

THE DEVELOPMENT OF
SIMPLE, LOW-COST AND POWER-EFFICIENT
PIC-BASED
WIRELESS SENSOR NODE AND ACCESS POINT

AHMAD IZHAR BIN ROSLI

**The Development of Simple, Low-Cost and Power-Efficient PIC-Based Wireless
Sensor Node and Access Point**

by

Ahmad Izhar bin Rosli

(Supervisor: Abu Bakar Sayuti Hj. Mohd Saman)

Dissertation submitted in partial fulfilment of
the requirements for the
Bachelor of Engineering (Hons)
(Electrical and Electronics Engineering)

SEPTEMBER 2012

Universiti Teknologi PETRONAS
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

**The Development of Simple, Low-Cost and Power-Efficient PIC-Based Wireless
Sensor Node and Access Point**

by

Ahmad Izhar bin Rosli

A project dissertation submitted to the
Electrical and Electronics Engineering Programme
Universiti Teknologi PETRONAS
in partial fulfilment of the requirement for the
BACHELOR OF ENGINEERING (Hons)
(ELECTRICAL AND ELECTRONICS ENGINEERING)

Approved by,

(Abu Bakar Sayuti Hj. Mohd Saman)

UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK
September 2012

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

AHMAD IZHAR BIN ROSLI

ABSTRACT

Wireless sensor network (WSN) has been attracting growing interests for new development of much improved power efficiency, lower cost and smaller size without sacrificing the robustness and reliability of the sensor nodes. The advances in new embedded technology have made it possible as more features can be integrated into tiny but high performance processor, sensor and hardware. However, this high-end technology has cost the consumer to spend quite expensive investment which is not appropriate for simple data acquisition application in unsecured places. It is not practical and economical when the sensor nodes are lost or failed in the environment. This paper describes the design and the development of simple wireless sensor nodes and access point by using Peripheral Interface Controller (PIC) for low rate data logging application. It is to be integrated to the high-end WSN as the extension to the available sensor nodes or base station for large deployment in unsafe places. The project involves the hardware and software design of low-power and economic wireless sensor nodes. The nodes are configured manually to transmit the temperature data one node at a time to prevent interference of the shared 433MHz frequency band. PIC controls the operation of the wireless sensor node and access point for data encoding, decoding and the operating or sleep state of the nodes. For low cost sensor nodes, cheap RF modules, low-pin PIC and tiny-sized nodes are taken into consideration. To improve the data reliability in compensating the noisy cheap RF modules, useful PIC features such as timers, counters and interrupts are used to produce Manchester encoded and decoded data. In terms of power efficiency, the circuit design and hardware selection of eXtreme Low Power (XLP) PIC improves the energy consumption in addition to the integration of sleep mode to the nodes.

ACKNOWLEDGEMENT

First and foremost I would like to express my humble gratitude to Allah S.W.T, because of His blessings and guidance I am able to finish my Final Year Project (FYP) for this semester.

I would like to sincerely thank my university Universiti Teknologi PETRONAS (UTP) especially Electrical and Electronics Engineering Department for providing me with essential support and facilities in completing the project.

Special thanks and appreciation to my FYP supervisor, Mr. Abu Bakar Sayuti Hj. Mohd Saman for his sincere guidance, monitoring and teaching to make it possible for me to finish the project successfully. His strong support and willingness to share his knowledge and wide expertise has given me a very important experience in this project.

In this opportunity, I would like to thank UTP staff and lab assistants who helped me to complete the project especially in conducting the testing work. I really appreciate their help and the sharing of information in order for me to gain good result during the experiment.

Last but not least, deepest gratitude to my beloved family and friends in UTP for their continuous support and encouragement which enabled me to do my best for this project. I hope after this project completion, all the findings and knowledge that I shared through this report can be useful for everyone for study and personal research work. Thank you and Allah bless you all.

Table of Contents

| | |
|--|-----|
| CERTIFICATION OF APPROVAL | i |
| CERTIFICATION OF ORIGINALITY | ii |
| ABSTRACT..... | iii |
| ACKNOWLEDGEMENT | iv |
| List of Figures..... | vi |
| List of Tables | vii |
| CHAPTER 1: INTRODUCTION | 1 |
| 1.1 Background..... | 1 |
| 1.2 Problem Statement..... | 2 |
| 1.3 Objectives | 3 |
| 1.4 Scope of Study | 4 |
| CHAPTER 2: LITERATURE REVIEW/THEORY..... | 5 |
| 2.1 Wireless Sensor Network..... | 5 |
| 2.2 History of Wireless Sensor Network | 5 |
| 2.3 Applications of WSN..... | 6 |
| 2.4 Design Characteristics of Wireless Sensor Node..... | 7 |
| 2.4.1 Hardware Selection..... | 7 |
| 2.4.2 Low Power Consumption..... | 8 |
| 2.4.3 Cost Effective..... | 11 |
| 2.5 Sensor Network Communication Architecture | 12 |
| 2.6 Scheduled-based Protocols or Time Division Multiple Access (TDMA)..... | 14 |
| CHAPTER 3: METHODOLOGY/PROJECT WORK | 15 |
| 3.1 Methodology | 15 |
| 3.2 Overview of Project Work | 16 |
| 3.3 Manchester Encoding for Serial Digital Communication | 19 |
| 3.4 Data frame..... | 20 |
| 3.5 Overview of Manchester Encoding and Decoding of Data Frame | 21 |
| 3.6 Sensor Node Hardware Schematic..... | 22 |
| 3.7 Access Point Hardware Schematic | 24 |
| 3.8 Sensor Node Software Design | 25 |
| 3.9 Sensor Node Software Design | 28 |
| 3.10 Gantt Chart..... | 31 |
| 3.11 Tools | 31 |
| CHAPTER 4: RESULTS AND DISCUSSION | 32 |
| 4.1 Experiments and Project Work Result | 32 |
| 4.1.1 Testing of the Transmitter and Receiver Performance..... | 32 |
| 4.1.2 Sending Constant Data of One Byte without Sleep Mode..... | 35 |
| 4.1.3 Sending Constant Data of One Byte with Sleep Mode | 38 |
| 4.2 Final Prototype Testing and Result..... | 44 |
| CHAPTER 5: CONCLUSION AND RECOMMENDATION..... | 47 |
| REFERENCES | 48 |
| APPENDICES | 50 |

List of Figures

| | |
|--|----|
| FIGURE 1. The Main Application and Scope of Project..... | 3 |
| FIGURE 2. Typical Sensor Module Hardware [6] | 8 |
| FIGURE 3. Power Consumption for All Motes..... | 10 |
| FIGURE 4. Total Current Consumption by Sensor Node Core Components..... | 10 |
| FIGURE 5. Sensor Network Communication Architecture [6] | 12 |
| FIGURE 6. Time Division Multiple Access (TDMA) [19]..... | 14 |
| FIGURE 7. Flow Chart for Prototype Development | 15 |
| FIGURE 8. Manchester Encoded Data for Bit 1 and Bit 0..... | 20 |
| FIGURE 9. Data Frame | 20 |
| FIGURE 10. Manchester Encoded Data Frame | 21 |
| FIGURE 11. Sensor Node Hardware Schematic | 22 |
| FIGURE 12. Access Point Hardware Schematic | 25 |
| FIGURE 13. Flow Chart for Sensor Node..... | 26 |
| FIGURE 14. FSM Diagram for Manchester Encoded Data Frame | 26 |
| FIGURE 15. FSM Diagram for SENDING_N State | 27 |
| FIGURE 16. FSM Diagram for Access Point..... | 29 |
| FIGURE 17. FSM Diagram for Manchester Decoding | 30 |
| FIGURE 18. Data Frame Decoding Output..... | 30 |
| FIGURE 19. TX-433 Transmitter..... | 32 |
| FIGURE 20. RX-433 receiver | 33 |
| FIGURE 21. Receiver Output from Transmitter Sending Data of ZERO or Transmitter is OFF | 34 |
| FIGURE 22. Receiver Output from Transmitter Sending Data of ONE | 34 |
| FIGURE 23. Receiver Output from Transmitter Sending Long and Short Data Bit of ONE | 34 |
| FIGURE 24. Overview of Experimental Setup | 36 |
| FIGURE 25. Sensor Node..... | 36 |
| FIGURE 26. Access Point | 36 |
| FIGURE 27. Sending Manchester Encoded Data of Character '1' or Binary 00110001 | 37 |
| FIGURE 28. Longer Sweep Time from FIGURE 16. Repeated Constant Data Frame with Gap of 200ms (idle time) in Between | 37 |
| FIGURE 29. Decoded Output Signal from PIC..... | 38 |
| FIGURE 30. Character '1' is Produced at LCD Output | 38 |
| FIGURE 31. Noise Affecting the Data Frame | 39 |
| FIGURE 32. Isolating Sync Pulse from the Noise..... | 40 |
| FIGURE 33. Data Frame with Supress Noise Byte | 40 |
| FIGURE 34. Timer1 Gate for Sync Pulse Detection Coding | 42 |
| FIGURE 35. Timer1 Gate Rejecting Input Pulse of Less Than 20ms | 42 |
| FIGURE 36. Timer1 Gate Detecting Input Pulse of 20ms | 42 |
| FIGURE 37. Timer1 Gate Rejecting Input Pulse of More Than 20ms..... | 42 |
| FIGURE 38. Timer1 Gate Detecting 80ms Sync Pulse | 43 |
| FIGURE 39. Initial Node 1 and Node 2 Data Frame Transmission | 43 |
| FIGURE 40. Node 1 and Node 2 Data Frame Interferes with Each Other After Some Time | 43 |
| FIGURE 41. Successful Data Transmission Between Sensor Node and Access Point | 44 |
| FIGURE 42. Successful Data Transmission Between Two Sensor Nodes and Access Point | 45 |
| FIGURE 43. Final Prototype of Sensor Nodes Developed into PCB | 46 |

List of Tables

| | |
|---|----|
| TABLE 1. Market Survey of Crossbow Motes..... | 11 |
| TABLE 2. Market Survey of PIC-based Nodes..... | 12 |
| TABLE 3. Prototype Development Gantt Chart..... | 31 |
| TABLE 4. Total Cost for Sensor Node..... | 46 |
| TABLE 5. Total Cost for Access Point..... | 46 |

CHAPTER 1: INTRODUCTION

1.1 Background

Sensor links our real analogous world with virtual digital world by capturing the actual phenomena and converting it to some forms so that it can be processed and manipulated to meet our real life applications. For these applications, sensors are integrated into various kinds of machines that we use every day like machines, portable devices, vehicles and environmental applications. Sensors are important part of any device in performing its functionality to assist and enhance human life. For example, sensors are vital in keeping good car performance or providing accurate input for touch-screen hand phones, thus increasing human productivity through its contribution in making human life easier. Besides that, sensors play an important role in enhancing public safety for instance avoiding catastrophic infrastructure failure, providing early natural disaster warning system and through its application in the domestic security system.

The common use of distributed sensor systems is contributed by the phenomenal technology developments for example very large scale integration (VLSI), micro-electromechanical systems (MEMS) and wireless communications. The advances of semiconductor technology with major improvement in microprocessor processing speed which is offered in smaller package contributes to wide application of this distributed sensor system. The advances of computing shrinking and sensing technologies contribute to the growth of tiny, power-saving and economic sensor. From day to day, embedded technology finds itself to be more demanded by consumers as it plays important role in various area applications mainly in distributed sensor network system.

During its early development, the application of sensors in distributed system is focused in military defence and aerospace systems. But nowadays, sensor networks are appreciated due to its critical contribution in various systems for instance to monitor, protect civil infrastructure, preserve national power grid and diagnose domestic or oil & gas pipeline infrastructure. Nowadays, there are already

hundreds of sensor node networks which are at present being used to monitor large scale geographic regions. This application enables the modelling and forecasting of ecology conservation, environmental pollution and natural disaster, collecting structural health information on critical structures like bridges and sky-scrapers, controlling water usage and manipulating fertilizers or pesticides to improve plantations quality and quantity.

1.2 Problem Statement

The commercially available sensor node in the current market is found out to be expensive. As the consequence, this will become less economic wise when large numbers of sensor nodes are needed to be deployed for large geography data monitoring. In 2011, Malaysia has taken steps into receiving technology in applying wireless sensor network for domestic agriculture field. It can be expected that in the future, small-scaled farmer will be exposed to this new agriculture technology based on wireless sensor network and use it in their crops plantations. Through the application of this wireless sensing in agriculture, the environment parameters such as temperature, light and humidity can be constantly monitored, predicted and controlled so that the crops quality and production rate can be maintained or increased. Imagine that the local farmer with limited budget will have to deploy large number of high cost sensor nodes across the wide plantation fields. It will be impossible for this marginal farmer to implement technology to their cropland in order to increase plantations health and productivity.

Besides that, the commercially available sensor nodes are tends to be less power-friendly. Typical sensor node consumes a lot of power to maintain stable operation which is crucial in smooth and accurate data logging. High power consumption means high maintenance work which is the replacement of battery. It is not practical to change those large number of sensor nodes regularly one by one which are possibly situated in difficult and unreachable areas such as deep in the forest, up high on trees, active volcanic activity land, corrosive chemical tank, over delicate crops and others. This regular replacement of battery also consumes significant amount of maintenance cost which is not favourable for personal and domestic usage.

Based on the above situations, the problem statements for the project are as follows:

1. How to build a simple low cost and power-saving pair of wireless sensor node and access point?
2. What is the performance of the wireless sensor node in aspect of distance, power consumption and cost to make sure it reaches the objective?

1.3 Objectives

The objectives of the project are:

1. To build a simple pair of wireless sensor node and access point which are low-powered and cost-effective.
2. To analyse the performance of the built wireless sensor node in real life application.

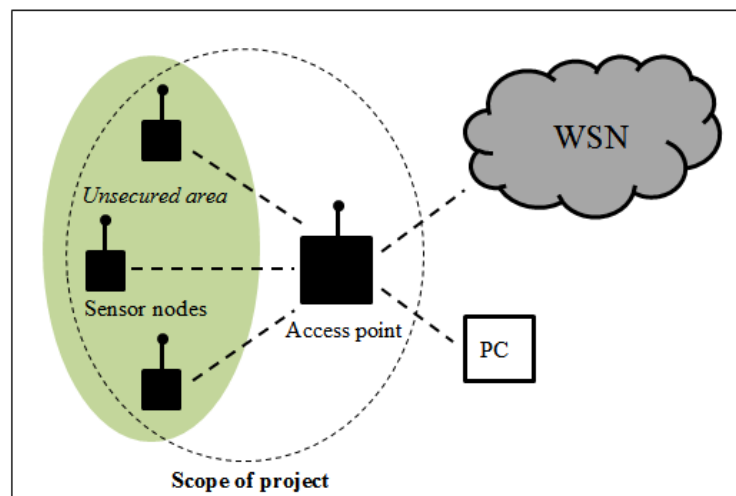


FIGURE 1. The Main Application and Scope of Project

The scope of the project as shown in Figure 1 is to build a pair of reliable wireless sensor nodes and access point which are small size, low cost and power effective. The main application of the sensor nodes is for the low rate data acquisition application which will be deployed in unsecured area or difficult places such as at the mangrove swamp, forest, top of trees, inside crops greenhouse, farms and even outside the house. Generally, it is to be integrated to WSN as the extension to the available sensor nodes or base station for deployment in unsafe places. The advantage of small-sized sensor nodes is that it can be placed in difficult or limited-

space area easily by just throwing them to the desired place. The lost or broken of the sensor nodes will not become economical issue as it is cheap, easy to build and replaced. In addition, the low power feature of sensor nodes makes it less frequent for maintenance work in replacing the battery.

1.4 Scope of Study

The development of the wireless sensor node and access point involves the knowledge of the author in embedded and telecommunication system in electrical and electronics engineering.

The hardware and software development of the wireless sensor node and access point requires knowledge in microcontroller which is PIC. The hardware work involves the selection of PIC which comes with only the needed feature of the sensor node to work such as I/O pins, ADC and others. This is to keep the simplicity, small size, low power and cost effective of the wireless sensor node. The data transmission between the sensor node and access point involves serial wireless communication between the transmitter and the receiver which requires knowledge in wireless telecommunication. The coding program that will be developed and downloaded to the microcontroller will be written in C language. C language is chosen instead of assembly because of its simplicity and the author experience in previous Engineering Team Project. Programming the microcontroller in C to control the sensor node and access point operation is the biggest challenge as it involves serial data communication and smart programming for the design low-power sensor node.

CHAPTER 2: LITERATURE REVIEW/THEORY

2.1 Wireless Sensor Network

Wireless Sensor Network (WSN) can be defined as a network of devices or *nodes* which collects the environmental parameters and communicate with each other through wireless links [1]. Besides communicating with each other, the sensor nodes also communicating with base station or access point where the data is gathered for remote processing, visualization, analysis and storage by connecting the base station to other device like PC or other network [2].

There are two types of WSN communication which are single-hop and multi-hop network [2]. Single-hop is a point-to-point network which in this topology, each sensor node communicated directly to the base station without depending on other nodes to find routing to send the sensed information. Usually, this kind of communication suits more on sensor nodes with large radio transmission ranges or for short-distance sensor nodes application. Whereas multi-hop is a mesh network where the sensor nodes communicates with each other and depends on nearby sensor nodes for finding data routing. Besides capturing and send its own sensed data, each sensor node acts as the relay which cooperates with others in forwarding and propagating the information to the base station.

2.2 History of Wireless Sensor Network

The early development of wireless sensor networks has been driven by the military and defence application [3]. The research of wireless sensor network started in 1978 when the Defence Advanced Research Projects Agency (DARPA) organized the Distributed Sensor Nets Workshop (DAR 1978). The objective of DAR 1978 is to study on the challenges of sensor network including networking technologies, signal processing techniques, and distributed algorithms. Significantly, in early 1980s, DARPA also conducted the Distributed Sensor Network (DSN) program which later followed by other sensor network research program called Sensor Information Technology (SensIT). These several early work mark the starting point

where the world beginning to realize the significance of WSN and more work and researches had been carried out to improve the technology [4][6][7][8][9][13].

DSN illustrates the basic WSN during its early operation, guiding and advanced the WSN technology as the world achieves now [3]. DSN program which is developed by DARPA involves large number of spatially distributed sensor nodes which works autonomously by transferring resources from each other. The operation assumed is to determine which the best node that the information will be routed to for certain data usage. At that early time, this is considered to be a very aspiring program. Due to the technology limitation at that time, the processing was mainly carried out on minicomputers and Ethernet without any personal computer and workstations.

2.3 Applications of WSN

When looked back to the history when first WSN research is conducted by DARPA in 1978, it can be seen that the main purpose of the early WSN development stage is for the military application in the battlefield [1]. Nowadays, the application of WSN in military comprises of the monitoring of hostile activities in remote areas such as strategic key roads and rural village and force protection for example ensuring the cleared building remain cleared from any kind of any hostile trespassing. One of WSN application that receives significant interest from military personnel recently is the support for base protection by using acoustic and electro-optical sensors [4]. The goal of this base protection is to give early warning system by gathering real-time sensor data from the wireless sensor nodes deployed across the surrounding mountains in a radius 4km from the centre of the military base station.

In Malaysia, agriculture field plays an important role in accelerating the economy for the country. In November 2011, Malaysia through Jabatan Pertanian Sabah (JPS) has taken further step in modern agriculture by receiving the technology of “Rangkaian Sensor Tanpa Wayar” (Wireless Sensor Network) [18]. The wireless technology application which is developed together by MIMOS, UPM-NEST, Kontron Design Manufacturing Services Sdn. Bhd. and Intel Technology Sdn Bhd.

will be used in the agriculture development program called “Precision Agriculture Farming” where the first WSN technology for domestic agriculture application is used to monitor the environmental parameters of the plantations. Tuaran Agriculture Research is selected to be the pioneer site for the application of WSN technology in the program.

2.4 Design Characteristics of Wireless Sensor Node

2.4.1 Hardware Selection

A sensor node consists of four major parts which are sensing, processing, transceiver and power unit [6]. In certain application, extra components such as location finding system, power scavenger or mobilizer can be added to the sensor node to enhance its capabilities. Typically, the sensing unit comprises of sensor and ADC. The ADC converts the sensed analogue data into binary for processing by the processing unit. The processing unit executes the various instructions based on the program and controls the sensor node operation such as routing according to the fed data. The data processed or certain status is then transferred wirelessly through transceiver unit. Power unit which is one of the most important components of a sensor node supplies all the node components to carry out each specific task. This power unit may be supplied by power harnessing unit such as solar cell modules.

One aspect of hardware selections that must be considered is the possibility of using PIC microcontroller that needs to be integrated with other core components such as sensor, transmitter, receiver, decoder, encoder and access point. Several previous researches have successfully use PIC microcontroller for WSN application [7][8]. Research by Reinch and Reilly (2004) concludes that the effective usage of PIC as a sensor node must include instruction codes written in simple and static manner. Complex instructions must be avoided as it is inefficient and impossible to implement in limited processing resources architecture of the PIC [7]. The hardware architecture used by the research is shown in Figure 2.

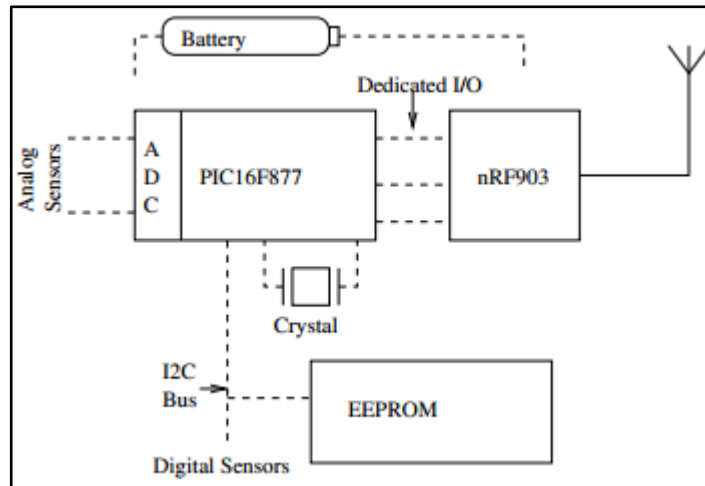


FIGURE 2. Typical Sensor Module Hardware [6]

2.4.2 Low Power Consumption

One of the important features of the sensor node design is to run in low power during its operation [6]. Practically, this enables the sensor node to be working in long time period with one time battery usage after it is being deployed. This gives practical and low cost benefits in maintaining large network of nodes. Research by Kimpe (2008) states that low-power WSN can be achieved by using several ways [11]:

Low-power Wireless Mesh Routing

The technique is to create mesh or multi-hop WSN. As stated previously, in multi-hop topology, each sensor node is dependent on the other nodes to propagate or forward the data to the base station. Note that all nodes are sleeping most of the time. Through mesh routing, this opens up for time spacing for the nodes to wake up in split second to transmit the data to the other nodes and come back to sleep mode. For example, by expecting the next node wake-up status, the current node goes to sleep so that only half awake period of each node is used to communicate or transmit data with each other. This means the method uses strict time synchronization instead of retrieving each nodes status (wake-up/sleep) repeatedly which can waste the power. In simple words, this involves the accurate synchronization of the sleep and wake-up cycles of the nodes in relative to each other.

Peak Current

Peak current or current spike can take place when functional blocks switch on simultaneously which usually tends to occur during the start-up of different functional blocks. The way to prevent peak current is to carefully manage the on-off time of each functional block to avoid processing too many complex functions such as ADC conversion and serial data transmission simultaneously.

Graceful Power Failure

This involves smart power management system by the nodes by sending early warning status and other critical information to other nodes before the power runs out. This is to ensure smooth operation by putting itself in a state for quick recovery when the power is replenished (through battery change or solar scavenger).

Sleep Current

The sleep current characteristic relates to the hardware selection. Careful choice can be made by taking power consumption of each sensor nodes component in the design. The longer time for the node is expected to run on a single battery life, the longer the sleep time must be configured between the two wake-up periods.

Wake-up Time

The wakeup or start-up time cycles are needed to be in smart sequencing and kept as brief as possible in reaching active-state of node components such as voltage regulators and clock oscillators to conserve energy consumption by the node. During sleep state, turn off as many unneeded or idled components or functions as possible without compromising normal operation to minimize duty-cycling.

Research by Torres (2006), with smart power management system implemented in a sensor node design, there is significant power saving during the sensor node operation [9]. The hardware design involves the usage of well-known [12] Crossbow Technology Mica2 (MPR400) motes with Atmel Atmega128L microcontroller, radio and TinyOS operating system. Figure 2 shows the power consumption of the motes deployed in the research.

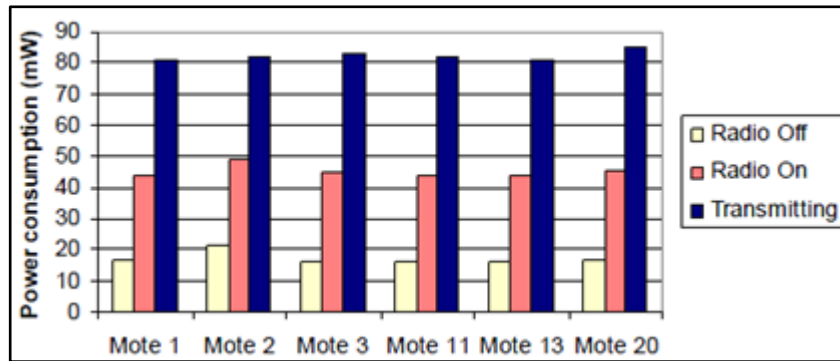


FIGURE 3. Power Consumption for All Motes

Research by Corke et. al. (2009) concludes one of the challenges of data monitoring using WSN is the power consumption. This statement is based on the research of environmental WSN which involves the deployment of to test for the application of WSN technology for long duration and large scale rainforest ecosystems monitoring in Queensland. The sensor nodes deployed are expected to hold sufficient energy during the operation. The technology used is to combine un-rechargeable and solar-chargeable battery as the power sources. Conclusively, the research team found out that although the solar power is integrated with the battery in each sensor node, it is still not enough for the large transducers power demand [13]. The research also showed that there is some CPU energy consumption saving by using sleep mode feature integrated in the nodes [13]. Figure 3 shows the breakdown of total current consumption by core components of a sensor node (CPU and radio) applying 100% duty cycle.

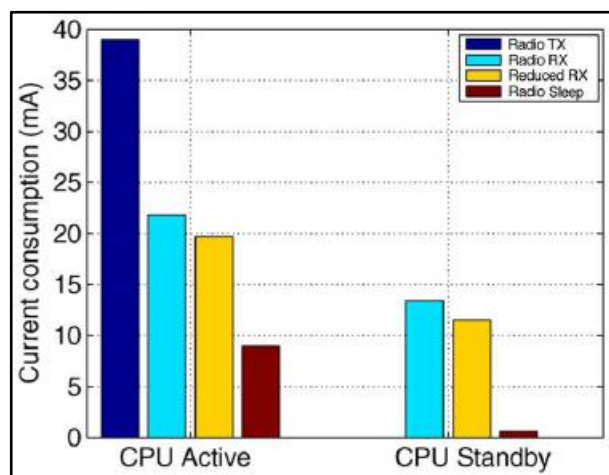


FIGURE 4. Total Current Consumption by Sensor Node Core Components

2.4.3 Cost Effective

The cost of a single node is significant in determining the overall cost of the networks as the networks may consist of enormous number of nodes [6]. Thus, this shows that it is important to take into account the overall budget in building WSN which cost spent is majorly dependent on the large amount of sensor nodes. For instance, for domestic application of WSN such as for small cropland monitoring for marginal farming, it is not cost wise to implement those expensive environmental sensor nodes and access point.

Crossbow Berkeley Motes from Crossbow Technology may be one of the most versatile wireless sensor network devices available in the market today [10][12]. Crossbow motes can be set as the standard or benchmark as a popular wireless technology device package used for WSN application [14]. Thus, Crossbow motes can be compared relatively to PIC-based WSN network based on the aspect of cost.

The WSN solution packages offered by Crossbow are very expensive [10]. Table 1 below shows the price of the Crossbow components:




| | | | |
|-----------|---|--|---|
| Component |  MIB510 Serial Gateway |  MPR400 Mica2 Mote |  MTS300 Sensor Board |
| Price | \$95.00 each | \$150.00 each | \$120.00 each |

TABLE 1. Market Survey of Crossbow Motes

Whereas, the common market price for major components for the development of PIC-based sensor node and access point [15][16][17] is determined and roughly calculated to be far cheaper that of Crossbow. Note that the price is roughly converted from MYR to USD as the components are from local Malaysian

market. Table 2 below shows the cost of building a PIC-based sensor node and access point.





| | | | | |
|-----------|---|---|--|---|
| Component |  PIC16F877 Development Kit |  TX9902B Transmitter with PT2262 Encoder IC |  RX9926 Receiver with PT2272 Decoder IC |  MCP9701 Temperature Sensor |
| Price | \$25.00 each | \$5.00 each | \$5.00 each | \$1.00 each |

TABLE 2. Market Survey of PIC-based Nodes

As a conclusion, the price comparison as stated above shows that the development of WSN using PIC-based materials is found to be very cost effective. In addition, this low cost covers up the design, experimentation and testing which is very practical for early project prototyping. The total cost in building actual final prototype is expected to be more or less from the price margins as stated above.

2.5 Sensor Network Communication Architecture

Protocol stack is used by sensor nodes and access point [6]. The function of the protocol stack is to support cooperative tasks between the sensor nodes. This is to ensure authentic and smooth routing, efficient power management and integration of data with the networking protocol.

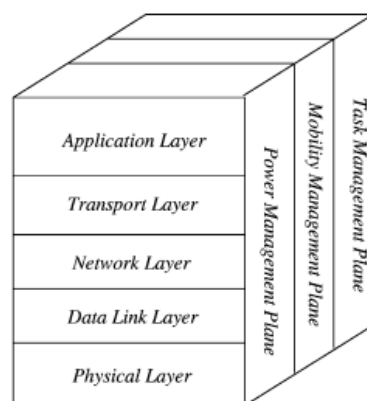


FIGURE 5. Sensor Network Communication Architecture [6]

The protocol stack comprises of *application layer, transport layer, network layer, data link layer, physical layer, power management plane, mobility management plane* and *task management plane*. At application layer, different types of application software can be used depending on the sensing task. The transport layer maintains the flow of data for the sensor network application. Usually, it is needed when the system is needed to be accessed through internet or other external network. The network layer handles the routing from data supplied by the transport layer. The data link layer which usually consisting of MAC (Medium Access Control) must be able to minimize collision with the environment noise and neighbours' broadcast. The physical layer demands for simple but robust modulation, transmission and receiving techniques.

In my project, the physical layer consists of first stage which is the ASK/OOK analogue modulation by the RF modules and the second stage which is the digital modulation by using Manchester Encoded data. It involves the design of transmitting and receiving techniques of the data frame by using specific PIC components such as timers and interrupts. The data link layer involves the control for the integration of other sensor nodes in the single-point sensor network without conflicting with each other. This will be the next step of the project.

Besides that, power, movement and task distribution between the sensor nodes are monitored by the power, mobility and task management planes. The power management plane tells the node how to use its power. For example, after the sensor node has transmitted the data, it can turn off its transmitter to save energy and prevent getting false or duplicate messages. Another example is that the sensor node can tells the access point that it is in low power level so that the nodes can only do sensing task without routing for other nodes (for multi-hop network). The mobility plane detects the sensor nodes' neighbours for maintaining the route back to the access point so that the power and task usage can be balanced. The task management plane then supervises and schedules the sensing tasks among the nodes. This is important so that the nodes can work in power efficient manner by sharing resources between them.

2.6 Scheduled-based Protocols or Time Division Multiple Access (TDMA)

There are three categories for existing MAC protocols (data link layer) in traditional wireless networks which are *scheduling-based*, *collision-free* and *contention-based*. This protocol functions as the mechanism to avoid data collision between the nodes to the access point [19]. A collision can occur when two nodes send data at the same time by using the same shared medium. For this project, simple TDMA-like protocol will be used to schedule the transmitting time of the two sensor nodes.

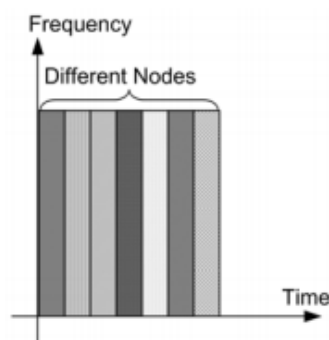


FIGURE 6. Time Division Multiple Access (TDMA) [19]

Scheduling-based protocols use centralized scheduling algorithm that decides the time at when the node can start its transmission. TDMA [20][21][22] is a scheduling-based protocol that gained much interest in wireless networks. Basically, in a period of time, TDMA is done by dividing the shared channel into N time slots in which each time slot has time duration. In each time slot, only one node is allowed to transmit the data in the single frequency channel.

CHAPTER 3: METHODOLOGY/PROJECT WORK

3.1 Methodology

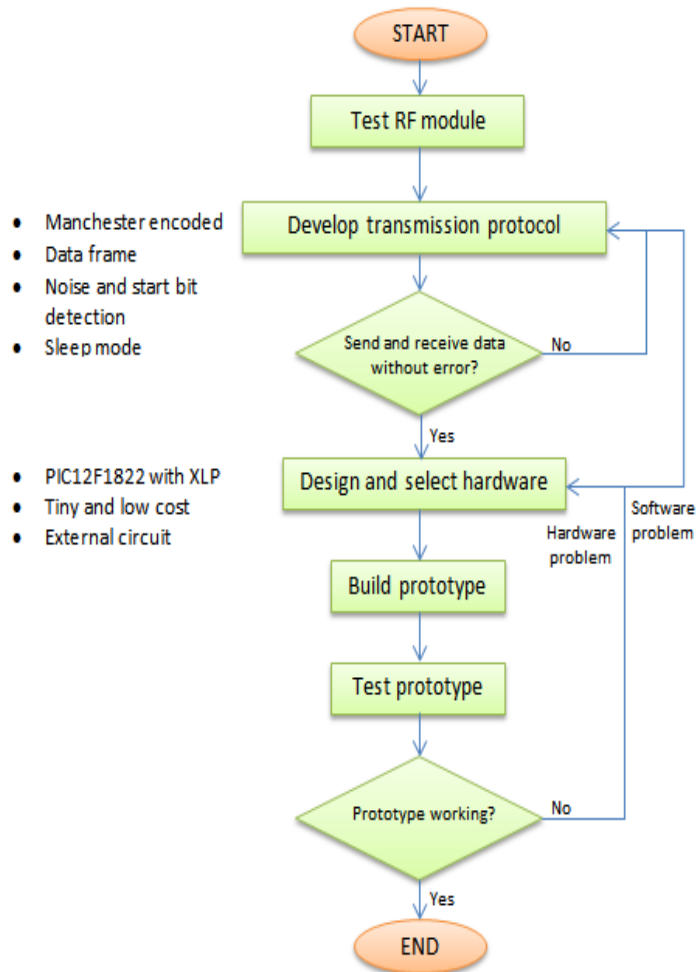


FIGURE 7. Flow Chart for Prototype Development

Figure 7 shows the flow chart for the development of sensor node and access point prototype. The development of the project starts with the testing of the cheap 433 MHz ASK/OOK RF modules which are RF-TX-433 transmitter and RF-RX-433 receiver. The performance of the RF modules is tested in terms of input/output signal from the receiver, noise, operating voltage and working distance. The characteristics of the RF modules is important for further project step which is the design of the data frame which can compensate with the drawbacks of the low cost RF modules.

The project work involves hardware and software development. For the software development, the early phase of the project is to develop data frame which will be encoded into Manchester encoding. The data transmission between the nodes

and access point is tested by sending repeated stream of data frame. Note that one frame is sent for every cycle of data transmission before the PIC goes to sleep.

When the basic transmission of data is approved to be working, sleep mode is then integrated into the sensor nodes. This sleep state of the PIC is important in conserving energy and minimizing current consumption when the node is not sending data. During this sleep state, other nodes are allowed to send the data one node at a time to avoid data collision and interference. This is because the transmitters share the same 433 MHz channel of the ASK modulation.

The integration of the sleep mode made the receiver to not receive data for a period of time and this contribute to the high frequency noise spikes which are picked up and produced at the receiver data out pin. To solve this problem, at the sensor node, header bit is integrated in the data frame to isolate the start of frame pulse which is the sync bit so that the sync bit is detected correctly by the PIC of the access point. Timer1 gate feature of the PIC is used to the filter the sync bit from the noise so that the sync bit can be detected correctly at the access point to decode the signal back from Manchester to actual data.

After the software development for reliable data transmission with sleep integration is successfully designed, the project moves to hardware selection and prototype development. 8-pin PIC12F1822 is used at the sensor node as it is small enough for tiny sensor node development. Besides cheap, the PIC model also has eXtreme Low Power (XLP) technology built in for the lowest current consumption as possible during operating and sleep mode. For the access point, PIC16F882/887 is used because of its high pin number for the LCD input and more importantly, it has the Timer1 gate function for the noise filtering to detect the sync bit correctly.

3.2 Overview of Project Work

For the early development of the project, the work involves the testing of the RF module performance by injecting some signals to transmitter to observe the digital output at the receiver's end. The physical layer protocol for simple RF wireless transmission such as data frame and basic asynchronous communication protocol is studied and designed.

The work continues in developing basic coding using Hitech-C for Manchester encoder at the node and corresponding Manchester decoder at the access point using PIC. At the node, the generation of Manchester encoded data is produced by using Timer0 overflow interrupt which is a built-in feature in PIC. By using Timer0 overflow interrupt service routine, the basic data frame consisting of start bit, data and stop bit can be generated almost precisely. Noted that to ensure smooth development of the project, the initial step is to build working transmitter and receiver to send constant data and not from the sensor. The integration of the sensor to the node is done later when the basic wireless transmission between the PIC's is proved to be working. At the access point, the Manchester decoder also uses Timer0 overflow interrupt and external edge interrupt to decode the data. At the first week of semester of FYP2, the author has successfully developed basic wireless transmission of one byte between the node and access point using Manchester as digital modulation.

One method to save the node's power is to introduce sleep mode to the PIC. Instead of sending continuous stream of data frames, the node is instructed to enter sleep mode for several seconds after sending one data frame. This improves the power consumption as when PIC enters sleep mode, internal components such as timers, internal oscillator and ADC module is OFF until the node wakes up. Problem arises when the receiver does not receive any data during node's sleep time and random noise is picked up by the receiver.

The first approach to solve the noise problem is by using low pass filter to filter the high frequency noise using the cut-off frequency. Sallen-Key 2nd order Butterworth active low pass filter is designed using op-amp IC LM358. The filter manages to filter some of the noise but the long high signal produces from the receiver cannot be rejected. The filter output produces the average of the high signal. The average signal is almost same level as the high pulses of the data and this can produces problem for the decoding process by the access point.

Another approach to solve the problem is by using CCP (Capture/Compare/PWM) module of the PIC to measure the pulse width of the signal

and rejecting the short pulse width of the signal. This is done by using the Capture interrupt mode sourced from CCP1 pin. The value of Timer1 is captured automatically to CCPR1H register for every low-to-high (start) and high-to-low (end) of the pulse. This module manages to filter almost all the noise but there is a problem where the first pulse of the data frame is missing.

At the access point, Timer1 Gate control is a software configurable PIC feature that not all PICs have it. The usage of PIC16F877A is changed to PIC16F887 to use the feature. By using Timer1 with gate control, pulse width can be measured without using interrupt. Besides that, instead of detecting the receiver output state at the initial and final duration using two interrupt service routine, it is more reliable to measure the pulse width of the sync bit. This is because the high state of the noise can give false state to the routine input check. T1G pin is used as the gate input. When low-to-high transition is detected, Timer1 is started and stopped automatically when there is high-to-low transition. Then Timer1 value is checked for the desired pulse width. Other than this value, the signal is rejected or ignored. An experiment using Timer1 gate application to check the pulse width is conducted and the result is successful and verified.

After the experiment of detecting the sync pulse is successful, the next work is to integrate the Timer1 gate coding to the developed Manchester decoder coding. Before this, the access point detects for idle time and low pulse of the start bit to verify the data frame. But by using Timer1 gate as the sync bit detector, the access point constantly checks for the 20ms sync bit pulse and if detected, the program continues to the decoding instructions.

The project is then focused on the sleep mode for the integration of other nodes to the access point. Firstly, WDT overflow is used to wake up the PIC from sleep. The result is that the nodes transmissions collide with each other after some time because the WDT timing is not accurate and drifts away. The nodes transmission overlaps and produces interference which corrupts or blocks the data from being received by the receiver.

To solve this problem, timer1 overflow is used to wake up the PIC from sleep. To generate timer1 overflow interrupt during sleep mode, Timer1 must be configured to be working in asynchronous counter mode. This mode is achieved by connecting external crystal oscillator for the timer1 count source. To generate accurate real time clock, crystal oscillator of 32.768kHz is introduced to the timer1 oscillator circuit pin of T1OSI and T1OSO. This crystal oscillator can produce sleep mode of 2 to 16 seconds based on the software timer1 prescaler setting. The nodes are no longer colliding with each other and can send the data one by one when other nodes are in sleep mode.

Based on the result obtained, it is desirable to change the 40-pin PIC model of PIC16F877A to 8-pin PIC12F1822 for the sensor node. Using PIC12F1822, smaller and cheaper sensor node can be obtained. With XLP technology and wide operating voltage of 1.8V to 5.5V, this PIC can save more power for longer operating time for one time battery usage.

3.3 Manchester Encoding for Serial Digital Communication

One of the well-known and tested serial communication protocols which had been applied to various low rate data transfer application is Non-Return-Zero (NRZ) or Non-Return-Zero-Inverted (NRZI). NRZ/NRZI is the simplest method to transmit digital serial data by using separate clock and data line. This method is synchronous which uses separate clock channel of fixed frequency to synchronize with the data. It is done by latching the actual data with the rising or falling edge of the clock.

The disadvantage of using this kind of digital modulation is that it is needed to have other separate channel for the synchronization clock. The elimination of the clock channel is desirable but introducing bit error problem when phase shift between the transmitter and receiver occurs. It is critical for the receiver to maintain almost perfect synchronization of internal clock with the transmitter. Other limitation of NRZ/NRZI is that the possibility of bit encoding error is high when sending long one state data. This is because it is difficult to detect the bit boundaries which have no transitions.

Digital to digital modulation or encoder that will be used in this project is Manchester encoding. Asynchronous serial communication is used instead of synchronous as the NRZ/NRZI modulation because for this project it is desired to maintain low cost wireless communication of using one ASK frequency channel from the cheap RF modules. Besides that, Manchester is used to encode the data because it has several advantages over other modulation. The advantage is that as it is self-clocking which can retrieve the transmitter clock from the data, Manchester encoding reduces bit error when long string of zeros and ones are sent. In terms of modulation energy, zero DC offset can be achieved as one bit of data is represented by two opposite phase shift. The application of Manchester through current wireless or transmission device can be seen in Ethernet and IEEE standard.

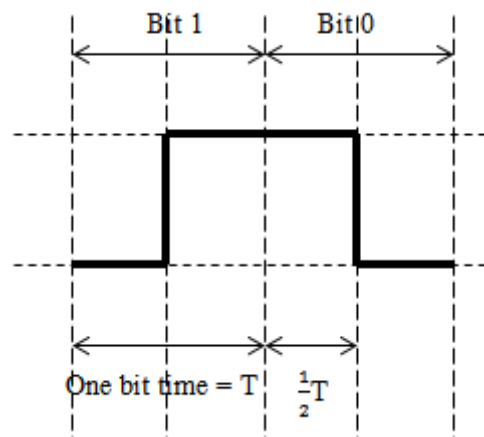


FIGURE 8. Manchester Encoded Data for Bit 1 and Bit 0

Manchester encoding is generated from phase modulation technique. It consists two opposite phase which represents one data bit. Based on the Figure 11, negative to positive transition means data bit 1 while positive to negative transition means data bit 0. The left Manchester bit is the clock while the right one is the data.

3.4 Data frame

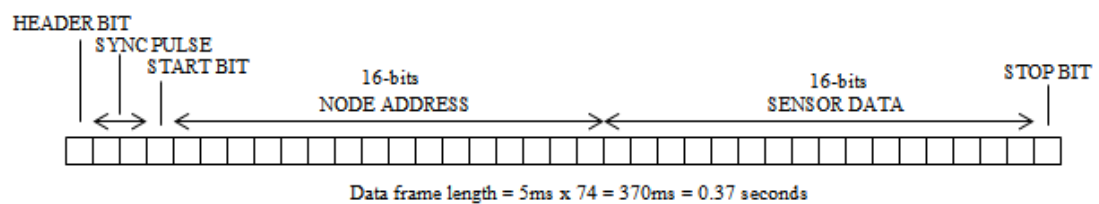


FIGURE 9. Data Frame

Figure 32 shows the data frame for one cycle of data transmission. Each box shows one bit of actual data which consists of two Manchester bits of 5ms each. The data frame consists Manchester bits of 2 bits of header bit, 4 bits of sync pulse, 2 bits of start bit, 32 bits of node address, 32 bits of sensor data and 2 bits of stop bit. The data frame length is 0.37 seconds. This short data frame is necessary to reduce the possibility of data collision from other nodes when the sleep time is short.

3.5 Overview of Manchester Encoding and Decoding of Data Frame

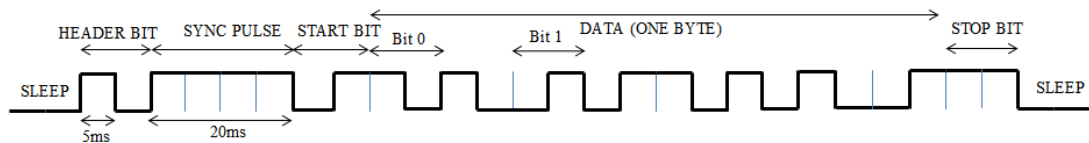


FIGURE 10. Manchester Encoded Data Frame

Figure 33 shows the Manchester encoded data frame. At the sensor node, Manchester bits of 5ms each are generated from Timer0 overflow interrupt to generate Manchester encoded data frame. To generate 5ms of interrupt, Timer0 is preloaded with 217 counts before it is started. The function of header bit is to isolate the data frame from the noise so that the sync pulse can be detected correctly. The sync pulse is 20ms wide and acts as the start of frame (SOF) pulse. This is to tell the receiver that the positive transition after the sync pulse is the centre of the start bit.

At the access point, 20ms SOF pulse is detected by using timer1 gate at theRB0 pin. Timer1 is started when high pulse is detected and stops when low pulse is encountered. The value of the Timer1 is compared to 10000 counts which equals to 20ms duration. In the software, if the value falls in between count range of 10000 to 11000, the sync pulse is successfully detected.

The function of the start bit is to start the sampling of the data. At the transition of the start bit which is detected by the external edge interrupt, Timer0 is started to generate overflow interrupt and sample the Manchester bits for actual data. In other words, after Timer0 overflows in 7.5ms, the actual data is sampled. After the sampling, at the same time timer0 is configured to generate another interrupt on overflow for 5ms. If no edge or transition is detected, it is the stop bit and the data

sampling is finished. When an edge is detected and edge interrupt is generated, timer0 is started again for 7.5ms overflow and the sampling is repeated over again. The sampling process stops when stop bit is detected which is there is no transition after 5ms period.

3.6 Sensor Node Hardware Schematic

Figure 34 shows the hardware schematic of the sensor node. The sensor node consists of PIC12F1822 microcontroller, power supply from two 3V cell battery, RF-TX-433 433Mhz ASK/OOK transmitter and LM35 temperature sensor.

PIC12F1822 is an 8-pin microcontroller from Microchip. The tiny feature of this low-pin PIC is to save space for the development of small-sized wireless sensor node. This is to make it possible to be deployed at difficult areas for data acquisition application such as at the top of trees, high corners of plantation greenhouse or inside the wild animal cage. The sensor node needs only small number of I/O so it is not desirable for high-pin microcontroller which can contribute to higher cost development.

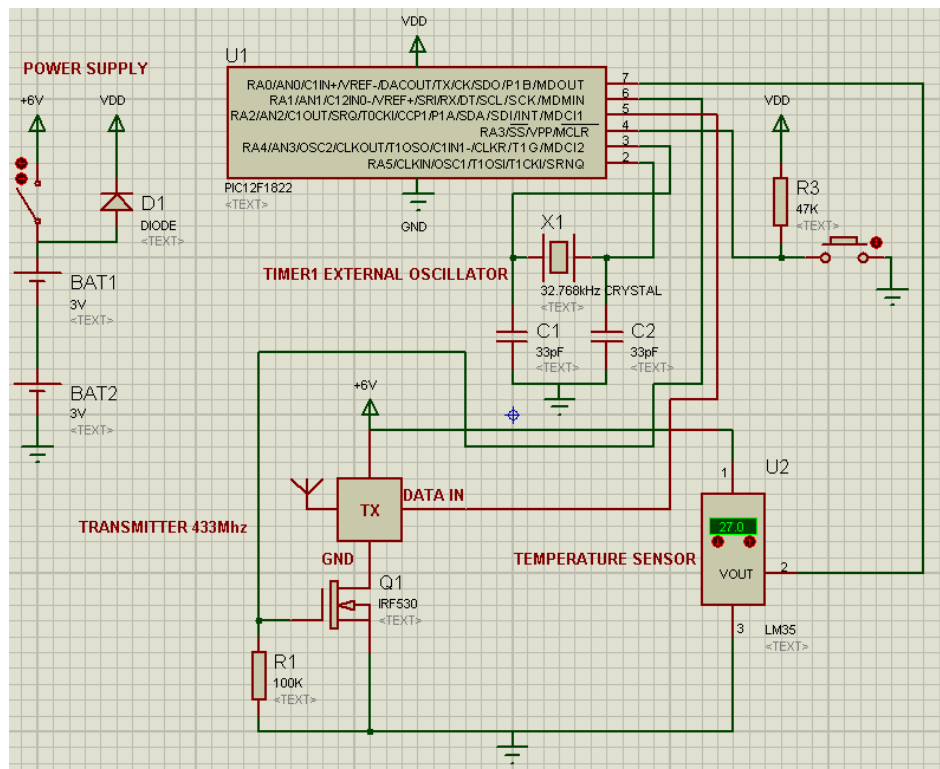


FIGURE 11. Sensor Node Hardware Schematic

The function of PIC is to encode the designed data frame into Manchester encoded data. The data frame will consist of the temperature sensor value resulted from ADC conversion by the PIC. By using Timer0 overflow interrupt, the bit length of the Manchester signal can be generated precisely and accurately. This is crucial as wireless asynchronous transmission depends basically on the time frame of the data.

PIC12F1822 is a new model from Microchip which uses eXtreme Low Power (XLP) technology. For very low power application, the PIC can be operated at low voltage which is ranging from 1.8V to 5.5V. As provided by Microchip, the typical current consumption at 1.8V operating voltage for this model is as follows:

- Sleep mode: 20nA
- Watchdog Timer: 300nA
- Timer1 Oscillator: 650nA at 32 kHz
- Operating Current: 30 μ A/MHz

It can be calculated that at 5.3V, the approximate PIC current consumption is 58nA during sleep mode, 1.91 μ A for Timer1 oscillator and operating current of 60 μ A at 2MHz using internal oscillator. Note that watchdog timer is not used by the node and it is turned off permanently to save energy.

PIC12F1822 has built in Timer1 oscillator circuit between the T1OSI and T1OSO pins. These pins can be connected with external 32.768kHz crystal oscillator so that Timer1 can operate in asynchronous counter mode in order be able operating when the PIC sleeps. This is necessary to generate Timer1 overflow interrupt to wake up the PIC from sleep. This frequency of crystal oscillator is the real time clock frequency which can produce accurate 2 to 16 seconds overflow based on the software configured prescaler. Timer1 is configured to have prescaler of 1:4 and 8 seconds of timer1 overflow is obtained to wake the IC from sleep. Based on previous testing, the wake-up from sleep due to WDT overflow of is not accurate. It drifts away and after some time collides with the other node data transmission and interference occurred. The integration of 32.768kHz external oscillator to Timer1 prevents this problem and accurate 8 seconds of sleep can be achieved.

The sensor node captures the surrounding temperature and transmits the sensor reading to the access point wirelessly by using 433MHz transmitter. The transmitter is modulated by using ASK/OOK modulation with maximum bit rate of 10kbps. The temperature sensor used in the sensor node is LM35. It produces voltage output proportional to the surrounding temperature which is 10mV/degC. The Vref+ for the ADC is connected to the 1.024V internal fixed voltage reference of the PIC. This stabilized low voltage reference can produce more accurate and reliable ADC result in the cost of lower maximum temperature ADC sensing capability which is 102degC instead of 150degC.

The external components which are transmitter and temperature sensor are enabled or disabled by switching on or off the power supply from the components. The purpose of switching off the components is to save energy from the current drawn by the component as it is not necessary during sleep mode. The temperature sensor and transmitter are allowed to be on starting from the ADC conversion by the PIC and after one frame of data is sent. Then, before the PIC enters sleep all peripherals and I/O are turned off and brought low except timer1 oscillator circuit. MOSFET IRF530 is used to control power supply of 6V to the transmitter and temperature sensor.

3.7 Access Point Hardware Schematic

Figure 35 shows the schematic of the access point. The access point consists of PIC16F882 microcontroller, 433MHz receiver and LCD module. PIC decodes the Manchester encoded data from the transmitter and displays it on the LCD display module. The PIC uses internal oscillator of 2MHz for the system clock.

The receiver's output is fed to the RB0 pin which carries two functions which are INT and T1G. The T1G or Timer1 gate pin function is to detect the low-high-low pulse for the 20ms sync pulse width detection. After the sync pulse is detected, the pin function is configured to be INT which will generate interrupt on external edge detection and call interrupt service routine function when particular low-to-high edge transition is detected. This interrupt will start the Timer0 for overflow interrupt to sample the data. After data bits of the data frame are decoded back into actual data,

INT function will be stopped and T1G function is turned on again to detect the 20ms sync pulse (SOF).

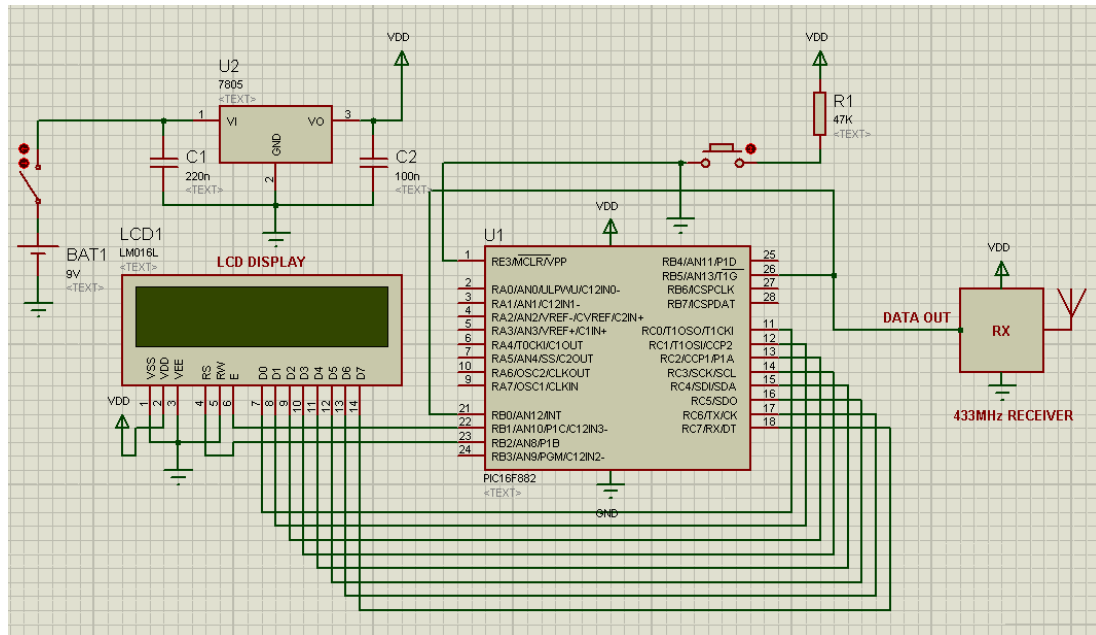


FIGURE 12. Access Point Hardware Schematic

3.8 Sensor Node Software Design

Figure 13 shows the main program flow chart for sensor node. The program starts with the initialization of the system, constants, variables and functions. The external circuit which consists of LM35 sensor and transmitter are switched on to perform the analogue to digital conversion and transmit the data wirelessly. After the ADC is performed, the ADC result and the sensor node address are put in a buffer. The data in the buffer is then encoded into Manchester data to produce data frame consisting of header bit, sync bit, start bit and stop bit for the asynchronous digital data transmission.

After one data frame is sent, the external circuit is switched off to minimize current consumption during sleep mode of PIC. Before the PIC is put to sleep, Timer1 is turned on to generate interrupt on overflow which is after 8 seconds. The PIC enters sleep mode and after wake up, it resumes the next instruction by switching on the external circuit and start the whole process again.

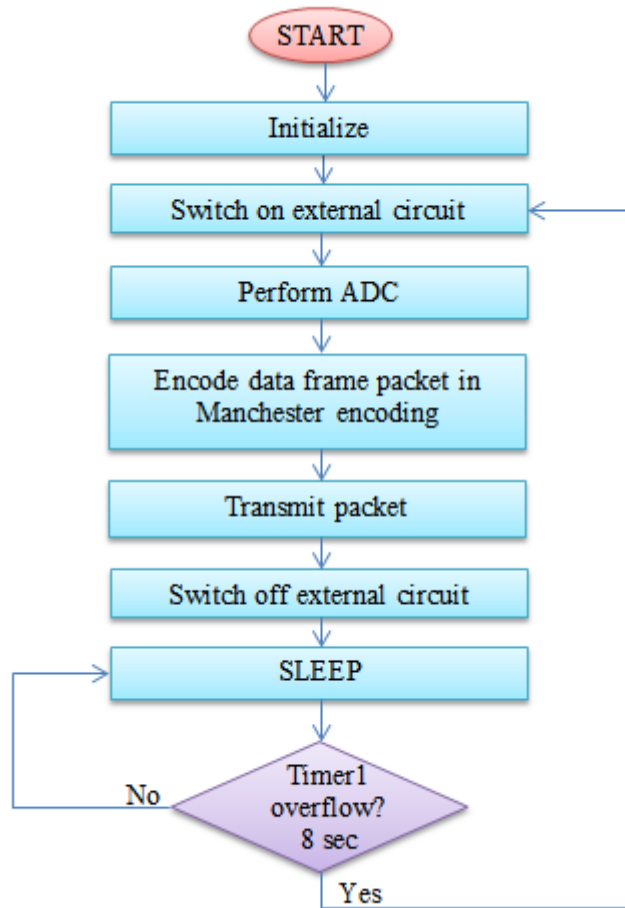


FIGURE 13. Flow Chart for Sensor Node

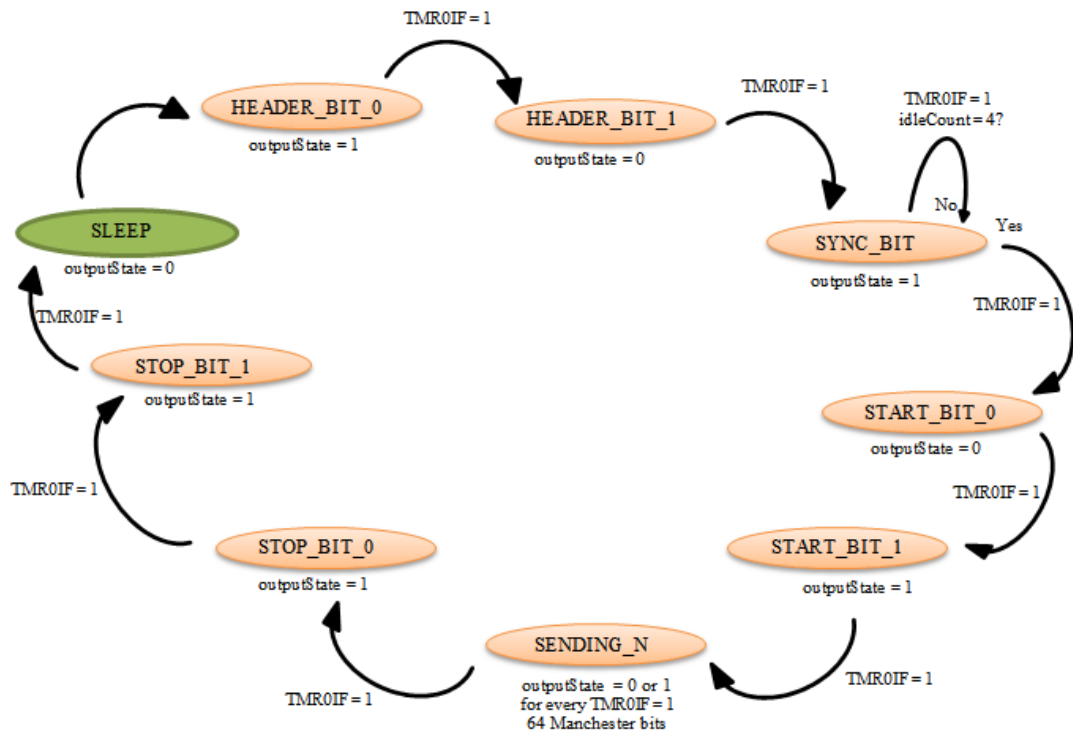


FIGURE 14. FSM Diagram for Manchester Encoded Data Frame

Figure 14 shows the finite-state-machine (FSM) diagram for the generation of Manchester encoded data frame. After the PIC initializes and carries the functions, it will encode the 16-bit sensor node address and 16-bit ADC data to generate 32 Manchester bits into the data frame which consist of Manchester bits of 2 for header bit, 4 for sync bit, 2 for start bit, 32 for data and 2 for stop bit.

The PIC enters sleep mode and wake up again to encode another data frame for next data transmission. In each state which will be entered when Timer0 Interrupt Flag bit is set ($TMROIF = 1$), the output of the RB2 pin which will be fed to the data input of the transmitter will become low or high for 5 ms based on the outputState or INT_OUT. By loading 217 into Timer0, 5 ms interrupt is generated. Then, interrupt service routine is fetched, perform some instructions to produce RB2 pin output and clears TMROIF again for next Timer0 overflow interrupt.

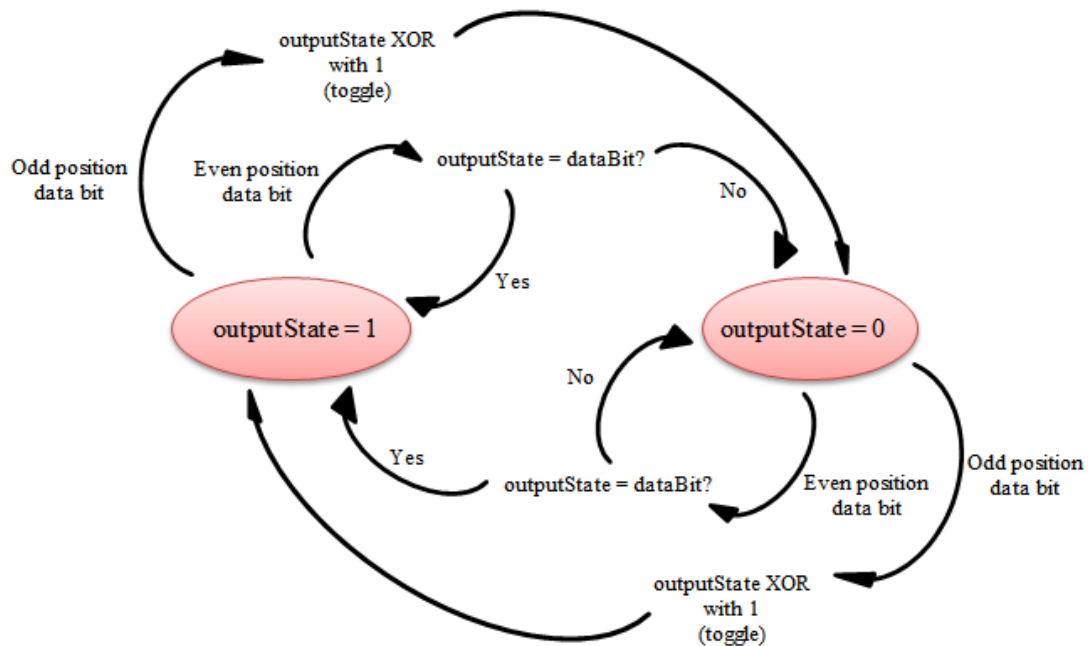


FIGURE 15. FSM Diagram for SENDING_N State

Figure 15 shows the FSM diagram for the SENDING_N state. It is important to explain the scheme of 32-bit data encoding into 64-bit Manchester bits using Timer0 overflow interrupt. After the start bit is produced, the last state of the PIC output is 1. So, the PIC enters SENDING_N state with outputState equals to 1. The

interrupt routine is started by testing for the position of the Manchester bits either odd or even. If it is the first Manchester bit it is in even position, if it is the second Manchester bit it is in odd position, for third Manchester bit it is in even position again, for fourth Manchester bit it is in odd position again and so on.

For current outputState of 1 (which is from start bit of state 1), for even data Manchester bit generation, the MSB of 16-bit data of the buffer is masked to get the single bit dataBit. If dataBit equals to outputState, the outputState is toggled (new current outputState is 0). If dataBit is not equal to outputState, the outputState is not toggled (new current outputState is maintaining 1). After that for odd data Manchester bit generation, the new outputState is XOR with 1 or toggled.

3.9 Sensor Node Software Design

Figure shows the main program finite-state-machine diagram for the access point. The function is mainly to decode the Manchester data and displays it on the LCD module. The output from the receiver will be fed into two PIC input pins which are RB0/INT which for the external edge interrupt and RB5/T1G for Timer1 gate source. Timer1 gate source is the function that the Timer1 counts automatically when high transition is detected and stops immediately after low transition occurs. This function is important to measure the pulse width accurately and directly without using Compare/Capture/PWM (CCP) interrupt. Three PIC component functions which are external edge interrupt, Timer1 gate and Timer0 overflow interrupt are used to decode the Manchester encoded data frame.

The decoding program starts in IDLE mode. In IDLE mode, high transition is polled to start Timer1 and stops counting when low transition is detected. After the pulse is measured, the counts in the 16-bit TMR1 register is compared to 10000 to 11000 counts which is equals to 20 ms. If 20 ms pulse which is sync bit is detected, the decoding process is started and repeated again until all the Manchester bits are decoded or the stop bit is encountered. The actual data which consists of 16-bit source address and 16-bit sensor data is put in an array and it is checked for the sensor node address. The source address for sensor node 1 is 0b101010101 while 0b00000001 for sensor node 2. Sensor node 1 data is displayed

on the first row while for sensor node 2 data is displayed on the second row of the LCD module.

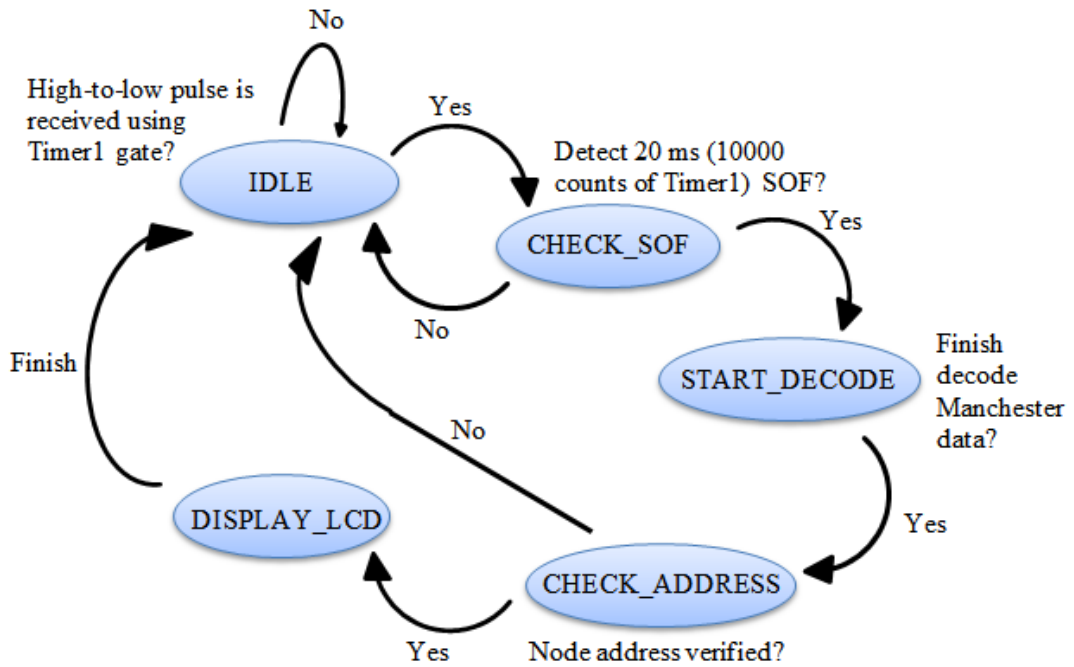


FIGURE 16. FSM Diagram for Access Point

Figure 16 shows the FSM for the Manchester decoding scheme. In ENTER_INTERRUPT state, Timer0 overflow interrupt (TMR0IE) and external edge from RB0/INT pin interrupt (INTE) is both enabled. Note that INTEDGE is set to 1 to detect high edge at the centre of start bit. TMR0IF equals to 1 means that Timer0 interrupt is triggered and INTF equals to 1 means that external edge interrupt is triggered. It is vice versa for 0 value. If TMR0IF is set, it means there is no data (start bit) detected 10 ms after the sync bit and the state will move to FINISH_DECODE state to indicate that decoding is finished. The access point will start again to enter ENTER_INTERRUPT state and decode the next data received.

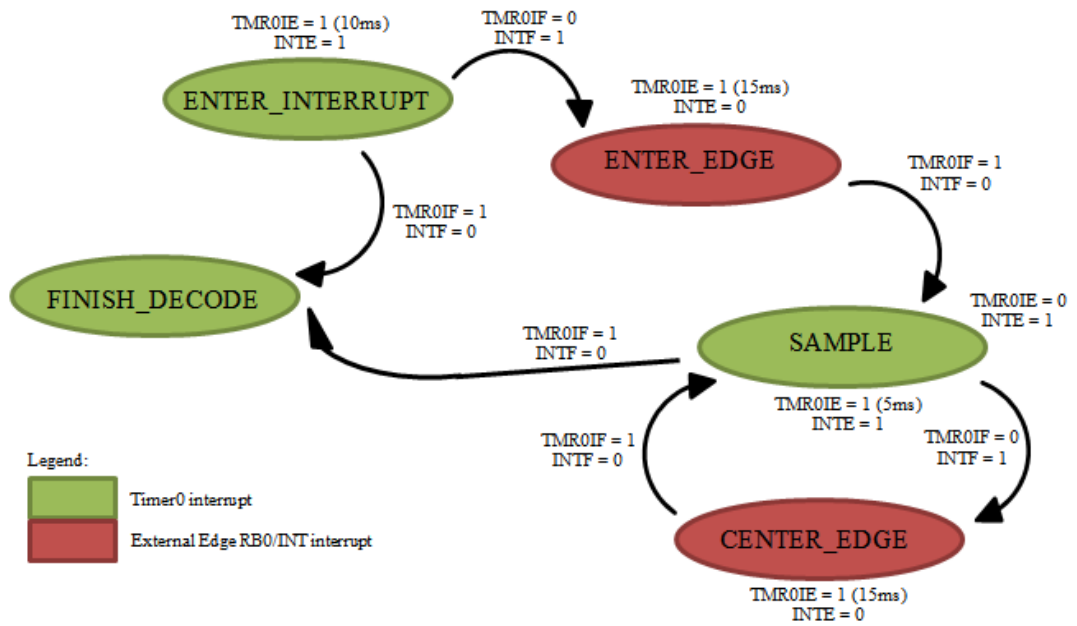


FIGURE 17. FSM Diagram for Manchester Decoding

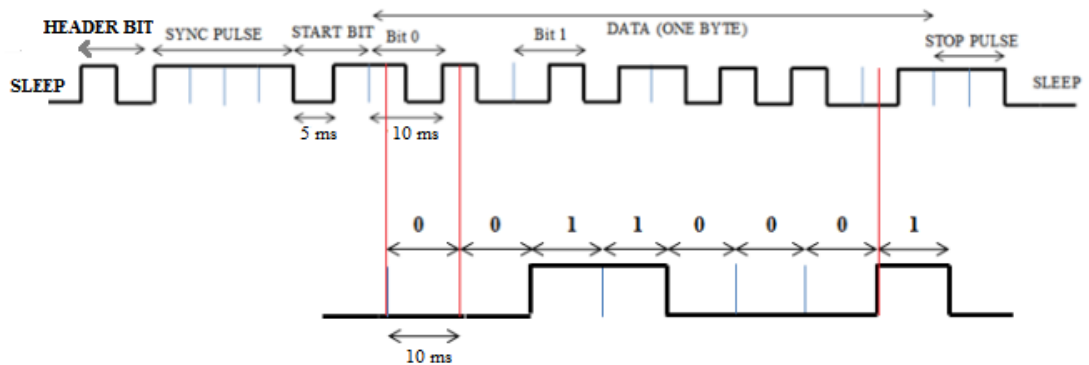


FIGURE 18. Data Frame Decoding Output

If INTF is set, external interrupt routine is called at SAMPLE state and ready to sample the data by loading 197 (15ms) into Timer0 register and starting the timer. After 15 ms, Timer0 interrupt is generated and the data bit is sampled. If the value of the data detected is 0, receivedByte which LSB is 0 is XOR with 1 and the actual sampled bit is 1. Then, receivedByte is shifted one place to the left and has LSB of 0 again. If the value of data detected is 1, the actual sampled bit is zero and receivedByte is shifted again to sample other bit. During the bit sampling, Timer0 is started again to generate interrupt for 5ms. If another edge is detected, the process is repeated again and enters CENTER_EDGE state. If Timer0 interrupt is triggered, it means it is the stop bit and the decoding process is finished. The program will loop to start again and enter ENTER_INTERRUPT state to decode the next data received.

3.10 Gantt Chart

| Steps | FYP1 | | | | | | | | | | | | | FYP2 | | | | | | | | | | | | |
|---|------|---|---|---|---|---|---|---|---|----|----|----|----|------|---|---|---|---|---|---|---|---|----|----|----|----|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| Data gathering & Analysis (Hardware & Software) | █ | █ | █ | █ | | | | | | | | | | | | | | | | | | | | | | |
| Hardware Selection | | | | █ | █ | █ | | | | | | | | | | | | | | | | | | | | |
| Design | | | | | █ | █ | █ | █ | | | | | | █ | █ | █ | █ | █ | | | | | | | | |
| Early Prototype Construction | | | | | | | | █ | █ | █ | █ | █ | | | | | | | | | | | | | | |
| Functional Testing | | | | | | | | | | | █ | █ | | | | | | | | █ | █ | | | | | |
| Final Prototype Construction | | | | | | | | | | | | | | | | | | | | | █ | █ | █ | | | |
| Simulation | | | | | | | | | | | | █ | | | | | | | | | | | | █ | | |
| Data gathering & Analysis | | | | | | | | | | | | | | | | | | | | | | | | █ | █ | |
| Documentation | | | | | | █ | | | | | | █ | █ | | | | | | | | | | | █ | █ | |

TABLE 3. Prototype Development Gantt Chart

3.11 Tools

Proteus ISIS

Wherever possible, ISIS is used to verify and simulate the circuit design. It is very helpful as ISIS contains large number of components in its libraries. This software is easy and provides the needed components in simulating the hardware circuit in real time. By using this software, time and cost in developing the program and hardware circuit can be saved as the debugging and troubleshooting can be simulated and confirmed before moving to the real hardware development.

Eagle and ELECTRA

Eagle software is used to create node circuit schematic and board layout for the PCB development. To make the routing process easier and effective, auto-routing software ELECTRA is used for the purpose. The board layout file from Eagle is used into ELECTRA for auto-routing process. The resulted script is then exported and imported back into the board layout in Eagle.

MPLAB IDE/Hitech-C

In programming the PIC, Hitech-C is used to write the program in C language. C language is used instead of assembly language because it is easier to write and debug. The compiler then creates hex file of the program to be uploaded into the PIC. Before that, PIC Simulator IDE and Real PIC Simulator are used to simulate and verify the program output to make sure it is working as desired.

CHAPTER 4: RESULTS AND DISCUSSION

4.1 Experiments and Project Work Result

4.1.1 Testing of the Transmitter and Receiver Performance

The RF transmitter and receiver used for this project are TX-433 and RX-433 respectively. It is cheap China made RF module from Cytron Technologies. The RF module uses ASK/OOK and acts as the first layer or physical layer of the wireless transmission. The useful feature is that the receiver produces TTL digital output as fed to the transmitter. This output can be connected directly to the PIC for further processing. The information of the transmitter and receiver used in this project as in the datasheet is as following.

TX-433 Transmitter

Pin functions:

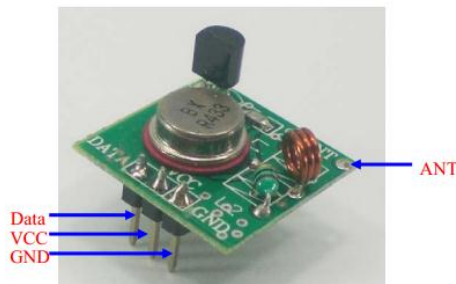


FIGURE 19. TX-433 Transmitter

| Label | Description |
|-------|------------------------------------|
| Data | Data pin |
| VCC | Power supply |
| GND | Ground |
| ANT | Hole to solder and connect antenna |

| Specifications | RF Transmitter |
|-------------------|--|
| Operating voltage | 3V to 12V |
| Operating current | Max $\leq 40\text{mA}$ (12V), Min $\leq 9\text{mA}$ (3V) |
| Oscillator | SAW (Surface Acoustic Wave) oscillator |
| Frequency | 433.92MHz |

| | |
|--------------------|---------------------------------|
| Frequency error | $\pm 150\text{kHz}(\text{max})$ |
| Modulation | ASK/OOK |
| Transfer rate | $\leq 10\text{Kbps}$ |
| Transmitting power | 25mW (at 12V) |
| Antenna length | 18cm |

RX-433 Receiver

Pin functions:

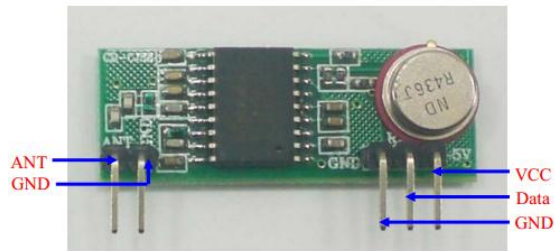


FIGURE 20. RX-433 receiver

| Label | Description |
|--------------|--|
| VCC | Power supply (5V) |
| GND | Ground (The 2 GND are internally connected each other) |
| Data | Data pin |
| ANT | Hole to solder and connect antenna |

| Specifications | RF Receiver |
|-----------------------|---------------------------------------|
| Operating Voltage | $5.0\text{V} \pm 0.5\text{V}$ |
| Operating Current | $\leq 5.5\text{mA} @ 5.0\text{V}$ |
| Operating Principle | Monolithic super heterodyne receiving |
| Modulation | OOK/ASK |
| Frequency | 433.92MHz |
| Bandwidth | 2MHz |
| Sensitivity | -100dBm |
| Rate | $< 9.6\text{Kbps}$ (315MHz @ -95dBm) |
| Data Output | TTL |
| Antenna length | 18cm |

The testing involves output signal check by using an oscilloscope at the receiver data pin when the transmitter is transmitting some signals. As the data output from the receiver is TTL, it can be connected directly to the microcontroller for further data decoding and processing. This testing is importance in determining the behaviour of the cheap RF module in terms of noise immunity (from the receiver sensitivity) and received data integrity (erroneous data or not) to develop the necessary coding scheme for the PIC microcontroller.

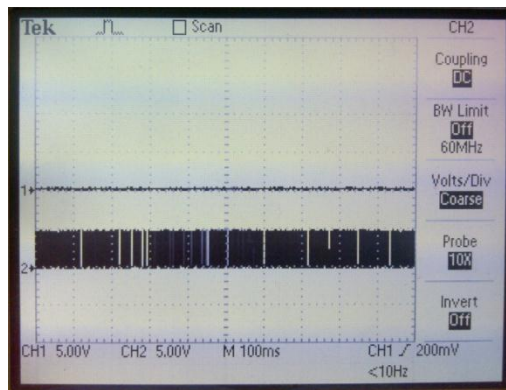


FIGURE 21. Receiver Output from Transmitter Sending Data of ZERO or Transmitter is OFF

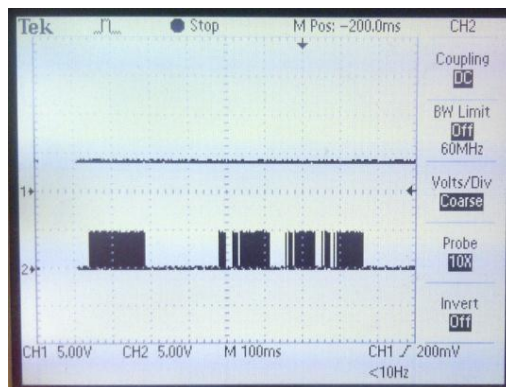


FIGURE 22. Receiver Output from Transmitter Sending Data of ONE

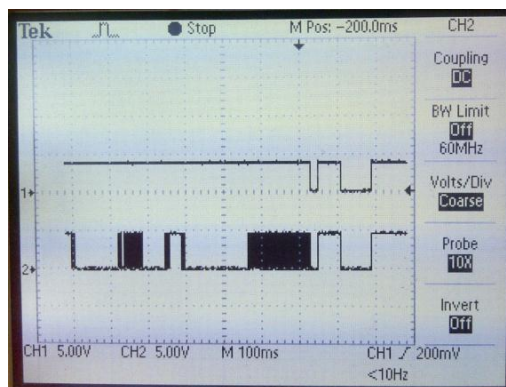


FIGURE 23. Receiver Output from Transmitter Sending Long and Short Data Bit of ONE

Based on the figures, when the transmitter is sending constant zero and constant one the receiver output is very noisy. The receiver tends to pick up external noise by producing short pulse of voltage spikes of 5V. Figure 3 shows that short data bit pulse width from the transmitter is received perfectly but too long data bit pulse width tends to pick up random noise. Based on this result, careful design of data frame must be introduced to overcome this problem.

4.1.2 Sending Constant Data of One Byte without Sleep Mode

The early development stage of the project is to successfully send one byte of constant data instead of sensor data to the receiver. This is to test for the basic wireless communication of the cheap RF modules integrated with the PIC by using Manchester encoded data frame. Note that sleep mode is not integrated yet to the receiver. As there is conflict to the receiver data output when sleep mode is introduced to the transmitter (due to sensitivity of the receiver when receiving long constant zero or one), the code development of the program is firstly focused on the program development for transmission of fast repeated data with short gap between the data frame. The program development is to test the functionality of the node Manchester encoding and the access point Manchester decoding.

For digital modulation, the data that will be sent from the transmitter using RF ASK/OOK will be encoded to Manchester. After received at the receiver, it will be decoded back into actual data by the microcontroller. The use of PIC as the encoder/decoder is possible because beside flash-programmable, PIC also has several special functions such as built-in timers, timer with gate control, internal overflow/external interrupt, Watch Dog Timer (for sleep function) and ADC. This is important for simple but reliable data processing, encoding and decoding.

To practically demonstrate the wireless transmission of Manchester encoded data between the transmitter and receiver, the equipment is setup as Figure 16. The distance between the sensor node and access point is about 6m. Repeated data of character '1' is sent to the access point. For the early testing purpose and fast Manchester coding development, PIC development kit SK40C is used for both node and access point. The PIC used is PIC16F877A.

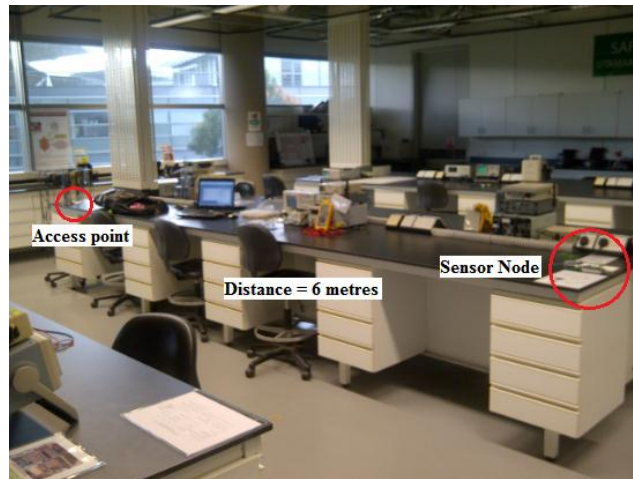


FIGURE 24. Overview of Experimental Setup

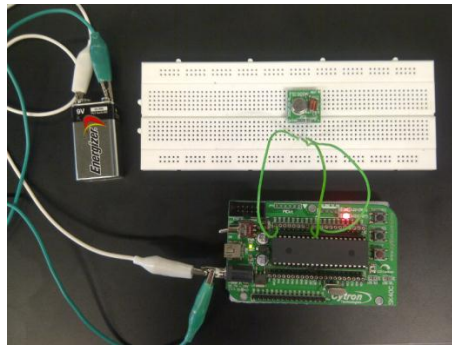


FIGURE 25. Sensor Node

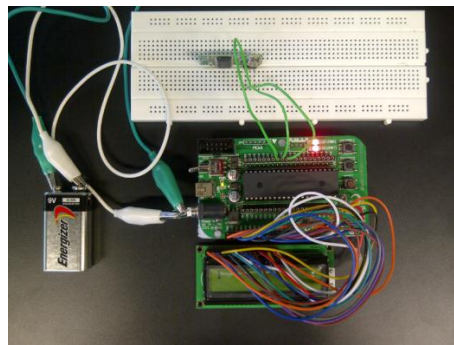
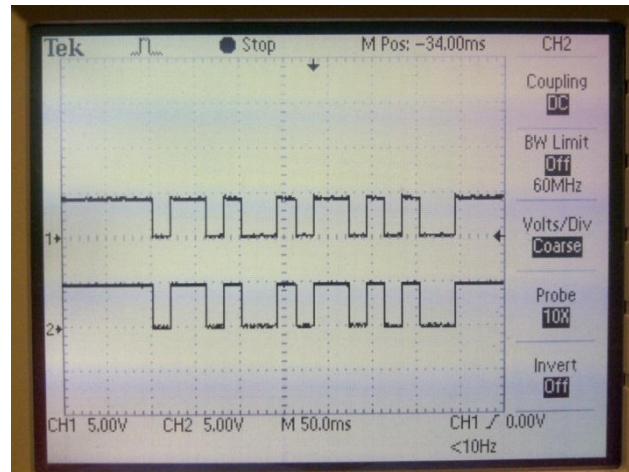


FIGURE 26. Access Point

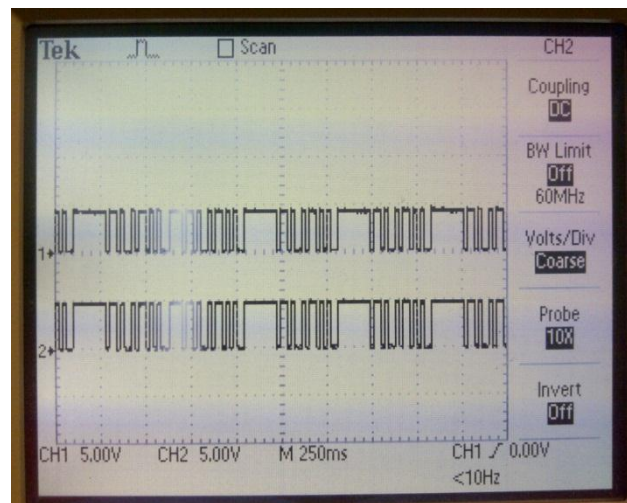
Figure 19 and 20 of Channel 1 shows the input signal to the transmitter. The signal is generated from the node's PIC microcontroller from pin RB7 and received. It is the signal of Manchester encoded data for character '1' which equals to binary 00110001. Each of the data bit is 20ms generated by using TMR0 overflow interrupt. The first two bit low and high is the start bit of the data frame. The next 16 bits are the data bits of binary 1010010110101001 representing one byte of actual data. The

high signal after the data bits shows the sensor node is in idle state. Figure 19 and 20 of Channel 2 is the output signal from the receiver. From the result, it can be said that Manchester encoded signal is successfully generated and received perfectly at the receiver end.



Channel 1: Transmitter input
Channel 2: Receiver output

FIGURE 27. Sending Manchester Encoded Data of Character ‘1’ or Binary 00110001



Channel 1: Transmitter input
Channel 2: Receiver output

FIGURE 28. Longer Sweep Time from FIGURE 16. Repeated Constant Data Frame with Gap of 200ms (idle time) in Between

To verify the correct decoded signal at the receiver end, the actual decoded output data from the receiver’s PIC is generated at RB1 pin. Figure 21 shows the oscilloscope result at RB1 pin producing actual data of 00110001. Noted that each data bit produced is 40ms in which two times the Manchester encoded bit of 20ms. There is a delay of 10ms between the Manchester data and produced actual data as centre of the Manchester bit is sampled to get the actual data.

To demonstrate the binary decoded data into character '1', LCD is used as the output from PIC microcontroller. The LCD data input is fed from PORTD of the PIC. Figure 10 shows string '1' is produced at the LCD output. This shows that wireless transmission of sending constant data of 8 byte between node and access point is successful.

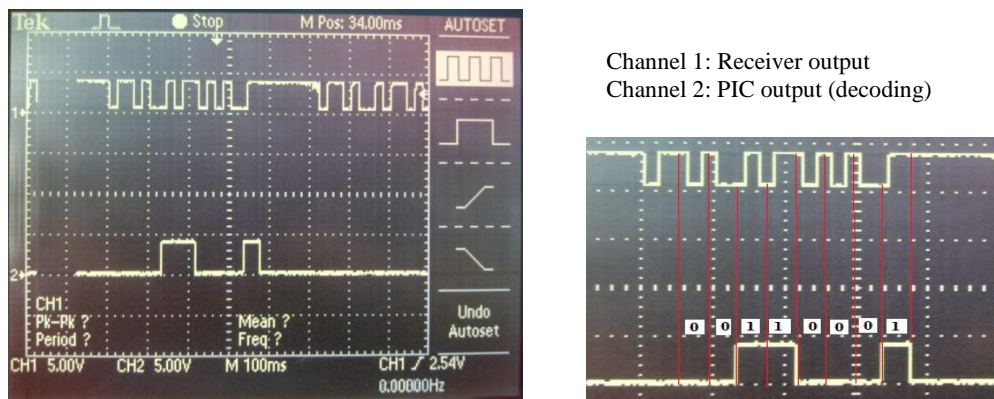


FIGURE 29. Decoded Output Signal from PIC

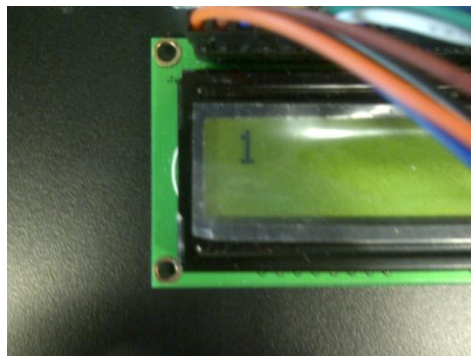


FIGURE 30. Character '1' is Produced at LCD Output

4.1.3 Sending Constant Data of One Byte with Sleep Mode

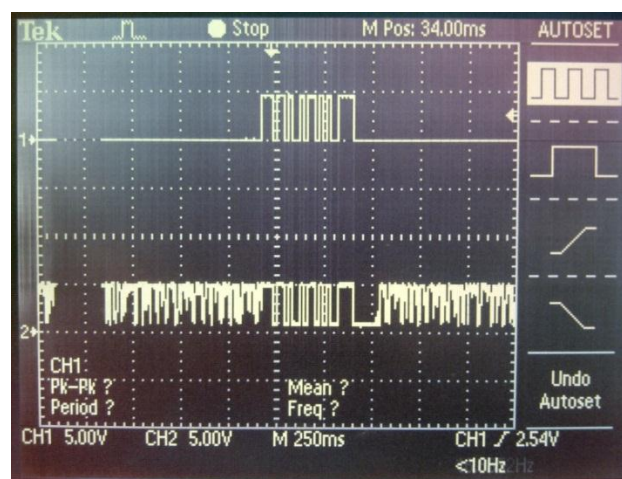
To consume less power, the node must be programmed to send data for every 2 or 3 seconds. In achieving this, Watch Dog Timer (WDT) is used to generate sleep time of 2.3 seconds by using prescaler of 1:128. In the coding, Hitech-C macro function of SLEEP() is used to enter sleep mode. After WDT overflows, the software continues its current instruction. By using WDT with prescaler, Timer0 is no longer assigned to the prescaler of 1:256 to generate 20ms of interrupt. When Timer0 uses no prescaler, it can only produce a very short time of interrupt. Because

of this, the author has tried to use 100us for each Manchester data bit pulse. The result is the receiver output receives false data as the bit rate is very high. This shows that the RF module must use low bit rate like 20ms to improve its reliability.

To overcome this problem, Timer1 is used for the generation of overflow interrupt to produce corresponding Manchester encoded data. Timer1 can also generate overflow interrupt like Timer0 except for it is 16-bit instead of 8-bit of Timer0 (capable of getting longer interrupt time with usage of same prescaler) and has other special functions. So, to achieve 20ms interrupt Timer1 does not have to be assigned to any prescaler due to its higher counts to overflow.

The Improvement of Data Frame by Adding Suppress Noise Byte or Header Bit

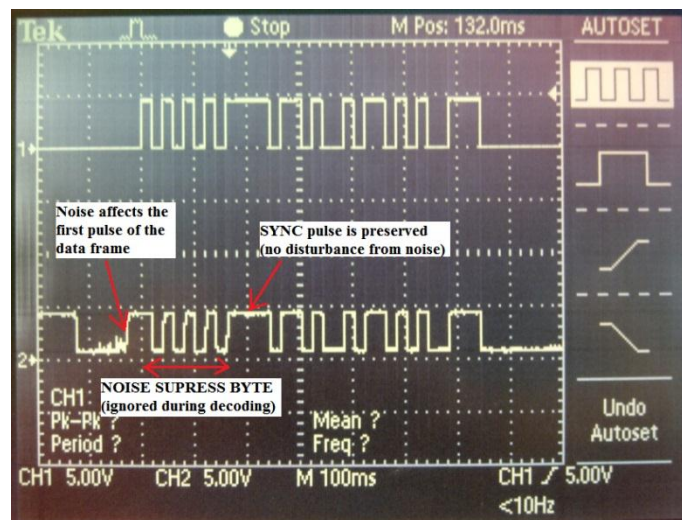
With the sleep mode integrated in the design, another problem arises at the receiver's end. This is because the PIC's data output to the transmitter input is pulled down (zero) when the PIC is in sleep mode. This causes the receiver does not receive any signal for the sleep time period and tends to pick up random noise as shown in Figure 25. This noise also disturbs the data pulse when it triggers near the first pulse of the data frame. This gives problem in verification of low start bit by detecting at least 60ms of idle time before the start bit. This is due to the high frequency pulse of 5V noise. At the access point, the PIC detects these high noise spikes when it checks for the receiver's output for idle time. Consequently, wrong start bit is detected after the false idle time detection and this produces erroneous decoded data.



Channel 1: Transmitter input
Channel 2: Receiver output

FIGURE 31. Noise Affecting the Data Frame

Based on the receiver output testing, the noise only exists at very long constant zero or one. High-low data with short pulse width is received perfectly at the receiver's end. This means only the pulse at the start of the frame is affected by noise. The Manchester encoded data after that is preserved. To overcome the problem of noise affecting the first pulse (Idle pulse) of data frame, one byte of data 10101010 called Noise Supress byte will be included in the data frame. This byte will separate the noise from the start bit. The result is as shown in the figure.



Channel 1: Transmitter input
Channel 2: Receiver output

FIGURE 32. Isolating Sync Pulse from the Noise

The data frame is improvised as to compensate the problem raised from noise due to the integration of sleep mode. This is because the receiver does not receive any data for long time due to transmitter is off from sleep mode.

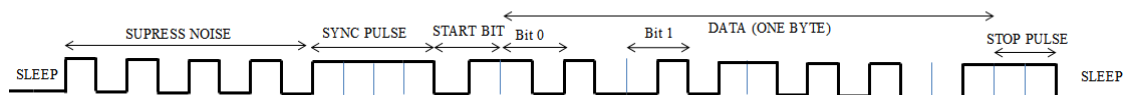


FIGURE 33. Data Frame with Suppress Noise Byte

Figure 23 shows the improvised data frame. The data frame consists of 32 bits or 4 bytes. Each of the bits is the Manchester bit of 20ms. The data frame is sending one byte of data. The actual data bit is 40ms which comprises of two Manchester bits. The data frame consists of Suppress Noise pulses consists of 8 bits,

Sync pulse of 4 bits (80ms), Start bit of 2 bits (LOW and HIGH), one data byte and Stop bit of 2 bits.

For final development of the data frame, the suppress noise byte which consists of 8 bits of Manchester encoded data is decreased to 2 bits only. The purpose is to shorten the data frame so that the transmitter is on for shortest time as possible to transmit the data frame before the PIC goes to sleep. This is to reduce the possibility of signal collision between node 1 and node 2 which can cause interference in the shared 433MHz frequency band.

The Improvement of Detecting Sync Bit by Using Timer1 Gate Function

As stated earlier, the PIC wrongly detects high frequency 5V pulses as high signal and this disturbs the detection of the high 80ms Sync pulse. Previously, Timer0 interrupt is used to generate interrupt at the starting and the ending of 80ms period to check for the sync pulse. This gives problem as the two times interrupt routines only check for the start and end of the signal without verifying it is constant high pulse or not.

In solving this problem, Timer1 with Gate control is used to detect the 80ms sync pulse. As not all PICs have this Timer1 Gate control feature, the access point uses PIC16F887 instead of PIC16F877A. This allows the PIC to directly time external event using T1G as source input pin of RB5 transitions without any usage of Capture-Compare interrupt which complicates more the coding. The function of this Timer1 gate in this project is to capture the pulse duration of the sync pulse. This is done by capturing the value of Timer1 starting from low-to-high transition, stopping from high-to-low, resetting back Timer1 value, waiting again for another low-to-high transition and starting again the process. So, if value of Timer1 is 40000 to 40500 for 80ms to 81ms of sync pulse, the sync pulse is verified and encoding process of the next byte starting from the start bit transition is continued. If not verified, the Sync pulse detection process is started again until another sync pulse is found. To demonstrate the functionality of Timer1 Gate to detect high pulse of 20ms, a test is conducted with PIC's coding and the result is as follows:

```

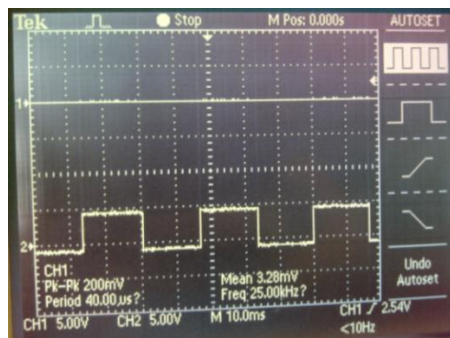
void setup(void)
{
  PORTC = 0;
  CM1CON0 = 0;
  CM2CON0 = 0;
  TIGSS = 1;
  TRISE = 0b00100000;
  TMR1 = 0;
  T1GINTV = 1;
  TMR1GE = 1;
  TMR1ON = 1;
  ANSELH = 0;
  ANSEL = 0;
  RBC = 0;
}

void main(void)
{
  setup();
  while(TRUE)
  {
    RBC = 0;
    while(!RBS);
    while(RBS);
    captureValue = TMR1;
    TMR1 = 0;

    if(((captureValue >= 40000) || (captureValue <= 40500)) //40000 to 40500 accept 80ms to 81ms only)
    {
      RBC = 1;
      delay_ms(80);
    }
  }
}

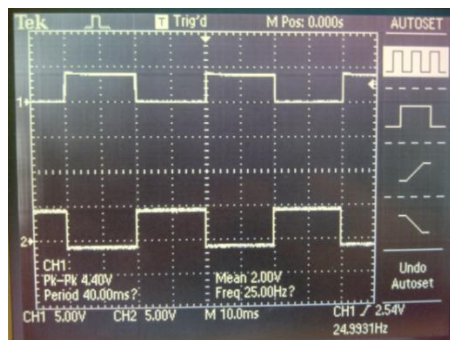
```

FIGURE 34. Timer1 Gate for Sync Pulse Detection Coding



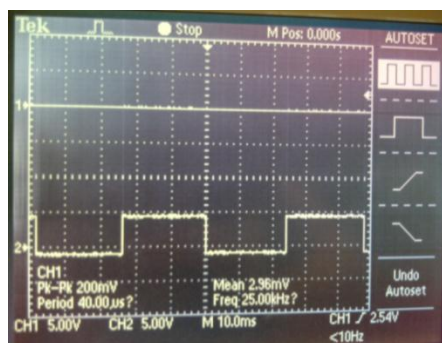
Channel 1: Output
Channel 2: Input

FIGURE 35. Timer1 Gate Rejecting Input Pulse of Less Than 20ms



Channel 1: Output
Channel 2: Input

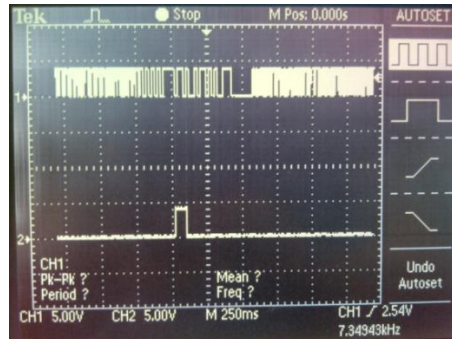
FIGURE 36. Timer1 Gate Detecting Input Pulse of 20ms



Channel 1: Output
Channel 2: Input

FIGURE 37. Timer1 Gate Rejecting Input Pulse of More Than 20ms

To show the PIC manage to detect 80ms sync pulse and reject other signal less than 80ms including the noise, the node is transmitting the Manchester encoded data with sleep mode of 3.2 seconds. The result is as follows:



Channel 1: Input
Channel 2: Output

FIGURE 38. Timer1 Gate Detecting 80ms Sync Pulse

4.1.3.3 The WDT Timing Drifting Problem of Sensor Node in Sleep Mode

Figure 33 shows the data frames from two sensor nodes. Each node is configured for 8 seconds sleep. To avoid collision, the nodes are switched on at different time which after node 1 is switched on, Node 2 is started 4 seconds later. After some time, the nodes' data frame is drifted slowly and interferes with each other. This problem is successfully solved by using Timer1 external oscillator of 32.768kHz real-time crystal oscillator clock.

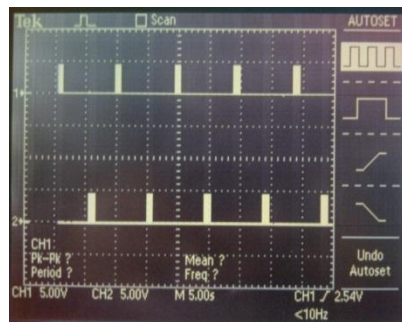


FIGURE 39. Initial Node 1 and Node 2 Data Frame Transmission

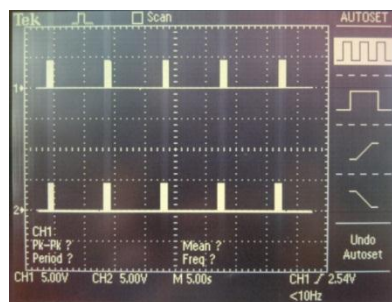


FIGURE 40. Node 1 and Node 2 Data Frame Interferes with Each Other After Some Time

4.2 Final Prototype Testing and Result

After experiments and testing are done to verify the data reliability of the sensor node and access point, the final hardware and software design are developed for final prototype development.

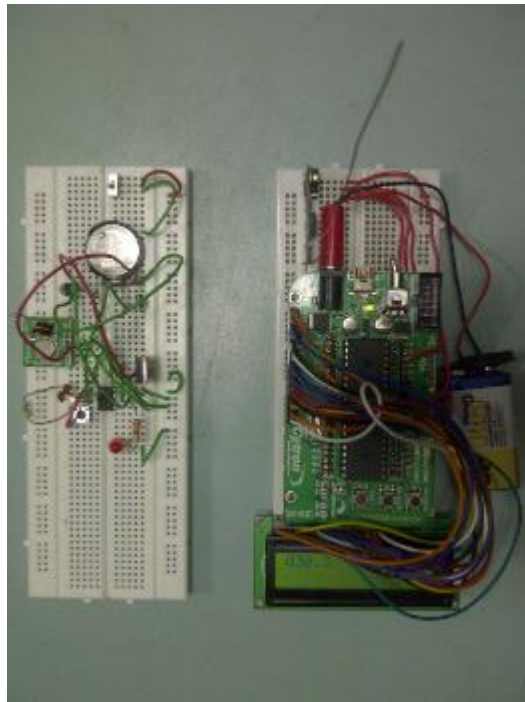


FIGURE 41. Successful Data Transmission Between Sensor Node and Access Point

Figure shows the sensor node is transmitting data successfully to the access point. The ADC conversion is verified to be true by reading the temperature sensor output voltage. By using multimeter, the voltage reading is 0.3228V and this shows the actual surrounding temperature is true as being displayed by the LCD which is 32.3 degC. As designed, the sensor reading from sensor node 1 is displayed at the first row of the LCD.

PIC12F1822 is used for the sensor node while PIC16F887 is used as the access point. The sensor node uses two pieces of 3V CR2016 Maxell watch battery which produce total supply voltage of 6V for the transmitter and temperature sensor. A diode is used to drop the voltage to 5.3V for the PIC V_{DD} . For smaller sensor node, the final design will be developed in PCB.

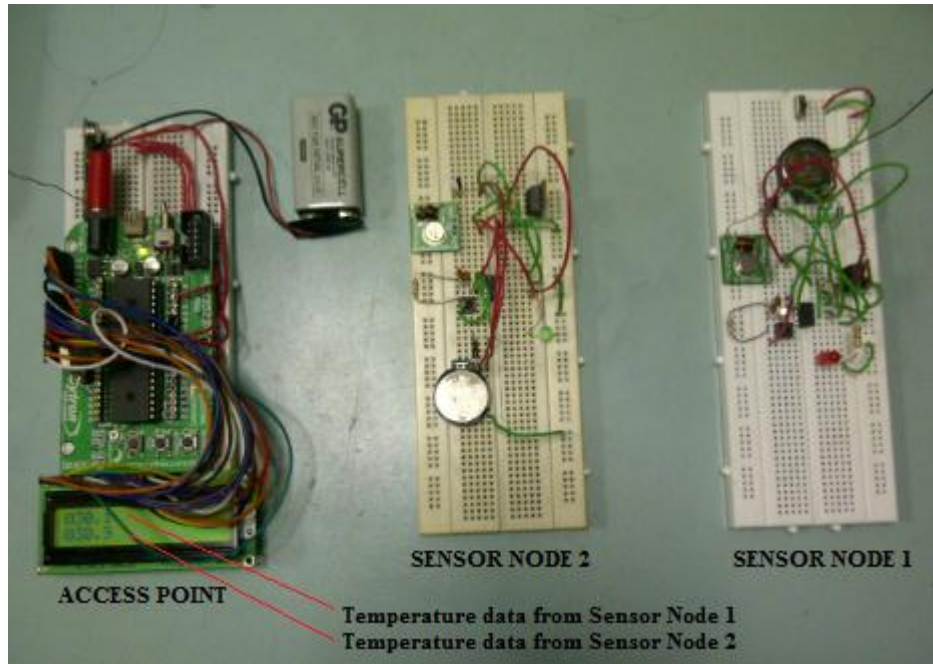


FIGURE 42. Successful Data Transmission Between Two Sensor Nodes and Access Point

Figure shows the testing conducted to verify the data transmission between two nodes. Sensor Node 2 is turned on 4 seconds after Sensor Node 1. This means the access point will receive new data every 4 seconds from each node. Note that each node sleeps for 8 seconds by using the real time clock from 32.768 kHz crystal. The nodes are left to transmit data for 2 hours and no data collision occurs. This result verifies the solution of using the real time crystal instead of the PIC WDT timer to wake up the PIC from sleep accurately without on-period drifting.

The maximum transmission distance of the sensor node and access point is about 20m with the usage of 18cm antenna and 6V transmitter supply voltage. The testing place is in closed room. Without antenna, the working distance is reduced to 13.8m only. The strength of the prototype design is that the wireless communication between the sensor node and access point does not require any line of sight to transmit the data.

For easy placement and mobilization, the sensor nodes are made tiny in size by developing the circuit design into PCB as shown in Figure 43.



FIGURE 43. Final Prototype of Sensor Nodes Developed into PCB

| Sensor node components | Price |
|-------------------------|----------------|
| PIC12F1822 | RM5.00 |
| RF-TX-433 Transmitter | RM20.00 |
| LM35 Temperature Sensor | RM8.00 |
| 3V cell battery × 2 | RM7.00 |
| 32.768kHz crystal | RM2.00 |
| TOTAL | RM40.00 |

TABLE 4. Total Cost for Sensor Node

| Access point components | Price |
|-------------------------|-----------------|
| PIC16F887 | RM26.00 |
| RF-RX-433 Receiver | RM30.00 |
| SK40C | RM40.00 |
| 9V battery | RM9.80 |
| 16 × 2 LCD | RM20.00 |
| TOTAL | RM125.80 |

TABLE 5. Total Cost for Access Point

Table 4 and Table 5 shows the total cost of the components for sensor node and access point respectively. Cheap sensor node of under RM50 cost is successfully developed. By using smaller PIC and without SK40C development, cheaper access point can be build.

CHAPTER 5: CONCLUSION AND RECOMMENDATION

At the end of the project, it can be verified and concluded that cheap and reliable sensor node with significant power saving features and access point is successfully developed. For wireless mobile low power sensor node operation, sleep mode and low power PIC selection are important in improving the energy consumption so that longer battery life can be achieved. By using the common PIC features such as timers, interrupts and pin functions, a reliable but cost effective sensor node and access point can be built to compensate the noisy cheap RF modules.

However, there is restriction for the application of more than one sensor node to the access point. The sensor nodes must be manually configured to be turned on one by one and not at the same time to avoid interference between them. Longer sleep mode must be introduced to the sensor node so that more nodes are allowed to use the frequency channel one node at a time. In addition, the data frame must be not long enough as this will make the transmitter turned on for longer time. This will bring interference problem as nodes are more possibly to collide when short sleep time is used.

The recommendation and future work is to use two pair of transmitter and receiver to act as one transceiver to the PIC. The above restriction can be solved when the sensor node and access point has the ability to transmit and receive so that acknowledgement signal can be send and obtained to precisely synchronize the nodes transmission schedule. Besides that, without using pair of transmitter and receiver, real time clock using DS1307 IC can be retrieved by the PIC for real time nodes transmission time. This can avoid the transmission drifts and collision between the nodes. In addition, further testing can be conducted to test the sensor node and access point performance in terms data integrity at larger distance. Precise power consumption measurement of the sensor node can be done to analyse the power saving feature of the sensor node more accurately.

REFERENCES

- [1] Hac, A. *Wireless Sensor Network Designs*. John Wiley & Sons Ltd: Etobicoke, Ontario, Canada, 2003.
- [2] W. Dargie, C. Poellabauer, 'Fundamentals of Wireless Sensor Networks: Theory and Practice', John Wiley & Sons, 2010, pp. 8.
- [3] Chong, C.Y. & Kumar, S.P. (2003). Sensor Networks: Evolution, opportunities, and challenges, *Proceedings of the IEEE* **91**(8): 1247 – 1256.
- [4] M. Winkler, K.D. Tuchs, K.Hughes, G. Barclay (2008). Theoretical and Practical Aspects of Military Wireless Sensor Network. *Journal of Telecommunications and Information Technology*, 37-44.
- [5]* Pennstate University Library. In *APA Quick Citation Guide*. Retrieved 25 June 2012, from http://www.libraries.psu.edu/psul/lls/students/apa_citation.html
- [6] Georgia Institute of Technology. (2001, December). Wireless sensor networks: A survey. Atlanta, USA: I.F. Akuildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci.
- [7] Cork Institute of Technology. (2004, June). PIC-based sensor operating system. Cork, Ireland: C. Lynch, F. O. Reilly.
- [8] Universiti Teknologi MARA. Development of a PIC – Based Wireless Sensor Node Utilizing XBee Technology. Shah Alam, Malaysia: Husna, Ruhani, F. Hanum.
- [9] University of Pittsburg School of Information Science. (2006, April). Energy consumption in wireless sensor network using GSP. Madellin, Colombia: M. G. C. Torres.
- [10] Keith. (n.d.). Beginners Guide to Crossbow Motes. Retrieved 24 June 2012, from: <http://www.pages.drexel.edu/~kws23/tutorials/motes/motes.html>
- [11] W.D. Kimpe. (2008, June). In *How To Design A Low Power Wireless Sensor Network*. Retrieved 25 June 2012, from <http://www.greenpeak.com/Press/PressKit/200806HowtoDesignLowPowerWSN.pdf>
- [12] Reuters (2008, January). In *Crossbow Unveils eKo Pro Series Wireless Crop Monitoring System at Unified Wine Symposium*. Retrieved 25 June 2012, from <http://www.reuters.com/article/2008/01/29/idUS204443+29-Jan-2008+BW20080129>

- [13] P. Corke, T. Wark, R. Jurdak, W. Hu, P. Valencia, D. Moore (2010). Environmental Wireless Sensor Networks, *Proceedings of the IEEE* **98**(11): 1903 – 1917.
- [14] Crossbow Technology (n.d.). Retrieved 24 June 2012, from <http://www.xbow.com>.
- [15] BizChip (n.d.). Retrieved on 25 June 2012, from <http://www.bizchip.com/kits.html>
- [16] Remote Control Parts (n.d.). Retrieved on 25 June 2012, from <http://www.escol.com.my/Sensors.html>
- [17] Digi-Key Corporation (n.d.). Retrieved on 25 June 2012, from <http://www.digikey.com/product-detail/en/MCP9701-E%2FTO/MCP9701-E%2FTO-ND/1212513>
- [18] Borneo Post Online (2011, December). In *Kementerian Sains Teknologi dan Inovasi Jalin Mou dan Moa Dengan Kerajaan Negeri Sabah*. Retrieved 23 June 2012, from: <http://www.theborneopost.com/2011/10/12/kementerian-sains-teknologi-dan-inovasi-jalin-mou-dan-moa-dengan-kerajaan-negeri-sabah/>
- [19] Polytechnic Institute of Porto. (2005, November). Lower Protocol Layers for Wireless Sensor Networks: A Survey. Porto, Portugal: A. Koubâa, M. Alves, E.Tovar.
- [20] D. J. Baker and J. Wieselthier, “A distributed algorithm for scheduling the activation of links in a self-organizing, mobile, radio network,” in Proc. Int. Conf. Commun., 1982, pp. 2F.6.1–2F.6.5.
- [21] D. J. Baker, A. Ephremides, and J. Flynn, “The design and simulation of a mobile radio network with distributed control,” *IEEE Trans. Select. Areas Commun.*, vol. SAC-2, pp. 226–237, Jan. 1984.
- [22] D. S. Stevens and M. H. Ammar, “Evaluation of slot allocation strategies for TDMA protocols in packet radio networks,” in Proc. IEEE MILCOM Conf., 1990, pp. 835–839.

Note: *The citation in APA format is based on the stated source.

APPENDICES

Sensor Node Program

```
#include<pic.h>
#include<htc.h>
#include<pic12f1822.h>
#include "mypic.h"
#include "adc.h"
__CONFIG(FOSC_INTOSC & WDTE_OFF & PWRTE_OFF & MCLR_ON & CP_OFF & CPD_OFF & BOREN_OFF & CLKOUTEN_OFF & IESO_OFF & FCMEN_OFF);
__CONFIG(WRT_OFF & PLLEN_OFF & STVREN_OFF & LVP_OFF);
#define _XTAL_FREQ 2000000

#define INT_OUT RA2
#define TMR0_T 217

#define HEADER_BIT_0 0
#define HEADER_BIT_1 1
#define SYNC_BIT 2
#define START_BIT_0 3
#define START_BIT_1 4
#define SENDING_N 5
#define STOP_BIT_0 6
#define STOP_BIT_1 7
#define DONE 8

void interrupt manchesterTx(void);
void setup(void);
void startTx(void);
void txPacket(void);
adc_celcius(void);
void startTxSensor(void);
void stopTxSensor(void);

unsigned int buffer[2];
unsigned int data;
char index;
char intCount;
char dataBit;
char outputState;
char machineState;
char txDone;
unsigned int *p;
char idleCount;

void txPacket(void)
{
    buffer[0]= 0b00000001; //source address
    buffer[1]= adc_celcius(); //temp sensor data
    p = &buffer[0];
    data = *p;
}

void setup(void)
{
    OSCCON = 0b01100010;
    T1CON = 0b10101100;
    __delay_ms(1);
    PORTA = 0;
    ANSELA = 0b00000001;
    TRISA = 0b00001001;
    WPUAO = 0;
    CCP1CON = 0;
    CM1CON0 = 0;
    CM1CON1 = 0;
    SDOSEL = 1;
    TXCKSEL = 1;
    P1BSEL = 1;
    T1GSEL = 1;
    CPSON = 0;
    DACCON0 = 0;
    DACCON1 = 0;
    OPTION_REG = 0b00000101;
    TMR0IE = 1;
}

void startTx (void)
{
    TMR1 = 0;
    index = 1;
    intCount = 0;
    INT_OUT = 0;
    machineState = HEADER_BIT_0;
    outputState = 0;
    txDone = FALSE;
    dataBit = 0;
    TMR0 = TMR0_T;
    TMR0IF = 0;
```

```

    TMROIE = 1;
    TMRIGE = 0;
    PEIE = 1;
    GIE = 1;
}

void startTxSensor(void)
{
    RA1 = 1;
}

void stopTxSensor(void)
{
    INT_OUT = 0;
    RA1 = 0;
}

adc_celcius(void)
{
    unsigned int adc_value, celcius = 0;
    adc_on();
    adc_value = ui_adc_read();
    celcius = adc_value;
    return celcius;
}

main(void)
{
    setup();
    adc_initialize();
    while(TRUE)
    {
        startTxSensor();
        startTx();
        adc_celcius();
        txPacket();
        while(txDone != TRUE);
        stopTxSensor();
        GIE = 0;
        TMR1IE = 1;
        TMR1IF = 0;

        TMR1ON = 1;
        SLEEP();
        TMR1ON = 0;
        TMR1IE = 0;
        TMR1IF = 0;
        GIE = 1;
    }
}

void interrupt manchesterTx(void)
{
    switch(machineState)
    {
        case HEADER_BIT_0:
            outputState = 1;
            machineState = HEADER_BIT_1;
            break;

        case HEADER_BIT_1:
            outputState = 0;
            machineState = SYNC_BIT;
            break;

        case SYNC_BIT:
            outputState = 1;
            idleCount++;
            machineState = SYNC_BIT;
            if(idleCount == 4)
            {
                machineState = START_BIT_0;
                idleCount = 0;
            }
            break;

        case START_BIT_0:
            outputState = 0;
            machineState = START_BIT_1;
            break;

        case START_BIT_1:
            outputState = 1;
            machineState = SENDING_N;
    }
}

```

```

break;

case SENDING_N:
    if(intCount & 0x01)
        outputState ^= 1;
    else
    {
        if(data & 0x8000)
            dataBit = 1;
        else
            dataBit = 0;
        if(dataBit == outputState)
            outputState ^= 1;
        data = data<<1;
    }
    ++intCount;
    if(intCount == 32)
    {
        intCount = 0;
        if(index < 3)
            data = *(p + index);
        ++index;
        if(index == 3)
            machineState = STOP_BIT_0;
    }
break;

case STOP_BIT_0:
    outputState = 1;
    machineState = STOP_BIT_1;
break;

case STOP_BIT_1:
    outputState = 1;
    machineState = DONE;
break;

case DONE:
    txDone = TRUE;
    TMR0IE = 0;
    outputState = 0;
break;

default:
    break;
}
INI_OUT = outputState;
TMR0 = TMR0_I;
TMR0IF = 0;
}

```

Access Point Program

```
#include<pic.h>
#include"mypic.h"
#include"led.h"
#include<pic16f887.h>
#include<htc.h>
__CONFIG(FOSC_XT & WDTE_OFF & PWRTE_OFF & MCLRE_ON & CP_OFF & CPD_OFF & BOREN_OFF & IESO_OFF & FCMEN_OFF & LVP_OFF & DEBUG_OFF);
__CONFIG(WRT_OFF);
#define _XTAL_FREQ 2000000

#define ENTER_INTERRUPT 0
#define SAMPLE 1
#define FINISH_DECODE 2

#define ENTER_EDGE 0
#define CENTER_EDGE 1
#define INPUT_RB0
#define OUT_RB1

#define CHECK_SOF 0
#define START_DECODE 1
#define CHECK_ADDRESS 2
#define TX1_DISPLAY_LCD 3
#define TX2_DISPLAY_LCD 4

#define TMR0_I 217
#define TMR0_2I 178

volatile char mainState, tmr0State, edgeState, bitCount, onePointFiveT;
volatile int receivedByte;
bit rxFinished;
unsigned int captureValue;

void setup(void);
void startRx(void);
void startTx(void);
void interrupt taskManager(void);
void tmr0Machine(void);
void edgeMachine(void);
void LCD_bcd(unsigned int number);

unsigned int buffer[2];
unsigned int data[2];
char index = 0;

void main(void)
{
    setup();
    startRx();
    while(TRUE)
    {
        switch(mainState)
        {
            case CHECK_SOF:
                while(!RB5);
                while(RB5);
                captureValue = TMR1;
                TMR1 = 0;

                if((captureValue >= 10000) && (captureValue <= 11000))
                {
                    mainState = START_DECODE;
                    TMR1GE = 0;
                    TMR1ON = 0;
                    TMR1 = 0;
                    tmr0State = ENTER_INTERRUPT;
                    TMR0 = 252; //250
                    TMR0IF = 0;
                    TMR0IE = 1;
                }
                break;

            case START_DECODE:
                if(rxFinished == TRUE)
                    mainState = CHECK_ADDRESS;
                break;
        }
    }
}
```

```

    case CHECK_ADDRESS:
        if(buffer[0] == 0b10101010)
        {
            data[0] = buffer[1];
            mainState = TX1_DISPLAY_LCD;
        }
        if(buffer[0] == 0b00000001)
        {
            data[1] = buffer[1];
            mainState = TX2_DISPLAY_LCD;
        }
        break;

    case TX1_DISPLAY_LCD:
        LCD_command(0x01);
        LCD_goto(0);
        LCD_bcd(data[0]);
        LCD_goto(20);
        LCD_bcd(data[1]);
        startRx();
        mainState = CHECK_SOF;
        break;

    case TX2_DISPLAY_LCD:
        LCD_command(0x01);
        LCD_goto(0);
        LCD_bcd(data[0]);
        LCD_goto(20);
        LCD_bcd(data[1]);
        startRx();
        mainState = CHECK_SOF;
        break;

    default:
        break;
}
}
}

void setup(void)
{
    PORTB = 0;
    PORTD = 0;
    TRISE = 0b00100001;
    TRISD = 0b00000000;
    ADON = 0;
    LCD_command(0b00111000); //function set
    LCD_command(0b00000110); //entry mode
    LCD_command(0b00001100); //display control
    LCD_command(0b00000001); //clear display
    rxFinished = TRUE;
    CM1CON0 = 0;
    CM2CON0 = 0;
    CCP1CON = 0;
    ANSELH = 0;
    OPTION_REG = 0b00000101;
    GIE = 1;
    PEIE = 1;
}

void startRx(void)
{
    PORTD = 0;
    OUT = receivedByte;
    rxFinished = FALSE;
    TMROIF = 0;
    TMR0IE = 0;

    T1GSS = 1;
    TMR1 = 0;
    T1GINV = 1;
    TMR1GE = 1;
    TMR1ON = 1;
    ANSELH = 0;
    ANSEL = 0;
}

```

```

void interrupt taskManager(void)
{
    if(rxFinished == FALSE)
    {
        if(TMROIF & TMROIE)
            tmrOMachine();
        if(INTF & INTE)
            edgeMachine();
    }
}

void tmrOMachine(void)
{
    switch(tmrOState)
    {
        case ENTER_INTERRUPT:
            bitCount = 0;
            edgeState = ENTER_EDGE;
            TMROIE = 1;
            INTE = 1;
            TMRO = TMRO_2T;
            tmrOState = FINISH_DECODE;
            INTEDG = 1;
            break;

        case SAMPLE:
            receivedByte = receivedByte<<1;
            if(INPUT == 0)
                receivedByte |= 0x01;
            tmrOState = FINISH_DECODE;
            TMROIE = 1;
            TMRO = TMRO_1T;
            edgeState = CENTER_EDGE;
            INTE = 1;
            if (INPUT)
                INTEDG = 0;
            else
                INTEDG = 1;
            break;

        case FINISH_DECODE:
            rxFinished = TRUE;
            TMROIE = 0;
            INTE = 0;
            break;

        default:
            break;
    }
    OUT = receivedByte;
    TMROIF = 0;
    INTIF = 0;
}

void edgeMachine(void)
{
    switch(edgeState)
    {
        case ENTER_EDGE:
            onePointFiveT = 197;
            tmrOState = SAMPLE;
            TMROIE = 1;
            TMRO = onePointFiveT;
            INTE = 0;
            break;

        case CENTER_EDGE:
            tmrOState = SAMPLE;
            TMROIE = 1;
            TMRO = onePointFiveT;
            INTE = 0;
            if(++bitCount == 16)
            {
                bitCount = 0;
                buffer[index] = receivedByte;
                index++;
                if(index == 2)
                    index = 0;
            }
            break;

        default:
            break;
    }
    TMROIF = 0;
    INTIF = 0;
}

```