - Z(\mathbf{x}_0)\right)\right] \tag{4.14}$$

Note that the second term inside the summation is zero, $E\left[\left(\hat{Z}(\mathbf{x}_0) - Z(\mathbf{x}_0)\right)\right] = 0$ because of the unbiasedness condition given in (4.13). Then the variance of the estimation error is written as,

$$\sigma^2_{\boldsymbol{\lambda}}(\mathbf{x}_0) = E\left[\left(\hat{Z}(\mathbf{x}_0) - Z(\mathbf{x}_0)\right)^2\right] \tag{4.15}$$

$$= E\left[\left(\sum_{i=1}^{N} \lambda_i Z(\mathbf{x}_i) - Z(\mathbf{x}_0)\right)^2\right]$$

$$= \sum_{i=1}^{N}\sum_{j=1}^{N} \lambda_i \lambda_j E[Z(\mathbf{x}_i)Z(\mathbf{x}_j)] - 2\sum_{i=1}^{N} \lambda_i E[Z(\mathbf{x}_i)Z(\mathbf{x}_0)] + E^2[Z(\mathbf{x}_0)]$$

Note that $Cov(\mathbf{x}_i, \mathbf{x}_j) = E[Z(\mathbf{x}_i)Z(\mathbf{x}_j)] - \mu^2$, $Cov(\mathbf{x}_i, \mathbf{x}_0) = E[Z(\mathbf{x}_i)Z(\mathbf{x}_0)] - \mu^2$ and $Cov(\mathbf{x}_i, \mathbf{x}_i) = E[(Z(\mathbf{x}_i))^2] - \mu^2$. Using (4.3) above yields the estimation error variance in terms of the variogram as,

$$\sigma^2_{\boldsymbol{\lambda}}(\mathbf{x}_0) = -\sum_{i=1}^{N}\sum_{j=1}^{N} \lambda_i \lambda_j \gamma(\mathbf{x}_i - \mathbf{x}_j) + 2\sum_{i=1}^{N} \lambda_i \gamma(\mathbf{x}_i - \mathbf{x}_0) \tag{4.16}$$

The task of the OK is to minimize the error in estimation $\sigma^2_{\boldsymbol{\lambda}}(\mathbf{x}_0)$ subject to the constraint that

$\sum_{i=1}^{N} \lambda_i = 1$. Namely,

$$\min_{\lambda_1,\ldots,\lambda_N} \quad \sigma^2_{\boldsymbol{\lambda}}(\mathbf{x}_0) = -\sum_{i=1}^{N}\sum_{j=1}^{N} \lambda_i\lambda_j\gamma(\mathbf{x}_i - \mathbf{x}_j) + 2\sum_{i=1}^{N} \lambda_i\gamma(\mathbf{x}_i - \mathbf{x}_0) \tag{4.17}$$

$$s.t. \qquad \sum_{i=1}^{N} \lambda_i = 1$$

The constraint of the optimization problem can be added into the objective function as,

$$\min_{\lambda_1,\ldots,\lambda_N} \sigma^2_{\boldsymbol{\lambda}}(\mathbf{x}_0) - \alpha\left(\sum_{i=1}^{N} \lambda_i - 1\right) \tag{4.18}$$

where α represents the Lagrange multiplier. The problem in (4.18) can be solved using Newton's method and the optimal solution set can be obtained from [54] as,

$$\begin{bmatrix} \lambda_1^{OK} \\ \lambda_2^{OK} \\ \vdots \\ \lambda_N^{OK} \\ \alpha^{OK} \end{bmatrix} = \begin{bmatrix} \gamma(\mathbf{x}_1 - \mathbf{x}_1) & \gamma(\mathbf{x}_1 - \mathbf{x}_2) & \ldots & \gamma(\mathbf{x}_1 - \mathbf{x}_N) & 1 \\ \gamma(\mathbf{x}_2 - \mathbf{x}_1) & \gamma(\mathbf{x}_2 - \mathbf{x}_2) & \ldots & \gamma(\mathbf{x}_2 - \mathbf{x}_N) & 1 \\ \vdots & \vdots & \ldots & \vdots & \vdots \\ \gamma(\mathbf{x}_N - \mathbf{x}_1) & \gamma(\mathbf{x}_N - \mathbf{x}_2) & \ldots & \gamma(\mathbf{x}_N - \mathbf{x}_N) & 1 \\ 1 & 1 & \ldots & 1 & 0 \end{bmatrix}^{-1} \begin{bmatrix} \gamma(\mathbf{x}_1 - \mathbf{x}_0) \\ \gamma(\mathbf{x}_2 - \mathbf{x}_0) \\ \vdots \\ \gamma(\mathbf{x}_N - \mathbf{x}_0) \\ 1 \end{bmatrix} \tag{4.19}$$

where $\boldsymbol{\lambda}^{OK} = [\lambda_1^{OK}, \ldots, \lambda_N^{OK}]^T$ represents the vector yielding the optimal OK weights, and α^{OK} is the Lagrange multiplier of OK. Upon computing the optimal OK weights and the Lagrange multiplier, the estimation error variance can be computed as,

$$\sigma^2_{\boldsymbol{\lambda}}(\mathbf{x}_0) = \alpha^{OK} + \sum_{i=1}^{N} \lambda_i^{OK}\gamma(\mathbf{x}_i - \mathbf{x}_0) \tag{4.20}$$

5. TEST RESULTS

In this chapter, we first present the test results obtained from the field using the developed wireless sensor network. Then, we also present results related with the proposed communication protocol.

As described in Section 2, we develop 11 sensors and the FC. The sensors and the FC are deployed inside the test area as shown in Fig. 5.1. The Simulation Laboratory, located inside the Yeditepe University, Department of Electrical and Electronics Engineering is selected as the test area. The light intensity of the field is estimated under two different cases respectively shown in Fig. 5.2 and Fig. 5.3. Under each case, after the user starts the field estimation script in MATLAB, 50 data requests are performed from the field with time interval between data gatherings 20 seconds. As a computer based software, we use MATLAB for data processing steps.

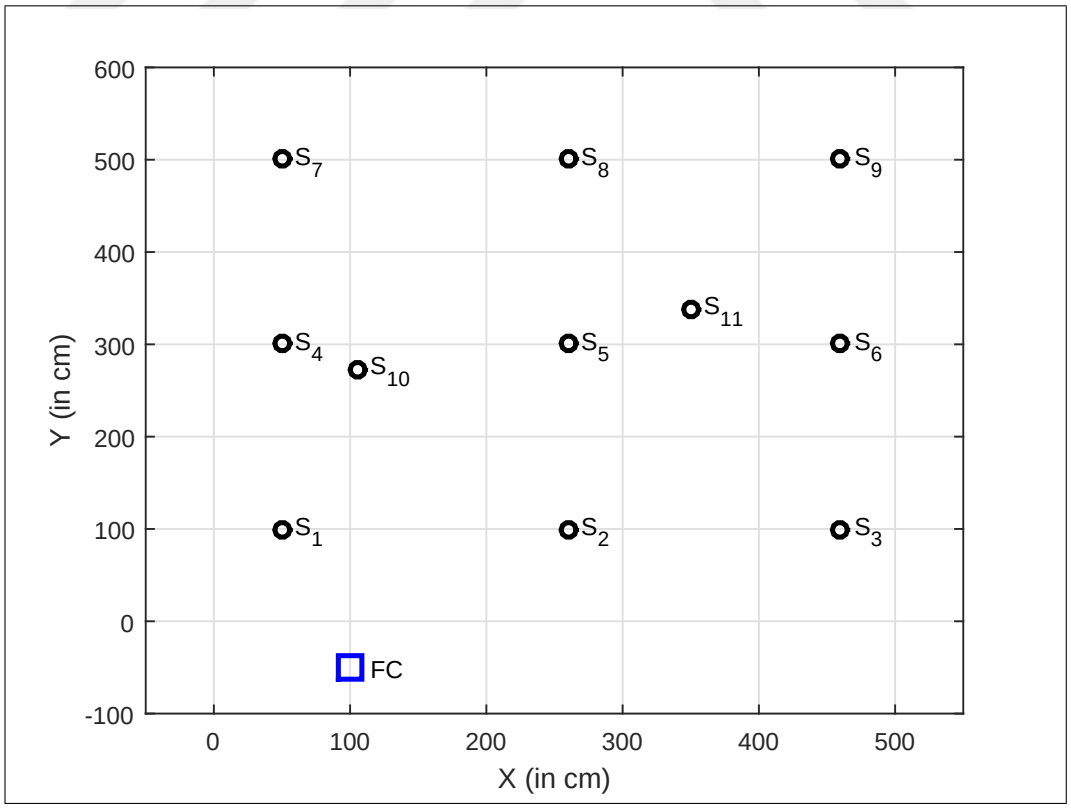


Figure 5.1. Deployment of Sensors and the Fusion Center inside the test area.

Figure 5.2. A view from the test area under CASE 1

Figure 5.3. A view from the test area under CASE 2

Fig. 5.4-(a) and Fig. 5.4-(b) show the sensor measurements collected at the fusion center at each data gatherings under CASE 1 and CASE 2 respectively. Fig. 5.4 shows that there are fluctuations on the sensor measurements even if they are observed under the same light conditions. This may be due to the interval wiring of the design. Therefore before field estimation, in order to decrease the effects of the measurement errors or the fluctuations on measurements, each sensor measurement is considered as the average of such 50 measurements. Then, we perform 500 data gatherings from the WSN. In Fig. 5.5, we show the histogram of the total time required from the beginning of the data request from the FC until the successful reception of the last sensor transmission. Our test results show that 60% of the time transmissions of 11 sensors are collected at the FC within 4 seconds.

For field estimation, we select $N = 10$ of the sensors and either of S_5, S_{10}, S_{11} as the test sensor whose measurement will be predicted using field estimation and compared with the actual measurement for case 1 and case 2 respectively. By using the measurements of the $N = 10$ sensors, we first obtain the experimental variogram as defined in (4.4). Then, we find the best variogram model presented in Section 4.1.2 fitting to the experimental variogram as in (4.11). For the optimization problem in (4.11), we determine the optimal set of variogram model parameters (a,b and v) minimizing the distance between the variogram model and the experimental variogram by using MATLAB's genetic algorithm with its default parameters. As an example, Fig. 5.6-(a) and Fig.5.6-(b) respectively show the bounded linear and the Gaussian variogram models fitted to the experimental variogram under CASE 1. Similarly, Fig. 5.7-(a) and Fig.5.7-(b) respectively show the bounded linear and the Gaussian variogram models fitted to the experimental variogram under CASE 2. We then execute the genetic algorithm 100 times and compute the mean and the standard deviation of the results where smaller standard deviation indicates that the solutions converge to closer values. Then, both Table 5.1 and Table 5.2 show that when used with the Gaussian variogram model, the relative error between the estimated field value and the actual field value are minimized. For CASE 1, the relative errors of Bounded linear, spherical, exponential and Matern models are respectively as 0.12, 0.11, 0.12, and 0.06 where the relative error of Gaussian model is only 0.02. Similarly for CASE 2, the relative errors of Bounded linear, spherical, exponential and Matern models are respectively as 0.02, 0.03, 0.02, and 0.01 where the relative error of Gaussian model is 0.01. Following the Gaussian model, Matern model can also model the

field with relatively small error. On the other hand, due to its increased number of optimization variables (3 variables, instead of 2) a larger standard deviation of solutions indicates that the solutions of the genetic algorithm may converge to a diverse set of solutions.

In Fig. 5.8-(a) and Fig. 5.8-(b), we reconstruct the field with a 3 cm resolution using (4.12) under CASE 1. Similarly, In Fig. 5.9-(a) and Fig. 5.9-(b), we reconstruct the field with 5 cm resolution under CASE 2. In both figures, we use the measurements of $N = 11$ sensors, and determine the optimal OK coefficients in (4.12) for the bounded linear and the Gaussian Variogram models.

Furthermore, in Fig. 5.10-(a) and Fig. 5.10-(b), we determine the field estimation variance using (4.20) with 3 cm resolution using under CASE 1. Furthermore, Fig. 5.11-(a) and Fig. 5.11-(b) represents the 3D version of the field estimation variance over the region of interest. Similarly, in Fig. 5.12-(a) and Fig. 5.12-(b), we determine the field estimation variance using under CASE 2. In both figures, we also use the measurements of $N = 11$ sensors, and determine the optimal OK coefficients in (4.12) for the bounded linear and the Gaussian Variogram models. When we compare the numerical results in Fig. 5.10-(a) and Fig. 5.10-(b), and than Fig. 5.12-(a) and Fig. 5.12-(b), we observe that the Gaussian variogram model provides smaller estimation error variance than that of the Bounded Linear Variogram model consistent with the results presented in Table 5.1 and Table 5.2.

Table 5.1. Optimized Ordinary Kriging Parameters under Case 1

		Test S_5		**Test** S_{10}		**Test** S_{11}	
Variogram	**Parameter**	**Mean**	**Std.**	**Mean**	**Std.**	**Mean**	**Std.**
Bounded Linear	a	464.40	340.34	431.80	411.10	688.17	514.50
	b	507.91	284.10	565.17	378.26	610.88	388.24
	Est. Val.	83.10	0.08	74.25	2.09	87.35	0.30
	Actual Val.	77.42		66.42		98.92	
	Rel. Error	0.07	0.00	0.12	0.03	0.12	0.00
Spherical	a	422.70	130.57	402.62	90.07	477.41	146.86
	b	408.32	55.93	452.66	43.27	369.79	62.15
	Est. Val.	82.53	0.27	73.62	0.26	88.28	0.23
	Actual Val.	77.42		66.42		98.92	
	Rel. Error	0.07	0.00	0.11	0.00	0.11	0.00
Exponential	a	233.88	148.20	176.82	89.96	346.88	213.87
	b	470.08	131.20	483.62	87.08	492.22	171.33
	Est. Val.	83.51	0.06	74.87	0.19	87.39	0.14
	Actual Val.	77.42		66.42		98.92	
	Rel. Error	0.08	0.00	0.13	0.00	0.12	0.00
Gaussian	a	176.38	0.00	182.77	30.68	237.52	59.66
	b	395.78	0.00	453.14	44.63	390.78	78.12
	Est. Val.	78.63	0.00	70.40	0.78	96.93	0.57
	Actual Val.	77.42		66.42		98.92	
	Rel. Error	0.02	0.00	0.06	0.01	0.02	0.01
Matern	a	218.39	362.87	177.23	333.61	280.58	460.01
	b	626.89	360.62	567.64	235.84	614.94	393.93
	v	22.72	19.96	20.66	19.05	16.22	18.97
	Est. Val.	80.41	1.82	71.81	2.11	93.45	3.60
	Actual Val.	77.42		66.42		98.92	
	Rel. Error	0.04	0.02	0.08	0.03	0.06	0.04

Table 5.2. Optimized Ordinary Kriging Parameters under Case 2

		Test S_5		**Test** S_{10}		**Test** S_{11}	
Variogram	**Parameter**	**Mean**	**Std.**	**Mean**	**Std.**	**Mean**	**Std.**
Bounded Linear	a	550.00	0.00	550.00	0.00	550.00	0.00
	b	2000.00	0.00	2000.00	0.00	2000.00	0.00
	Est. Val.	74.76	0.00	101.76	0.00	59.23	0.00
	Actual Val.	79.24		102.44		60.32	
	Rel. Error	0.06	0.00	0.01	0.00	0.02	0.00
Spherical	a	697.13	0.00	714.13	0.00	706.40	0.00
	b	2000.00	0.00	2000.00	0.00	2000.00	0.00
	Est. Val.	74.78	0.00	102.43	0.00	58.32	0.00
	Actual Val.	79.24		102.44		60.32	
	Rel. Error	0.06	0.00	0.00	0.00	0.03	0.00
Exponential	a	320.42	1.88	331.30	0.00	329.03	0.00
	b	1998.29	8.54	2000.00	0.00	2000.00	0.00
	Est. Val.	74.98	0.00	101.92	0.00	59.22	0.00
	Actual Val.	79.24		102.44		60.32	
	Rel. Error	0.05	0.00	0.01	0.00	0.02	0.01
Gaussian	a	327.84	0.00	336.71		329.98	1.05
	b	1999.39	3.05	2000.00	0.00	1998.98	7.21
	Est. Val.	78.22	0.00	102.72	0.00	60.01	0.01
	Actual Val.	79.24		102.44		60.32	
	Rel. Error	0.01	0.00	0.00	0.00	0.01	0.00
Matern	a	26.22	13.65	26.39	6.59	48.22	157.13
	b	1965.32	81.77	1975.57	81.36	1971.64	80.76
	v	45.65	9.64	44.20	10.34	43.50	11.99
	Est. Val.	78.23	0.17	102.64	0.06	59.90	0.49
	Actual Val.	79.24		102.44		60.32	
	Rel. Error	0.01	0.00	0.00	0.00	0.01	0.01

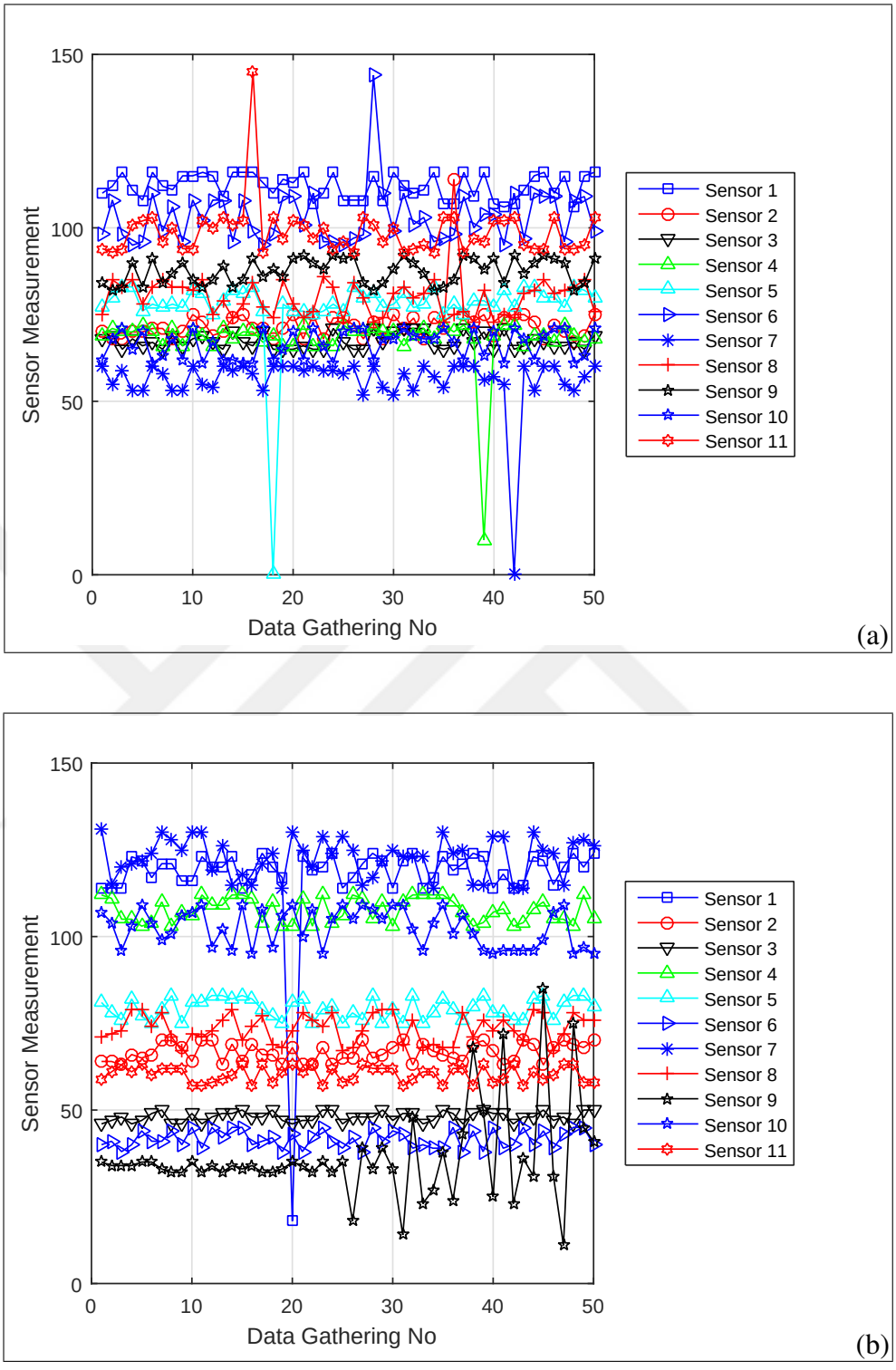


Figure 5.4. Sensor Measurements at each data gathering (a) CASE 1 (b) CASE 2

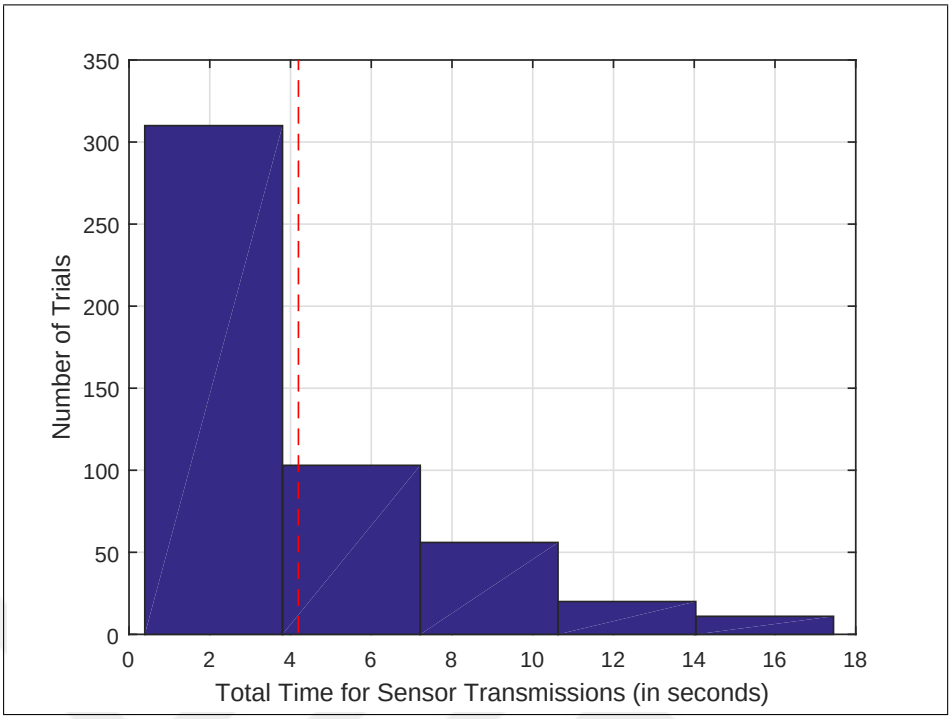
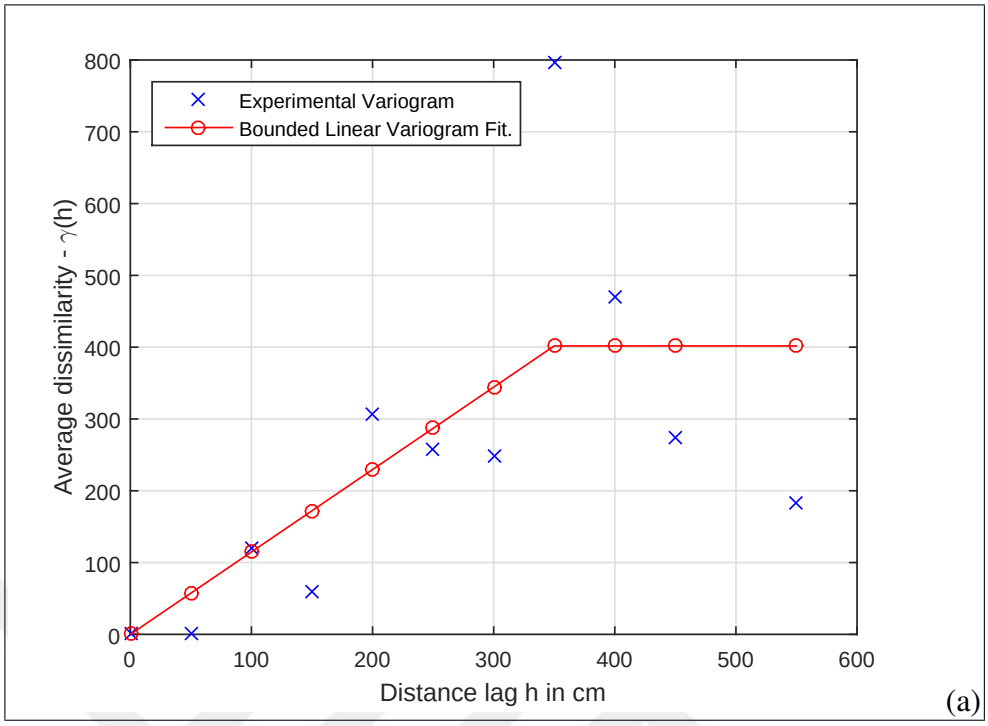


Figure 5.5. Total Transmission Time of Sensor Measurements



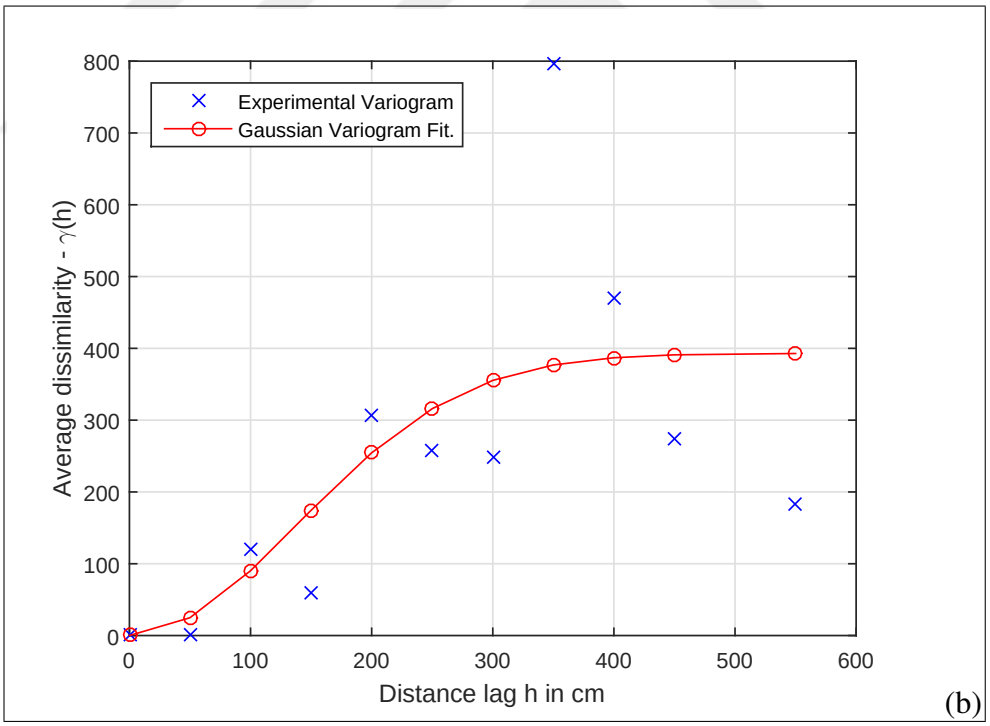


Figure 5.6. CASE 1: Experimental Variogram (a) Bounded Linear, (b) Gaussian.

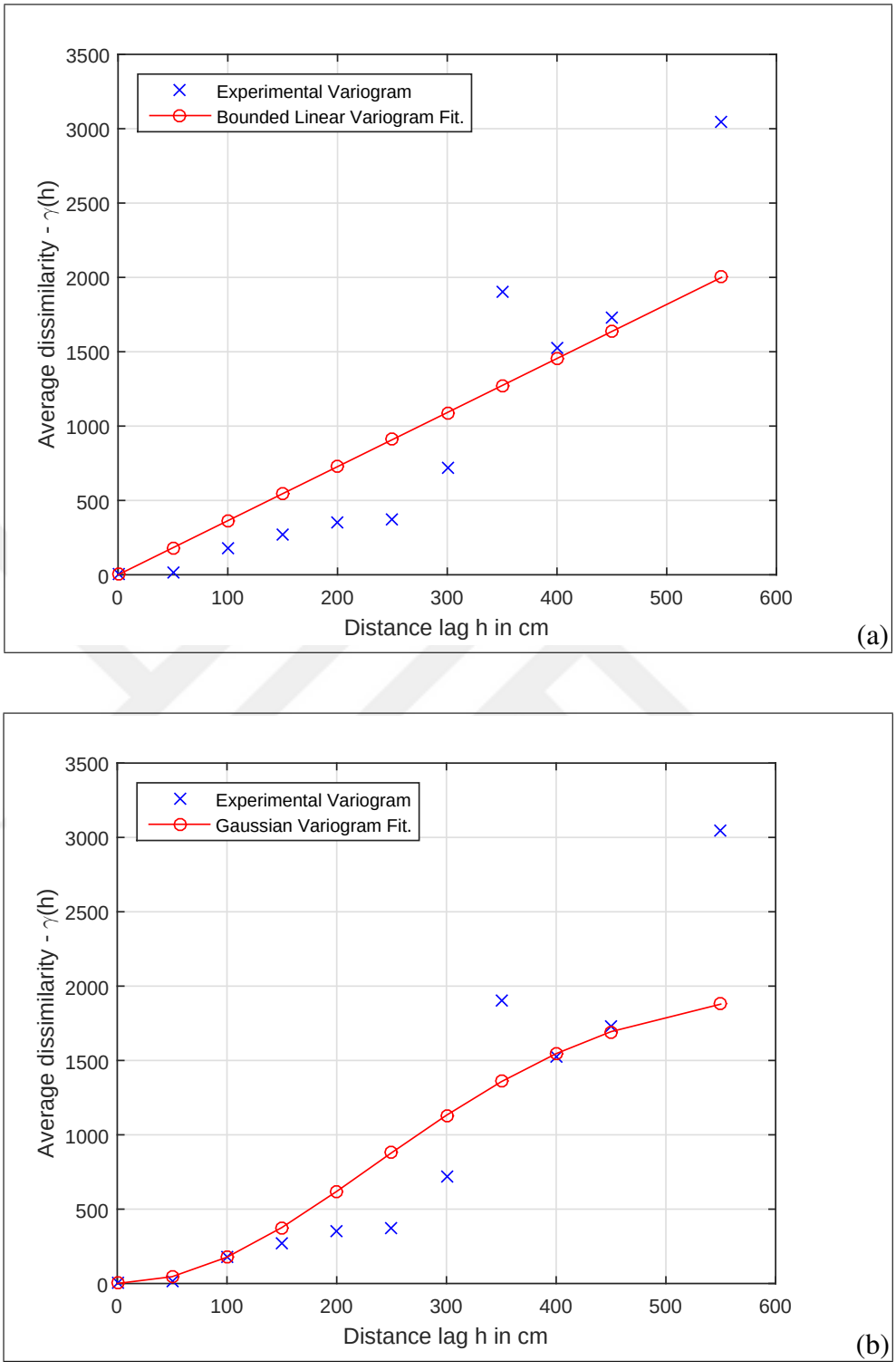


Figure 5.7. CASE 2: Experimental Variogram (a) Bounded Linear, (b) Gaussian.

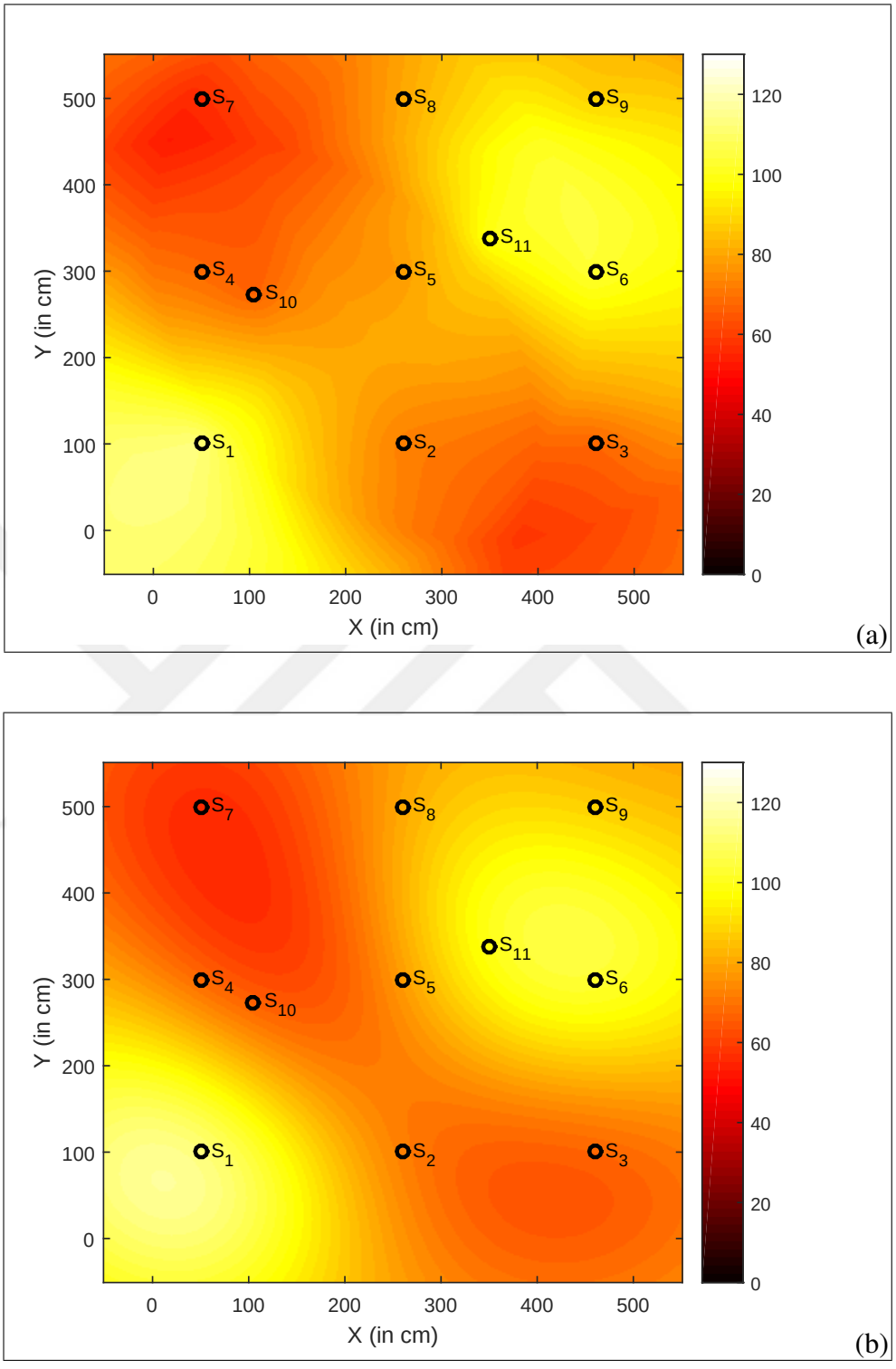


Figure 5.8. CASE 1 - Field Reconstruction (a) Bounded Linear, (b) Gaussian.

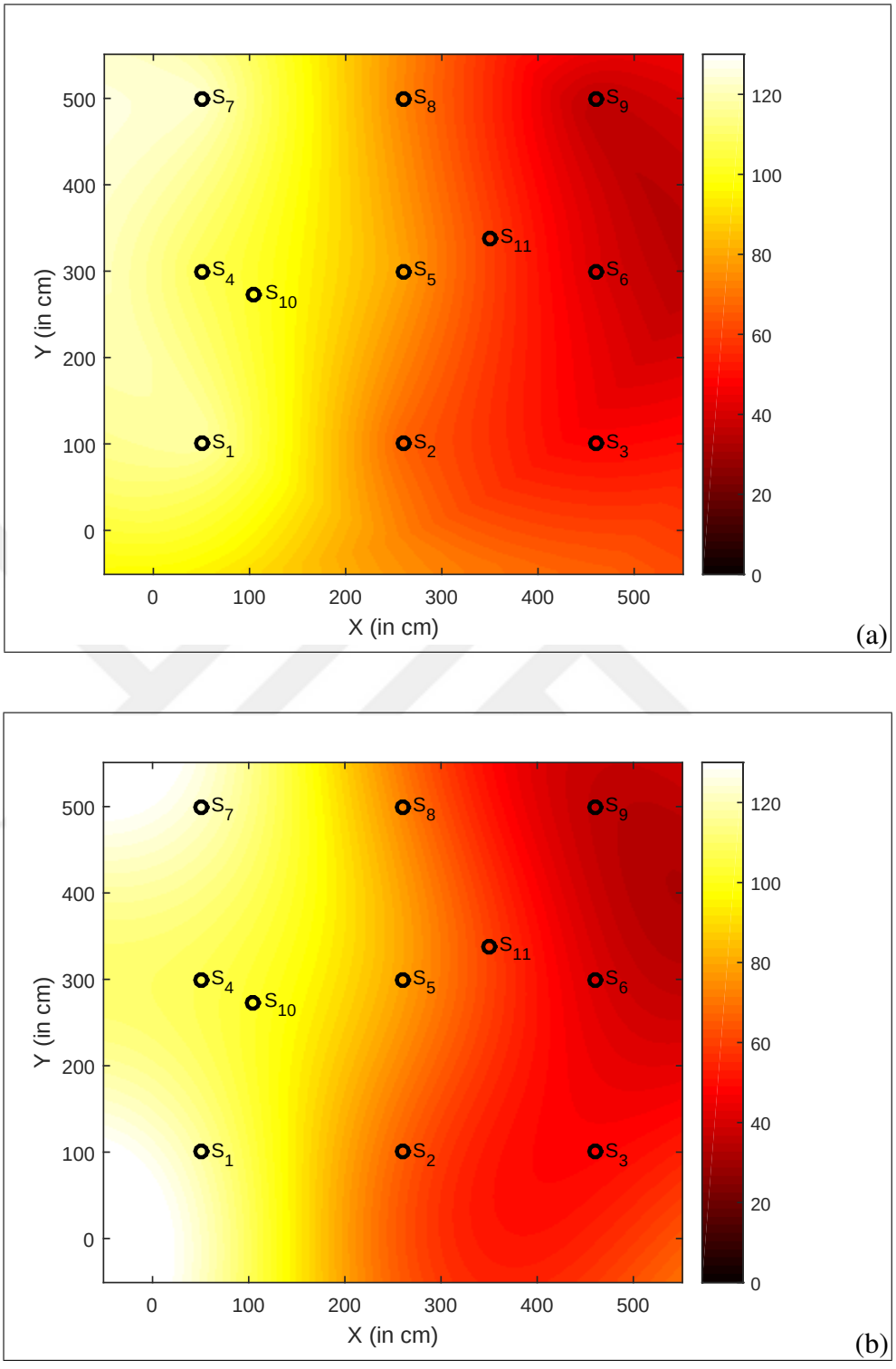


Figure 5.9. CASE 2 - Field Reconstruction (a) Bounded Linear, (b) Gaussian.

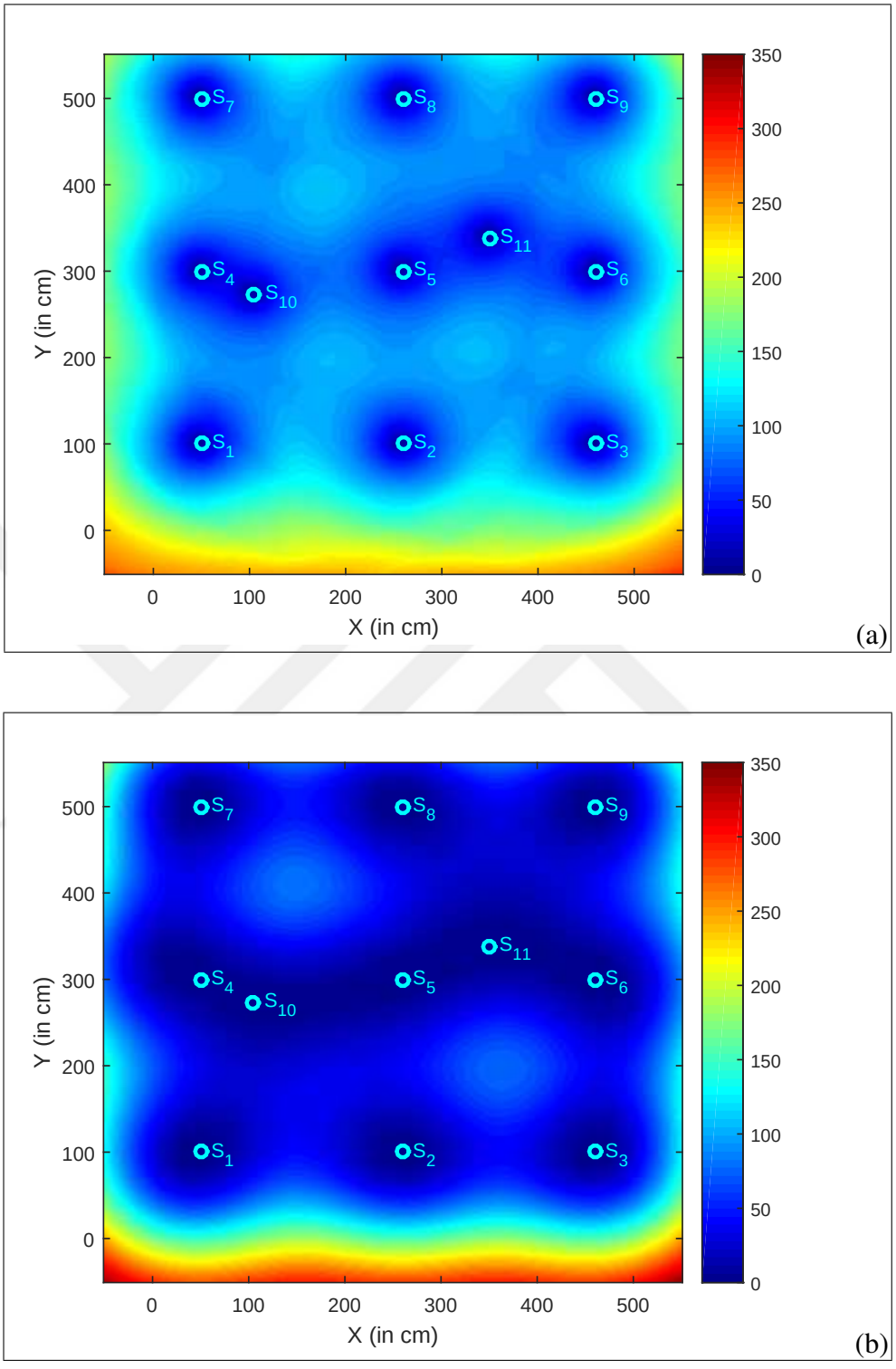


Figure 5.10. CASE 1: Estimation Error Variance (a) Bounded Linear (b) Gaussian.

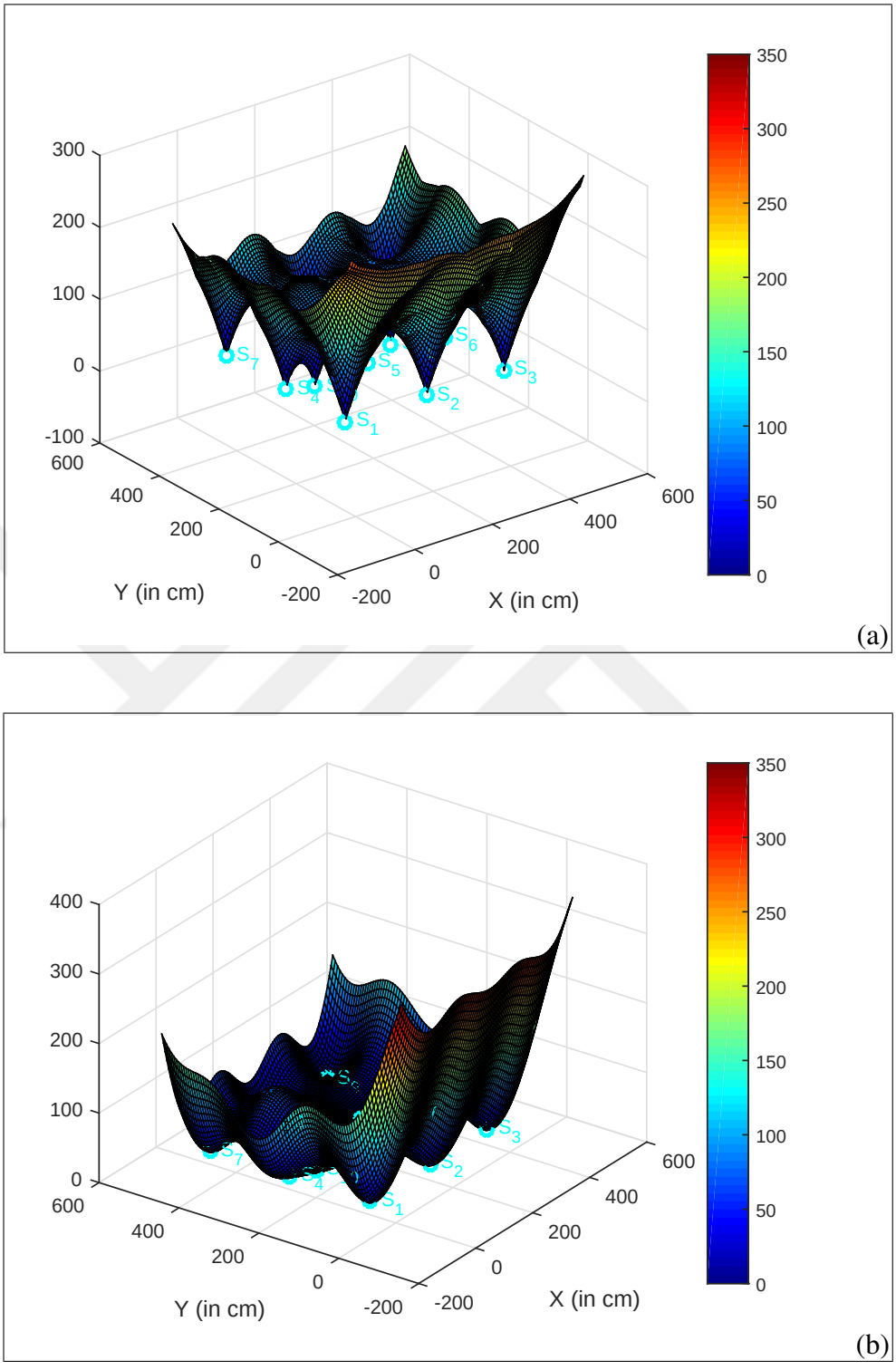


Figure 5.11. CASE 1: 3D Representation of Est. Error Var. (a) B. Linear (b) Gaussian.

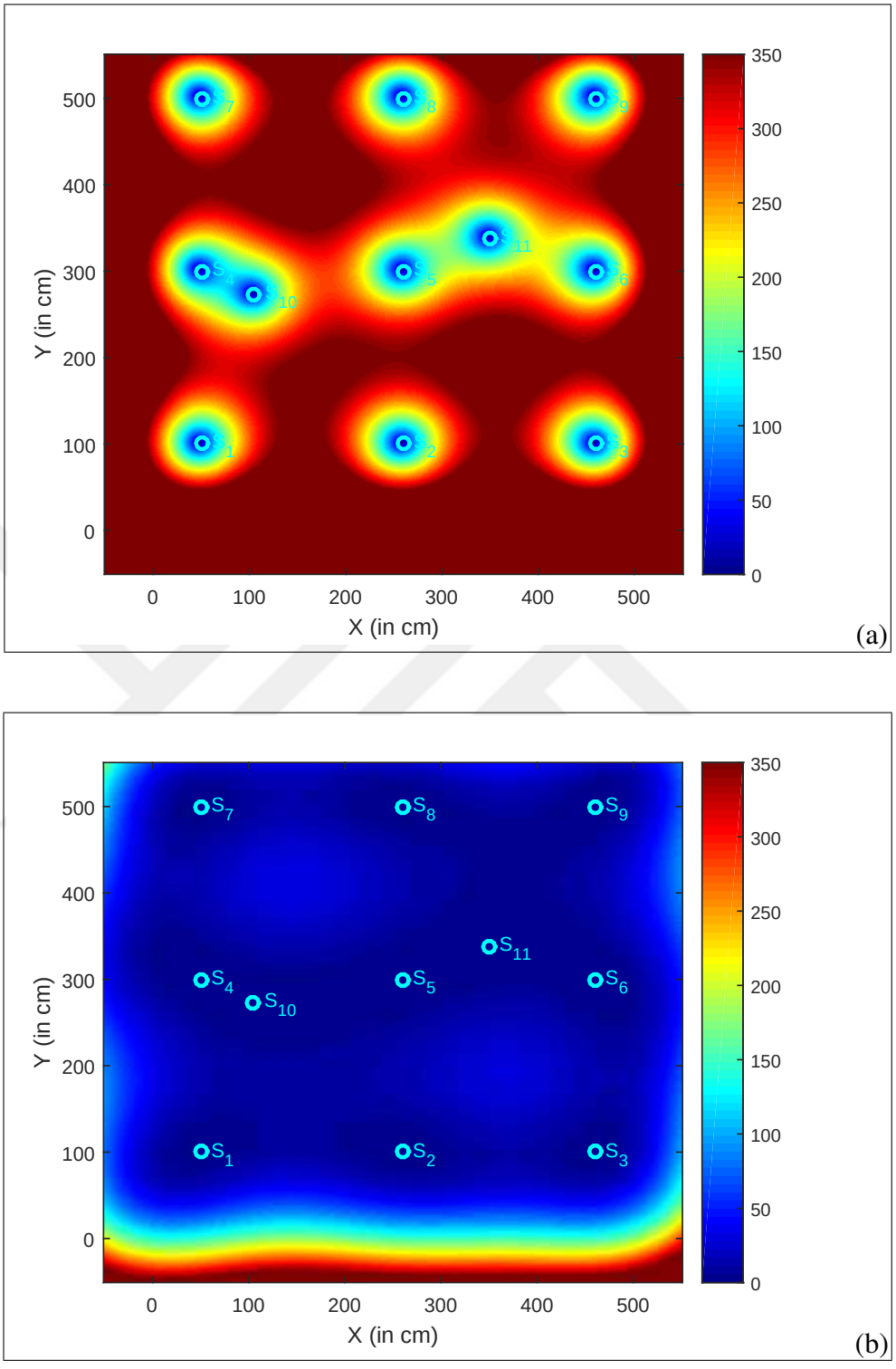


Figure 5.12. CASE 2: Estimation Error Variance (a) Bounded Linear, (b) Gaussian.

6. CONCLUSIONS AND FUTURE WORK

In this thesis, we performed field estimation by using sensors developed with low power consumption and low cost units, MSP430G2553 as the microcontroller and nRF24L01+ as the transceiver unit. Using such sensors, we then formed a WSN and define the communication protocol of the WSN in a CSMA manner. In this work, we implemented 11 sensors where measurements of $N = 10$ sensors were used to determine the experimental variogram and then to determine the mathematical variogram model which best fit to the experimental variogram. The measurement of the 11^{th} sensor was then used to validate that both actual and predicted measurements were close to each other. Under the Gaussian variogram model, the light intensity at a given point in the field was successfully estimated from the weighted sum of the sensor measurements using ordinary kriging.

In this work, we performed field estimation using received sensor measurements where the measurement noise is ignored. Instead, we received many measurements from each sensor and take the average of measurements as a final measurement value before field estimation. As a future work, we model the sensor measurement errors as measurement noise into the Ordinary Kriging model where the measurement noise may be due to calibration or internal circuit connections of each sensors.

As a future work, we plan to develop more sensors having additional sensing modalities such as temperature, humidity, pressure, etc. Rather than requesting measurements from all sensors in the WSN, the variogram model can be captured from a subset of sensors and then the sensors near the interested location can be activated to better estimate the intensity at the interested location. In the proposed sensor design, most of the energy was spent by the communication unit, NRF24L01+, where the communication unit was always on in the proposed communication protocol. For energy saving, each sensor can further turn off its communication module for a specific time after successfully sending its measurement to the FC. Furthermore, in this work we considered sensors transmit their measurements directly to the fusion center. As a future work, we will consider multi-hop transmissions for delivering sensor measurements over each other. Rather than gathering all sensor measurements at the

fusion center, a sensor selection procedure can be first performed to reduce the communication overhead. Furthermore, the FC or the central processor responsible for field estimation can be located at a far away location where the sensor measurements need to be transmitted over a Wi-Fi or a cellular network.

In this work, we performed spatial field estimation in a given region of interest. Field estimation problem can be generalized over time and space where both spatial and temporal characteristics of the field need to be determined based on received sensor measurements. Last but not least, rather than field estimation, other applications of statistical inference problems such as distributed detection, estimation, localization and tracking can be realized by using the WSN developed here.

REFERENCES

1. NORDIC Semiconductor. nRF24L01+ Single Chip 2.4GHz Transceiver Product Specification v1.0;. Available From: 2017-02-14. https://www.nordicsemi.com.

2. Electronics M. Texas Instruments MSP430;. Available From: 2017-02-14. http://www.mouser.com.tr/.

3. Ünsalan C, Gürhan HD. *Programmable Microcontrollers With Applications - MSP430 LaunchPad With CCS and Grace*. McGraw-Hill Education; 2014.

4. Popkin G, et al.. First Trip to the Stars. Nature Publishing Group Macmillan Building, 4 Crinan St, London N1 9XW, ENGLAND; 2017.

5. Akyildiz IF, Su W, Sankarasubramaniam Y, Cayirci E. A Survey On Sensor Networks. *IEEE Communications Magazine*. 2002 Aug;40(8):102–114.

6. Cressie N. *Statistics For Spatial Data*. John Wiley & Sons; 2015.

7. Lichtenstern A. *Kriging Methods In Spatial Statistics*. Technische Universität München. 2013;.

8. Kang J, Jin R, Li X. Regression Kriging-Based Upscaling of Soil Moisture Measurements From a Wireless Sensor Network and Multiresource Remote Sensing Information Over Heterogeneous Cropland. *IEEE Geoscience and Remote Sensing Letters*. 2015 Jan;12(1):92–96.

9. Nguyen L, Kodagoda S. Soil Organic Matter Estimation In Precision Agriculture Using Wireless Sensor Networks. In: *2016 14th International Conference on Control, Automation, Robotics and Vision (ICARCV)*; 2016. p. 1–6.

10. Nevat I, Peters GW, Septier F, Matsui T. Estimation of Spatially Correlated Random

Fields in Heterogeneous Wireless Sensor Networks. *IEEE Transactions on Signal Processing*. 2015 May;63(10):2597–2609.

11. Taylor K, Griffith C, Lefort L, Gaire R, Compton M, Wark T, et al. Farming the Web of Things. *IEEE Intelligent Systems*. 2013 Nov;28(6):12–19.

12. Postigo-Malaga M, Supo-Colquehuanca E, Matta-Hernandez J, Pari L, Mayhua-Lpez E. Vehicle Location System and Monitoring As A Tool For Citizen Safety Using Wireless Sensor Network. In: *2016 IEEE ANDESCON*; 2016. p. 1–4.

13. Baghaee S, Gurbuz SZ, Uysal-Biyikoglu E. Application and Modeling of a Magnetic WSN for Target Localization. In: *2013 UKSim 15th International Conference on Computer Modelling and Simulation*; 2013. p. 687–692.

14. Eris C, Gungor VC, Boluk PS. Analysis Of Battery-powered Sensor Node Lifetime For Smart Grid Applications. In: *2016 24th Signal Processing and Communication Application Conference (SIU)*; 2016. p. 2117–2120.

15. Chaves PR, Branquinho OC, Carvalho MFH. Criteria For The Setting Up Of Low Cost Wireless Sensor Networks In Small and Medium Size Manufacturing Enterprises (A Case Study). In: *2016 8th IEEE Latin-American Conference on Communications (LATINCOM)*; 2016. p. 1–6.

16. ADVANTICSYS. 802.15.4 Mote Modules;. Available From: 2017-02-14. https://www.advanticsys.com.

17. Rahman NAA, Jambek AB. Wireless Sensor Node Design. In: *2016 3rd International Conference on Electronic Design (ICED)*; 2016. p. 332–336.

18. Chen T, Zhao L. The Design of the Wireless Smart Subpoena System Based on MSP430. In: *2015 8th International Conference on Intelligent Networks and Intelligent Systems (ICINIS)*; 2015. p. 153–156.

19. Kumar V, Sonavane SS, Patil BP. Designing Ultra Low Power Wireless Sensor Network With TCP/IP Link. In: *2009 2nd International Conference on Adaptive Science Technology (ICAST)*; 2009. p. 86–91.

20. Uttarkar NK, Kommu A, Kanchi RR. Design and Development Of Data Acquisition System For A Remote Furnace Using MSP430G2553 and Zigbee. In: *International Conference on Information Communication and Embedded Systems (ICICES2014)*; 2014. p. 1–5.

21. Ardelean A, Mischie S. Development Of An Electronic Game Based On Bluetooth Communication. In: *2014 11th International Symposium on Electronics and Telecommunications (ISETC)*; 2014. p. 1–4.

22. Wang Y, Hu C, Feng Z, Ren Y. Wireless Transmission Module Comparison. In: *2014 IEEE International Conference on Information and Automation (ICIA)*; 2014. p. 902–907.

23. Weder A. An Energy Model of the Ultra-Low-Power Transceiver nRF24L01 for Wireless Body Sensor Networks. In: *2010 2nd International Conference on Computational Intelligence, Communication Systems and Networks*; 2010. p. 118–123.

24. Zhao D, Peng C. A Small Low-Power Reliable Communication Module in a Wireless Monitoring System. In: *2007 1st International Conference on Bioinformatics and Biomedical Engineering*; 2007. p. 1194–1197.

25. Zhang P, Sun L, Zhang P, Hou R, Tian G, Liu X. Wireless Network Design and Implementation in Smart Home. In: *2013 6th International Conference on Intelligent Networks and Intelligent Systems (ICINIS)*; 2013. p. 167–170.

26. Chen S, Yuan YJ. Wireless electronic tourist guide system based on microcontroller. In: *2011 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC)*; 2011. p. 1–4.

27. Shi S, Lu T, Zhang H, Xu L, Gulliver TA. A design of active RFID tags based on

NRF24L01. In: *2013 10th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*; 2013. p. 210–213.

28. Wang Y, Chi Z. System of Wireless Temperature and Humidity Monitoring Based on Arduino Uno Platform. In: *2016 Sixth International Conference on Instrumentation Measurement, Computer, Communication and Control (IMCCC)*; 2016. p. 770–773.

29. Ma Z, Pan X. Agricultural Environment Information Collection System Based On Wireless Sensor Network. In: *2012 IEEE Global High Tech Congress on Electronics*; 2012. p. 24–28.

30. Zhou Y, Fan L. Design of Miniature Unmanned Helicopter Attitude Monitoring System Based on nRF24L01+. In: *2013 Third International Conference on Instrumentation, Measurement, Computer, Communication and Control*; 2013. p. 588–592.

31. Hao S, Zongtao C. Design Of The Environmental Temperature and Humidity Wireless Monitoring System. In: *2015 12th IEEE International Conference on Electronic Measurement Instruments (ICEMI)*. vol. 03; 2015. p. 1652–1657.

32. Ni S, Su J, Nie L, Qu S. Design Of Multi-point Wireless Temperature Measuring System. In: *2012 Proceedings of International Conference on Modelling, Identification and Control*; 2012. p. 422–425.

33. Chen Z, Hu C, Liao J, Liu S. Protocol Architecture For Wireless Body Area Network Based On nRF24L01. In: *2008 IEEE International Conference on Automation and Logistics*; 2008. p. 3050–3054.

34. Christ P, Neuwinger B, Werner F, Rckert U. Performance Analysis Of The nRF24L01 Ultra-Low-Power Transceiver In A Multi-transmitter and Multi-receiver Scenario. In: *2011 IEEE SENSORS Proceedings*; 2011. p. 1205–1208.

35. Zhu N. Simulation and Optimization Of Energy Consumption In Wireless Sensor Networks. Ecole Centrale de Lyon; 2013.

36. Harrington B, Huang Y, Yang J, Li X. Energy-Efficient Map Interpolation for Sensor Fields Using Kriging. *IEEE Transactions on Mobile Computing*. 2009 May;8(5):622–635.

37. Demirkol I, Ersoy C, Alagoz F. MAC Protocols For Wireless Sensor Networks: A Survey. *IEEE Communications Magazine*. 2006 April;44(4):115–121.

38. Huang P, Xiao L, Soltani S, Mutka MW, Xi N. The Evolution of MAC Protocols in Wireless Sensor Networks: A Survey. *IEEE Communications Surveys Tutorials*. 2013 First;15(1):101–120.

39. Rajagopalan R, Varshney PK. Data-aggregation Techniques In Sensor Networks: A Survey. *IEEE Communications Surveys Tutorials*. 2006 Fourth;8(4):48–63.

40. Leon-Garcia A, Widjaja I. *Communication Networks*. McGraw-Hill, Inc.; 2003.

41. Kay SM. *Fundamentals Of Statistical Signal Processing*. Prentice Hall PTR; 1993.

42. Jedermann R, Palafox-Albarrn J, Barreiro P, Ruiz-Garca L, Robla JI, Lang W. Interpolation Of Spatial Temperature Profiles By Sensor Networks. In: *2011 IEEE SENSORS Proceedings*; 2011. p. 778–781.

43. Castello C, Fan J, Davari A, Chen RX. Temperature Control Framework Using Wireless Sensor Networks and Geostatistical Analysis for Total Spatial Awareness. In: *2009 10th International Symposium on Pervasive Systems, Algorithms, and Networks*; 2009. p. 717–721.

44. Ray P, Varshney PK. Estimation of spatially distributed processes in wireless sensor networks with random packet loss. *IEEE Transactions On Wireless Communications*. 2009 June;8(6):3162–3171.

45. Liu S, Masazade E, Fardad M, Varshney PK. Sparsity-aware field estimation via ordinary Kriging. In: *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*; 2014. p. 3948–3952.

46. Erickson P, Cline M, Tirpankar N, Henderson T. Gaussian Processes For Multi-sensor

Environmental Monitoring. In: *2015 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*; 2015. p. 208–213.

47. Atmel M. ATmega328/P;. Available From: 2017-07-14. http://www.atmel.com/.

48. Atmel M. Atmel ATmega640/V-1280/V-1281/V-2560/V-2561/V;. Available From: 2017-07-14. http://www.atmel.com/.

49. Atmel M. ATmega16U4/ATmega32U4;. Available From: 2017-07-14. http://www.atmel.com/.

50. Texas Instruments (TI). MSP430G2x53, MSP430G2x13 Mixed Signal Microcontroller (Rev. J);. Available From: 2017-02-14. http://www.ti.com/lit/gpn/msp430g2553.

51. Robotistan. Wireless NRF24L01+ 2.4GHz Transceiver Module;. Available From: 2017-02-14. http://www.robotistan.com/wireless-nrf24l01-24ghz-transceiver-modul-24ghz-alici-verici-modul-1.

52. Instructables. Communication Bluetooth Module With HC-05 HC-06;. Available From: 2017-07-10. http://www.instructables.com/id/Communication-Bluetooth-Module-With-HC-05-HC-06/.

53. GitHub. nRF24L01+ Library for MSP430 Microcontroller Line;. Available From: 2017-02-14. https://github.com/spirilis/msprf24.

54. Boyd S, Vandenberghe L. *Convex Optimization*. Cambridge university press; 2004.

APPENDIX A: MAIN NRL24L01+ FUNCTIONS

Basic nRF24L01+ functions are given as below [53].

- msprf24_init(), initializes SPI line and other GPIO ports to make ready for communication.
- w_tx_add(char *addr), writes a new transmit address where 'addr' is the transmit address up to 5 byte.
- w_rx_add(char pipe, char *addr), writes a new receiving address for determined pipe id where 'pipe' is the data pipe number and 'addr' is the receiving address up to 5 byte.
- w_tx_payload(char len, char *data), writes the payload with pre-defined len bytes to the TX FIFO where 'len' is the payload length configurable up to 32 byte and 'data' is the information transmitting represented with 'len' byte.
- w_tx_payload_noack(char len, char *data), writes the payload with pre-defined len bytes to the TX FIFO where Enhanced ShockBurst auto-ack property will be disabled. Therefore, the transmitter side will not wait for an ack package from the receiving side since receiver is informed with a "NOACK" bit set in the received package which means there is no need to send an acknowledgement package to the transmitter.
- r_rx_payload(char len, char *data), provides to read the contents of RX FIFO with len bytes.
- r_rx_peek_payload_size(), observes the payload size of the next coming data available at the RX FIFO.
- flush_rx(), provides to flush all RX FIFOs with their contents.
- msprf24_open_pipe(char pipeid, char autoack), enables to RX pipe at given id number represented with 'pipeid' where 'autoack' is 0 or 1 which determines autoacknowledgement is used or unused.
- msprf24_close_pipe(char pipeid), enables to close the RX pipe specified where 'pipeid' is the pipe number.
- msprf24_pipe_isopen(char pipeid), tests the specified pipe to observe that it is open or close at the query moment.
- msprf24_set_pipe_packetsize(char pipeid, char size), configures the packet size between

0 to 32 for the specified RX pipe where 'pipeid' is the pipe number and 'size' is the packet size which is usable at that pipe number 'pipeid'.

- msprf24_set_address_width(), configures the transceiver address size in byte which can be 3, 4 and 5 byte.
- msprf24_set_retransmit_delay(int us), defines the retransmission timeout in microseconds when there is a failure at the transmission operation where 'us' can be adjusted between 250 and 4000µs. It is known that minimum values can differ by RF speed and packet size.
- msprf24_set_retransmit_count(char count), defines the maximum number of transmissions tolerated before giving up where 'count' has the valid range between 1 and 15.
- msprf24_get_last_retransmits(), returns with the number of retransmissions which are recorded at the transceiver register at the last attempt for sending a package.
- msprf24_get_lostpackets(),returns with the total number of payloads lost that are recorded a register on the transceiver.
- msprf24_set_channel(), configures the RF channel which is used for transmit or receiver operations. It is used with global variable 'rf_channel' where valid range is from 0 to 125. It is considered that, the channel ranges represented with 'rf_channel' specifies 1MHz increments above 2400MHz. Therefore, minimum level channel 0 means that it operates at 2400MHz(2.4GHz) and maximum level channel 125 means that it operates at 2525MHz(2.525GHz).
- msprf24_set_speed_power(), configures the transmission speed of the RF transceiver and transmitting power level. It is used with global variable 'rf_speed_power'. Table 2.4 shows the configurable transmitting RF powers and current consumption of the transceiver for each power level.
- msprf24_current_state(), enables to read the RF transceiver state currently.
- msprf24_is_alive(), checks the nRF24L01+ module to know that it is present and communicating or not. The return value is a boolean integer where 0 is false means non-communicating and 1 is true means module is present and communicating.
- msprf24_powerdown(), powers down the nRF module to enter low sleep mode.
- msprf24_standby(), passes the nRF module to enter Standby-I.

- msprf24_activate_rx(), enables the receiving mode to start scanning the air and listening the medium.
- msprf24_activate_tx(), enables the transmitter mode to send the TX FIFO contents.
- msprf24_queue_state(), reads the RF24_FIFO_STATUS register of the transceiver and returns with information about RF24_QUEUE_TXFULL, RF24_QUEUE_TXEMPTY, RF24_QUEUE_RXFULL, RF24_QUEUE_RXEMPTY.
- msprf24_rx_pending(), reads the STATUS register and if there is incoming data available at the RX FIFO, it returns with 1 and 0 if not.
- msprf24_get_irq_reason(), reads the STATUS register and stores the necessary information about IRQ at the rf_irq global variable. It contains information about RX24_IRQ_TXFAILED, RX24_IRQ_RX and RX24_IRQ_FLAGGED.
- msprf24_irq_clear(char irqflag), clears all IRQs on the nRF24 device where 'irqflag' is that which IRQs should be cleared and removed from the device.
- msprf24_enable_feature(char feature), enables the feature which is given with the function.
- msprf24_disable_feature(char feature), disables the feature specified.
- msprf24_scan(), checks a RF signal presence via scanning the air at the current frequency channel. By using this function, a signal with a strength greater than -60dBm can be detected. Also, the return value gives the channel rating at the current RF channel between 0 and 255(8 bit unsigned integer).
- tx_reuse_last_payload(), is used in transmitter mode to obtain quick and efficient handle for retransmissions in case of packet loss situations.
- pulse_ce(), tells nRF24 to start transmitting its TX FIFO contents.
- w_ack_payload(char pipe, char len, char *data), is used to send a custom acknowledgement with payload received by reporting the pipe and length of data information instead of automated zero-length acknowledgements used by default where 'pipe' is the data pipe id, 'len' is the length of the payload and 'data' is the information which is received and acknowledged.

APPENDIX B: HEADER FILES

Basic nRF24L01+ headers are given as below [53].

MSP430 library for interfacing with the nRF24L01+ RF transceiver by Nordic Semiconductor:

/* Configuration variables used to tune RF settings during initialization and for runtime reconfiguration. You should define all 4 of these before running msprf24_init(); /

```
extern uint8_t rf_crc;
extern uint8_t rf_addr_width;
extern uint8_t rf_speed_power;
extern uint8_t rf_channel;
```

/* Status variable updated every time SPI I/O is performed */

```
extern uint8_t rf_status;
```

/* Test this against RF24_IRQ_FLAGGED to see if the nRF24's IRQ was raised; it also * holds the last recorded IRQ status from msprf24_irq_get_reason(); */

```
extern volatile uint8_t rf_irq;
```

/* RF speed settings – nRF24L01+ compliant, older nRF24L01 does not have 2Mbps. */

```
#define RF24_SPEED_250KBPS 0x20
#define RF24_SPEED_1MBPS 0x00
#define RF24_SPEED_2MBPS 0x08
#define RF24_SPEED_MAX RF24_SPEED_2MBPS
#define RF24_SPEED_MIN RF24_SPEED_250KBPS
#define RF24_SPEED_MASK 0x28
```

/* RF transmit power settings */

```
#define RF24_POWER_7DBM 0x07
// 7dBm available with SI24R1 Taiwanese knockoff modules
#define RF24_POWER_0DBM 0x06
#define RF24_POWER_MINUS6DBM 0x04
#define RF24_POWER_MINUS12DBM 0x02
#define RF24_POWER_MINUS18DBM 0x00
#define RF24_POWER_MAX RF24_POWER_0DBM
#define RF24_POWER_MIN RF24_POWER_MINUS18DBM
#define RF24_POWER_MASK 0x07

/* Available states for the transceiver's state machine */
#define RF24_STATE_NOTPRESENT 0x00
#define RF24_STATE_POWERDOWN 0x01
#define RF24_STATE_STANDBY_I 0x02
#define RF24_STATE_STANDBY_II 0x03
#define RF24_STATE_PTX 0x04
#define RF24_STATE_PRX 0x05
#define RF24_STATE_TEST 0x06

/* IRQ "reasons" that can be tested. */
#define RF24_IRQ_TXFAILED 0x10
#define RF24_IRQ_TX 0x20
#define RF24_IRQ_RX 0x40
#define RF24_IRQ_MASK 0x70
// Bit 7 used to signify that the app should check IRQ status, without // wasting time in the
interrupt vector trying to do so itself.
#define RF24_IRQ_FLAGGED 0x80

/* Queue FIFO states that can be tested. */
#define RF24_QUEUE_TXFULL RF24_FIFO_FULL
#define RF24_QUEUE_TXEMPTY RF24_TX_EMPTY
```

#define RF24_QUEUE_RXFULL RF24_RX_FULL
#define RF24_QUEUE_RXEMPTY RF24_RX_EMPTY

User configuration of nRF24L01+ connectivity parameters, e.g. IRQ, CSN, CE pin assignments, Serial SPI driver type:

/* CPU clock cycles for the specified amounts of time–accurate minimum delays * required for reliable operation of the nRF24L01+'s state machine. */
/* Settings for 1MHz MCLK.
#define DELAY_CYCLES_5MS 5000
#define DELAY_CYCLES_130US 130
#define DELAY_CYCLES_15US 15
*/

/* Settings for 8MHz MCLK. */
#define DELAY_CYCLES_5MS 40000
#define DELAY_CYCLES_130US 1040
#define DELAY_CYCLES_15US 120

/* Settings for 16MHz MCLK
#define DELAY_CYCLES_5MS 80000
#define DELAY_CYCLES_130US 2080
#define DELAY_CYCLES_15US 240
*/

/* Settings for 24MHz MCLK.
#define DELAY_CYCLES_5MS 120000
#define DELAY_CYCLES_130US 3120
#define DELAY_CYCLES_15US 360
*/

/* SPI port–Select which USCI port we're using. * Applies only to USCI devices. USI users can keep these * commented out. */

//#define SPI_DRIVER_USCI_A 1

#define SPI_DRIVER_USCI_B 1

/* Operational pins – IRQ, CE, CSN (SPI chip-select) */

/* IRQ */

#define nrfIRQport 2

#define nrfIRQpin BIT2

/* CSN SPI chip-select */

#define nrfCSNport 2

#define nrfCSNportout P2OUT

#define nrfCSNpin BIT1

/* CE Chip-Enable (used to put RF transceiver on-air for RX or TX) */

#define nrfCEport 2

#define nrfCEportout P2OUT

#define nrfCEpin BIT0

Register definitions for manipulating the Nordic Semiconductor nRF24L01+ RF transceiver chipsets:

/* Register Map /

#define RF24_CONFIG 0x00

#define RF24_EN_AA 0x01

#define RF24_EN_RXADDR 0x02

#define RF24_SETUP_AW 0x03

#define RF24_SETUP_RETR 0x04

#define RF24_RF_CH 0x05

```
#define RF24_RF_SETUP 0x06
#define RF24_STATUS 0x07
#define RF24_OBSERVE_TX 0x08
#define RF24_CD 0x09
#define RF24_RPD 0x09
#define RF24_RX_ADDR_P0 0x0A
#define RF24_RX_ADDR_P1 0x0B
#define RF24_RX_ADDR_P2 0x0C
#define RF24_RX_ADDR_P3 0x0D
#define RF24_RX_ADDR_P4 0x0E
#define RF24_RX_ADDR_P5 0x0F
#define RF24_TX_ADDR 0x10
#define RF24_RX_PW_P0 0x11
#define RF24_RX_PW_P1 0x12
#define RF24_RX_PW_P2 0x13
#define RF24_RX_PW_P3 0x14
#define RF24_RX_PW_P4 0x15
#define RF24_RX_PW_P5 0x16
#define RF24_FIFO_STATUS 0x17
#define RF24_DYNPD 0x1C
#define RF24_FEATURE 0x1D

/* Register Bits */
#define RF24_MASK_RX_DR BIT6
#define RF24_MASK_TX_DS BIT5
#define RF24_MASK_MAX_RT BIT4
#define RF24_EN_CRC BIT3
#define RF24_CRCO BIT2
#define RF24_PWR_UP BIT1
#define RF24_PRIM_RX BIT0
#define RF24_ENAA_P5 BIT5
```

```
#define RF24_ENAA_P4 BIT4
#define RF24_ENAA_P3 BIT3
#define RF24_ENAA_P2 BIT2
#define RF24_ENAA_P1 BIT1
#define RF24_ENAA_P0 BIT0
#define RF24_ERX_P5 BIT5
#define RF24_ERX_P4 BIT4
#define RF24_ERX_P3 BIT3
#define RF24_ERX_P2 BIT2
#define RF24_ERX_P1 BIT1
#define RF24_ERX_P0 BIT0
#define RF24_AW BIT0
#define RF24_ARD BIT4
#define RF24_ARC BIT0
#define RF24_PLL_LOCK BIT4
#define RF24_CONT_WAVE BIT7
#define RF24_RF_DR BIT3
#define RF24_RF_DR_LOW BIT5
#define RF24_RF_DR_HIGH BIT3
#define RF24_RF_PWR BIT1
#define RF24_LNA_HCURR BIT0
#define RF24_RX_DR BIT6
#define RF24_TX_DS BIT5
#define RF24_MAX_RT BIT4
#define RF24_RX_P_NO BIT1
#define RF24_TX_FULL BIT0
#define RF24_PLOS_CNT BIT4
#define RF24_ARC_CNT BIT0
#define RF24_TX_REUSE BIT6
#define RF24_FIFO_FULL BIT5
#define RF24_TX_EMPTY BIT4
```

```
#define RF24_RX_FULL BIT1
#define RF24_RX_EMPTY BIT0
#define RF24_EN_DPL BIT2
#define RF24_EN_ACK_PAY BIT1
#define RF24_EN_DYN_ACK BIT0

/* Instructions */
#define RF24_R_REGISTER 0x00
#define RF24_W_REGISTER 0x20
#define RF24_REGISTER_MASK 0x1F
#define RF24_R_RX_PAYLOAD 0x61
#define RF24_W_TX_PAYLOAD 0xA0
#define RF24_FLUSH_TX 0xE1
#define RF24_FLUSH_RX 0xE2
#define RF24_REUSE_TX_PL 0xE3
#define RF24_R_RX_PL_WID 0x60
#define RF24_W_ACK_PAYLOAD 0xA8
#define RF24_W_TX_PAYLOAD_NOACK 0xB0
#define RF24_NOP 0xFF
```