
Adaptive Wireless Sensor Network

António Sérgio Sena



An Honours Report submitted for the Degree of:
BEng in Electrical and Electronic Engineering

Heriot Watt University
Electrical, Electronic and Computer Engineering
School of Engineering and Physical Sciences

This copy of the Honours Report has been supplied on condition that anyone who consults it, is understood to recognise that the copyright rests with its author, and that no quotation from the Honours Report and no information derived from it, may be published without the prior written consent of the author or of the University (as may be appropriate).

Adaptive Wireless Sensor Network

António Sérgio Sena

A major problem of agricultural field data collection is the amount of time needed if a large field has to be analyzed for components in soil, along with site surrounding temperature, humidity and others. Field trips are necessary everyday, sometimes several times per day, for special crops, to collect samples or reading on the sites, which may become time consuming, along with an increase in costs.

The objective of this Project was to develop a simple and cost effective Low Power Sensor Network, that could be configured by the end user, with a minimum of training, to help to reduce time when analyzing agricultural fields. The advantage that can be taken of it is the up to date Information Available from sites spread over the field, that Transmit their Data and enables analysis to be made at the Remote Point Computer - Base Station, saving time with unneeded field trips, and giving a real-time crop situation, preventing disasters and, in some cases, with special custom sensors, pest control.

The final result for this Project is a Wireless System that is able of Transmitting Sensor Data across some Distance, having High Energy Savings that enables its Power Supply of delivering Energy for many months.

From Theoretical, System Design and Construction, to Practical Analysis, all information written here enables a complete System to be constructed, and Information is detailed on how to predict the Energy Life of the System, when Powered by Batteries.

Also, the System developed in this Project is able to be adapted to many types of Sensing Situations and Environments, thus giving it a wider choice of uses and opportunities.

Concluding, future enhancements, modifications and other applications are proposed, to enable the forthcoming Designers to continue the work, and also for End-Users to Assemble and Use the design proposed here.

I dedicate this work to my Parents, that helped me so much while i was living in Edinburgh. Also a big thanks to my girlfriend, Alexandra, for the patience she had with me, in some moments of our lives while i was here. Four years of hard work have passed, and many times the loneliness has fallen on me. To all the close friends that gave me their support. To all of you i owe my will of staying and finishing the Degree. Thank you!

Dedico este trabalho aos meus Pais, que tanto me ajudaram, e incentivaram, durante a minha estadia em Edimburgo. Também agradeço à minha namorada, Alexandra, pela paciência que teve comigo, mesmo nas alturas de maior ausência da minha parte. Um abraço a todos os meus amigos do coração, que fizeram força para o meu sucesso e, que sempre me apoiaram. Foram quatro anos de muito trabalho, e por muitas vezes a solidão me assolou. A vocês devo a minha permanência. Obrigado, bem hajam!

Acknowledgements

I would like to thank my Supervisor, Dr. Paul Record, for the strong believes he had on me and my capabilities, specially on those worst moments when the Project had a will of its own.

A big thanks to Ricardo Reis and Tiago Oliveira at ISEL - Instituto Superior de Engenharia de Lisboa (Lisbon Engineering Higher Institute), for their time, will, and help with the machines, when assembling the Radio Transceiver.

Also, a special thanks to my close Friends and Business Partners, Rodrigo Matias and Armando Fonseca, for their friendship and continuous support, and for sending me some Hardware for the Project. Even though, when they had to do the work i could not find the time to do. Thank you both!

Contents

Contents	v
List of figures	vii
1 Introduction	1
2 Project Background	4
2.1 Relevant Technical Literature and Papers	4
2.1.1 What is a Wireless Sensor Network and what uses can it have	4
2.1.2 Power Saving Modes of Operation	5
2.1.3 Error Correction Modes	7
2.1.4 Efficiency of the System and Battery Capacity	8
2.1.5 Communication Protocols	11
3 Theoretical Analysis	13
3.1 Sensor Nodes	14
3.1.1 Simplex Transmission	14
3.1.2 Half-Duplex Transmission	17
3.1.3 Polled Transmission	20
3.2 Base Station	23
4 Practical Analysis	25
4.1 Sensor Nodes	26
4.1.1 Components Used	26
4.1.2 Hardware Behavior	29
4.2 Base Station	32
4.2.1 Components Used	32
4.3 Energy Analysis and Life Expectancy	35
4.3.1 Energy Analysis	35
4.3.2 Life Expectancy	36

4.4	Firmware	41
4.4.1	System Configuration	41
4.4.2	Main Program	41
5	Practical Analysis Discussion	44
5.1	Machine Soldering	44
5.2	PLL Lock	45
5.3	Receiver Sensitivity	45
5.4	PCB Temperature	46
5.5	Practical Analysis Discussion Summary	46
6	Conclusions	48
6.1	Project Changes	48
6.2	Project Limitations	49
6.3	Project Future Modifications	49
A	Base Station Schematic and PCB	51
B	Sensor Node Schematic and PCB	55
C	Radio Transceiver Schematic and PCB	59
D	Bill of Materials	62
E	Source Code	66
E.1	Sensor Node Source Code	66
E.2	Base Station Source Code	70
E.3	RF Transceiver CC1020 Source Code	73
E.4	Temperature/Humidity SHT11 Sensor Source Code	83
F	Battery Discharge Characteristics	86
F.1	Effect of Temperature on Capacity	86
F.2	Effect of Discharge Rate on Capacity	87
	References	88

List of Figures

3.1	Simplex Scheme Fluxogram	14
3.2	Current Consumption for Simplex Scheme	16
3.3	Energy Peaks for Simplex Scheme	16
3.4	Half-Duplex Scheme Fluxogram	17
3.5	Current Consumption for Half-Duplex Scheme	19
3.6	Energy Peaks for Half-Duplex Scheme	19
3.7	Polled Scheme Fluxogram	20
3.8	Current Consumption for Polled Scheme	22
3.9	Energy Peaks for Polled Scheme	22
3.10	Base Station Fluxogram	23
4.1	Current Consumption for the 1 st part of Power Up	29
4.2	Current Consumption for the 2 nd part of Power Up	30
4.3	Current Consumption for the Transmission Cycle	30
4.4	Current Consumption for the Sleep Period	31
4.5	Energy Analysis table and Battery Life Expectancy	39
4.6	Energy Analysis and Battery Life Expectancy graphic	40
A.1	Base Station PC Communications	51
A.2	Base Station CPU, Transceiver Socket and Power Supply	52
A.3	Base Station Top PCB	53
A.4	Base Station Bottom PCB	53
A.5	Base Station Top Silk PCB	54
B.1	Sensor Node CPU	55
B.2	Sensor Node Sensor and Transceiver Socket	56
B.3	Sensor Node Power Supply	56
B.4	Sensor Node Top PCB	57

B.5	Sensor Node Bottom PCB	57
B.6	Sensor Node Top Silk PCB	58
C.1	Radio Transceiver Schematic	59
C.2	Radio Transceiver Top PCB	60
C.3	Radio Transceiver Bottom PCB	60
C.4	Radio Transceiver Top Silk PCB	61
F.1	Variation of Discharge Capacity with Temperature [5]	87
F.2	Effect of Discharge Rate on Capacity [5]	87

Chapter 1

Introduction

A Network of Sensors may be used on any kind of situation, and any kind of Sensing, Control or Data Exchange. Sensors can be used in the medical field, at home or office, in industry, environment analysis, automotive and avionics, and many others. Only the Hardware imposes limitations that make the system hard to port from one Sensing situation to the other.

If a remote large Agricultural field was being monitored by the User, with a Wireless Sensor Network, the failure of a part of that Network, due to lack of Power, would result in no Data Update in real-time. If Data were Updated four times per day, the problem would only probably be detected twelve hours, or even a day, after. If a close monitoring had to be followed, for new experiments, new fertilizers or Humidity conditions to maintain certain levels in a day, one can see how it would be a enormous disaster as the crops would be lost, or with its characteristics dramatically changed, thus wasting one crop or reducing its value to a bare minimum. The financial implications would be enormous, and the future of the Sensor Network would be terminated.

An analogous situation is the case of Life Dependency situations, where the micro Sensor Node can Not fail. The failure of the System, due to lack of Power or bad Design, could cause Health damages, and may even cause permanent irrecoverable damages.

Industry Machine monitor/control is another area that can Not tolerate lack of Data Update. If a factory has its machinery being monitored by the User, with a Wireless Sensor Network, then its the purpose of the Nodes to constantly give Real-Time data of what is happening with the machine. Control Sensor Nodes can be implemented, if in need of feedback to the machine. Along this line, if tanks are to be monitored for inside conditions, they need to update the Network in a reliable manner. If this tanks store food or fast deteriorating supplies, one can see the extreme importance of a carefully designed piece of Hardware System.

The objective of this Project was to develop a simple and cost effective Low Power Sensor Network, that could be configured by the end user, with a minimum of training, to help to reduce time when analyzing agricultural fields.

Due to its characteristics, and proposed work, the project is based on the philosophy that the Sensing Network will be used in Environment monitoring, and that the Hardware will be Adaptive enough, so that other type of Sensing situation can be used.

The One-Hop Transmission scheme was selected due to its non-dependent characteristics. It means that the Sensing Node will not need other Nodes to route the Data along the Network, but it will Transmit directly to the Base Station, thus being able to have higher throughput, and faster Access times, if needed. As also spending much less Energy processing the Routing tables throughout the Network, decoding packets and Correlating Data.

Thus, three schemes were proposed to study for Energy Savings and Battery Longevity :

- Simplex Transmission
- Half-Duplex Transmission
- Polled Half-Duplex Transmission

All of them were studied and analyzed theoretically, but only the Simplex Transmission Scheme was tackled in Practice.

The Theoretical Analysis details what each Transmission Scheme is, explains how the Algorithms works, develops Fluxograms and Current and Energy graphics for a better understanding of the Energy Savings that the Algorithm may have, and gives an Estimation of Energy Expenditure.

The Practical Analysis focus on the Hardware Design for both the Sensor Node and Base Station, lists and details components used, and focus on the Analysis of the Energy Behavior for each Hardware System. Also, it explains and Analyzes the Energy Usage of the Sensor Node, and presents some Life Expectancy calculations, according with the Capacity of the Batteries used. Concluding the Chapter, the Firmware section focus on particularly excerpts

of Source Code, that directly relates to the Energy Savings on how Firmware was written to deal with each Hardware Component of the Sensor Node.

Concluding the investigations, the Practical Analysis Discussion Chapter details the problems that were detected, conditioning the Primary Goal of the Project, and further discussions on how to solve them.

Chapter 2

Project Background

2.1 Relevant Technical Literature and Papers

The following sections provide an insight of the background that already has been researched for the Wireless Sensor Network topic.

Many different topologies for Energy Saving as Networking Routing Algorithms, as Data Gathering and Processing Algorithms, as Processing Power Management, Peripheral Management and Sleep Management Algorithms were studied, and excerpts of literature containing these topologies are added here.

A brief analysis of each section of literature is made, to reinforce this project with its own purpose and some background context.

2.1.1 What is a Wireless Sensor Network and what uses can it have

A wireless sensor network is very different from other types of networks, and the distance between Nodes is shorter. So, the cost of each Node has to be much lower, as also the Power Consumption and the Data Rates. Replacing the Power Source every month or so, for each Node, would be very difficult. [9]

Sensor Networks have to be seen as Networks of scattered pieces of hardware, limited in Energy, Size and Cost. Hence, and since the number of sensors is directly proportional to the monitoring area size, sensors tend to be very small, lightweight and Energy savers, as a whole and as an individual.

"(...) Some of the application areas are health, home, environment and others. In health, sensor nodes can be deployed to monitor patients and assist disabled patients. Some other commercial applications include managing inventory, monitoring product quality, and monitoring disaster areas. (...)" [7]

The Sensor Node has an onboard processor to handle the Data Transmission and Reception, and Peripheral Management, and that same processor can be used to compute Data gathered from sensors. This hardware savings benefit the Energy savings, as the Data Transmitted only has relevant computed Data, probably much smaller than Raw Sensor Data.

If wider areas are to be monitored, Sensor Nodes may as well be used, with bigger sizes, processing power and Transmitting power. The Network philosophy is maintained, for these same Nodes need to perform all the Managements to save Energy.

2.1.2 Power Saving Modes of Operation

Sensor Nodes must support operation in Power Saving mode. Although the easiest way to Save Energy is to turn the Radio Off when is is not required, not always this is the best method. The shorter the Packet sent, the less Energy it is saved, for the operation of turning the Radio On might spend more Energy than if it was left On all the time. [7]

Low Power Consumption is achieved in every aspect of the Design. The Algorithm for sending Data along the Network, maintaining the Hardware and long as possible in Sleep mode, the use of multiple Radio Channels reducing collision, will contribute to reduce Power as a whole. [10]

The Power Saving Mode is the most important one to take care about, for Energy is the most important thing on the Node. Lack of Energy in an Emergency situation, may lead to the failure of the Network, and/or failure of Task accomplishment. Power Saving Algorithm has to tackle the Network Occupancy, to decide if it should leave the Transceiver ON or OFF. These decisions are vital for the survival of the Node and Network, in hostile demanding environments.

One good method of Saving Energy, is to *"(...) aggregate packets along the sensor paths to reduce header overhead."* [2] Correlation between Sensor Nodes are exploited to enhance this Algorithm. Examples of correlated sensors include temperature and humidity sensors in the same near area.

A smart Data Correlation Gathering Algorithm is of vital importance to enhance the Energy Savings, for it can reduce the amount of Data sent to the Receiver, thus reducing the time spent Transmitting and reducing Network bandwidth, by gathering Data from several sensors and send it all in one Transmission.

Compression algorithms having lightweight encoders, can achieve significant savings. The correlation between Sensors determined by using an adaptive prediction algorithm, and need not to know the correlation structure. Only the number of bits that are needed to encode Data are important, meaning that fewer operations are required to encode Data. [2]

Distributed Compression and Adaptive Prediction Algorithms achieve 10 to 65% energy savings for each sensor in typical cases, and *"(...) distributed compression architecture can be combined with other energy saving methods, such as packet/data aggregation, to achieve further gains"* an in [2].

The main challenges in designing a system include: programming a computationally inexpensive encoder that can support multiple compression rates and determining an adaptive correlation-tracking algorithm that can continuously track the amount of correlation that exists between the sensor nodes. [2]

Pursuing even higher Energy savings, encoding already correlated Data, is a way to reduce Transmitted Data along the Network. Aggregating Data from several Nodes, and sending it as single packet throughout the Network or in single Transmission to the Base, saves a lot of Energy, for only one Transmission occurs. A Transmission involves Processing and Transmitting Power, so if Nodes gather Data to a single Transmitter Node, high Energy Savings can be achieved.

2.1.3 Error Correction Modes

When designing Wireless Networks, the Link Reliability is an increased important parameter for Sensor Networks, due to the nature of channels in various application scenarios, as in [7] High Data precision required by Machine monitoring or Mobile tracking, require good Link Reliability and the BER¹ is a good indication of it. The BER can be decreased by increasing the Output Transmitted Power, or by using FEC². Since a Sensor Node has limited power resources, FEC has to be adopted. A given BER can be achieved at lower transmit powers with the use of FEC, though the additional processing power that goes into the FEC Codec³ is drawn from the limited resources of the node. If the associated processing power, using FEC, is greater than the coding gain, thus the whole process is energy-inefficient. Also, it was also said [7] that FEC is generally inefficient if the decoding is performed using a microprocessor, and an onboard dedicated Viterbi decoder is recommended.

Although it may be Energy-Inefficient for short distance Network Nodes, it may be the best method of Transmitting Data when one-hop Transmission is used. One-hop long distance Transmissions require lower BER that can only be accomplished if a powerful FEC is used. Most of the times, raising the Power Output is not feasible, resulting in Node Failure. This way, the onboard processor could as well perform the FEC encoding, for the FEC decoding would only be done by the Base Receiver Station.

"(...) Error control codes are known to have higher power efficiency even with all the redundant bits added. This is of course at the cost of bandwidth. But bandwidth is not a serious concern for sensor networks. The tradeoff is really between the complexity of the encoder and decoder (computational power consumption) and transmission power efficiency. Because the distance between neighboring nodes is short, computational power can be comparable to transmission power. (...)" [9]

Again, there is a tradeoff between multi-hop and one-hop Networks. In the case of multi-hop,

¹Bit Error Rate

²Forward Error Correction

³Encoder and Decoder

the onboard processor spend more Energy Decoding the FEC Data, than acquiring Data, processing it to Transmission, and Transmitting it. While in one-hop Network, the Node spends that same amount of Energy, doing FEC Encoding, Data processing and Transmission for several times.

"(...) Conventional approaches like Forward Error Correction (FEC) or Automatic Repeat Request (ARQ) all require acknowledgement. Acknowledgement is needed in ARQ to let the sender know whether it needs to retransmit the packet. FEC eliminates the need for retransmission, but acknowledgement is still needed to let the sender know if the packet has been received at all by the destination, no matter it is correct or not. Acknowledgement is very expensive in terms of power (...)" [10]

There is quite a bit of disagreement on this statement, regarding to the ones before. It is known that if a FEC scheme is powerful enough, the Sender Node does not need Acknowledgement from the Receiving Base Station, to let it know that the packet was received. If a good FEC scheme is used, the BER can be made significantly low, even while having low SNR⁴. Thus, Acknowledgement would not be necessary. If the Receiving Base Station has a DSP⁵ unit, even lower SNR and lower BER can be accomplished.

Acknowledgment Expensive Power means that, with the need of the Transmitter to receive a confirmation of arrival of the Packet sent, the Energy Spent is much Higher than when using a simple One-way protocol, or when using FEC.

2.1.4 Efficiency of the System and Battery Capacity

A brief excerpt by [8], describes it in a good way:

"(...) A main challenge in the design of an energy-efficient wireless network is that, sending a bit of information through free space directly from node A to node B, incurs an energy cost E_t , which is a strong function of the distance d between the nodes. More precisely,

⁴Signal to Noise Ratio

⁵Digital Signal Processing

$E_t = \beta \times d^\gamma$, with $\gamma > 1$ as the path-loss exponent (a factor that depends on the RF environment, and is generally between 2 and 4 for indoor environments) and β is a proportionality constant. Given this greater than linear relationship between energy and distance, using several short intermediate hops to send a bit is more energy-efficient than using one longer hop. For example, assuming $\gamma = 4$, which is a common case in indoor environments, and femtojoules/meter⁴, one hop over 50 meters requires 1.25 nanojoules per bit, whereas five hops of 10 meters require only 5×2 picojoules per bit. The multihop approach in this example reduces transmission energy by a factor of 125 (...).”

Also by [4]:

”(...) Now, how much energy is needed for the processing? The energy consumed is the power over the time period, $E = \int_0^T P dt$. For the DC processing components $P = IV$ where the current I is a function of f . We assume that voltage and frequency vary independently, e.g. we remain away from fundamental CMOS limits. We hold the voltage V constant, so $E = \int_0^T I(f) dt$. (...).”

This demonstrates that the dominance of the Standby or Sleep currents, are the most important on the Node System. E.g. Transmit time of 1 second, Sleep time of 599 seconds. Transmit Current of 24mA, Sleep current of 500 μ A. The dominant Current is of the Sleep period, it has more than 99.7% of the occupancy of Total period. Hence, the Total Average Current for the 10 minute period is around 520 μ A.

The current Protocols problem is that they find the Lowest Energy Route on the Network, and use it always for every Communication, though it is not the best method to increase the Network Lifetime. Using also the same Low Energy path, leads to Network partition, as said by [17].

If the Receiving Base Station is positioned always on the same place, then the Network Routes will always be the same, from every Node. A random distribution of the Network Routes should be programmed, so the Energy spent is Linear throughout the Network, and not just on some Routes.

[8] said that, "(...) *Although batteries can store harvested energy that cant be used immediately, a continuous source of energy is desirable. Solar cells can contribute up to 15mW per square centimeter during direct sunlight hours and up to 0.15mW on cloudy days. (...)*".

This may be a somewhat pessimistic statement, regarding Solar Energy. For applications like this, at present days, it is still cheaper to have the Nodes powered by a small Solar Panel, that gives them some extra energy so sustain a much Higher Lifetime.

[11] also said that for a given battery, the amount of energy that can be used by the circuit is a function of the current discharge rate of that circuit. The battery life does not have a simple linear relationship with the power consumption of the circuit. A 2× increase in circuit power consumption may cause a 3× reduction in the battery lifetime, compared with the 2× reduction in the ideal case.

One of the major disadvantages of using non-linear Energy Sources. Perhaps using a system like Battery/Solar Panel, would help to increase lifetime.

An important analysis made by [11] says that "*One important characteristics of the battery is that some amount of energy will be wasted when the battery is delivering the energy required by the circuit. In analytical form, given a fixed battery output voltage, if the circuit current requirement for the battery is I , the actual current that is taken out of the battery is: $I^{act} = \frac{I}{\mu}$, $0 \leq \mu \leq 1$ where μ is called the battery efficiency (or utilization) factor. I^{act} is always larger than or equal to I . Defining CAP_0 as the amount of energy that is stored in a new (or fully charged) battery and CAP_{act} as the actual energy that can be used by the circuit, later equation is equivalent to: $CAP^{act} = CAP_0 \times \mu$, $0 \leq \mu \leq 1$. The efficiency factor μ is a function of discharge current I : $\mu = f(I)$ where f is a monotonic-decreasing function. Only the low frequency part of the current is relevant to changing the battery efficiency (...)*".

A good proof that the higher the discharge Currents, the higher the losses for that battery. A careful designed Energy Usage Algorithm, will enhance Battery life. The higher the period

between Transmissions, will for certain extend Battery life.

2.1.5 Communication Protocols

As stated by [19], when using a Direct Communication Protocol, means that each Sensor Node sends its data directly to the Base Station. If the Base Station is far away from the Nodes, direct communication will require a large amount of transmit power from each node. This method drains the Battery quickly, and reduces Network System lifetime. However, only the Base Station receives Data, so if the Base Station is in close proximity of the Nodes, this may be a possibly optimal method of communication. The other considerable approach is the Minimum-Energy routing protocol, where the intermediate Nodes are chosen causing the Transmit Energy to be minimized. *"(...) However, for this Minimum-Transmission-Energy (MTE) routing protocol, rather than just one (High-Energy) transmit of the data, each data message must go through N (low energy) transmits and N receives. (...) the total energy expended in the system might actually be greater using MTE routing than direct transmission to the base station. (...) When transmission energy is on the same order as receive energy, which occurs when transmission distance is short and/or the radio electronics energy is high, direct transmission is more energy-efficient on a global scale than MTE routing. Thus the most energy-efficient protocol to use depends on the network topology and radio parameters of the system.*

[19] also says that in MTE routing, the Nodes closest to the Base Station will be used to route a large number of data messages to it. Thus, these nodes will die out their Batteries quickly, and the Energy to get the remaining Data to the Base Station will increase and more Nodes will die. This creates a cascading effect that will shorten Network System life. Also, as Nodes close to the base station die, that area is no longer being monitored.

The usefulness of an algorithm that manages both Direct and MTE on the same sensor can be seen. There are many situations where both systems can be used together, without considerable loss of Energy on some/few Nodes. Again, depending on the Network Topology, a carefully designed Algorithm and the use, or not, of a renewable Energy source, the life

expectancy of the Network as a whole, can be very high.

Chapter 3

Theoretical Analysis

Each of the next sections will focus on the three schemes proposed to be studied, and will explore them, according to the Hardware used for each Sensor Node. The last section will explain what the Base Station is, and how it works.

To be able to prove and enhance this Theoretical Analysis, Electronic Designs were produced for a Sensor Node, and for a Base Station. Hardware will be explained with detail, in Chapter 4.

A Fluxogram and Current and Energy Analysis through Time graphics were produced, to better explain the behavior of each scheme on the Energy Consumption area. Comparisons were made, to show the pros and cons when in different RF channel conditions.

3.1 Sensor Nodes

3.1.1 Simplex Transmission

Simplex Transmissions are those where the Sensor Node only Transmits Data. The Sensor Node work is to collect Data, and send it at predetermined intervals (internal timer). The Sensor Receiver Hardware only exists to check for Carrier on the channel, or Busy Channel. Thus, Base Station can not control the Nodes. It has continuous work and can not be ShutDown.

Figure 3.1 gives a better explanation on how the Simplex Transmission Scheme works, as it sits in a continuous loop, only interrupted by the internal timer:

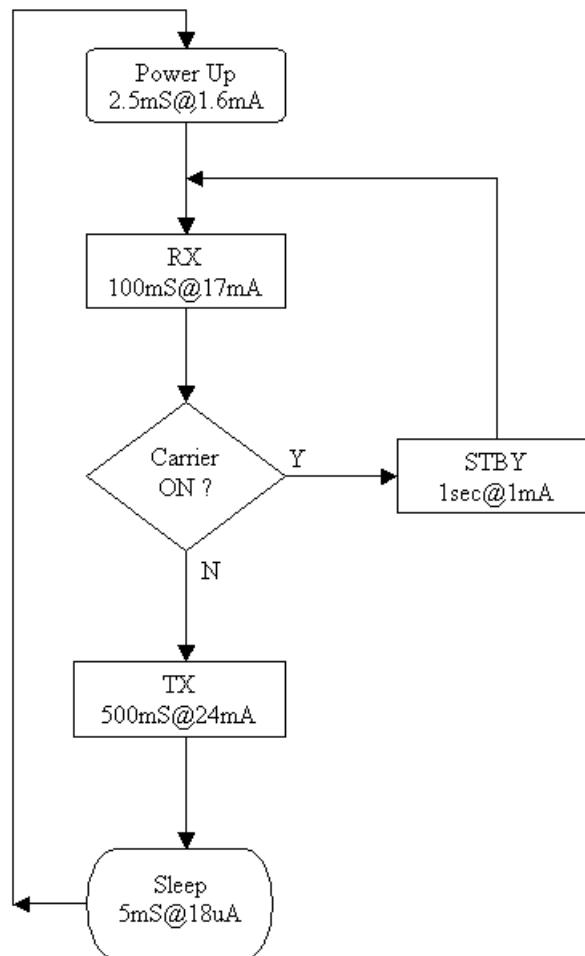


Figure 3.1: Simplex Scheme Fluxogram

Fluxogram, Current and Energy analysis graphics were drawn, according to components used on the Sensor Node Hardware. The Data for these graphics were taken from the manufacturers components datasheets.

These graphics try to show the Current Consumption behavior, and Energy Peaks, against Time, from the moment the Sensor Node wakes up, until Node enters Sleep mode. The Time shown is a sum of the Time needed to perform all the Tasks programmed, as per Figure 3.2. Comparison is made between a Channel with Carrier present before Transmitting, and between a Channel with NO Carrier present before Transmitting.

Trend lines are also shown to evaluate the Average Tendency for both situations. It can be seen that the existence of a Busy Channel, makes the Average Current to step up, on one Work Period of the Sensor Node (600 seconds or 10 minutes. This is due to the Standby period and Channel Busy status check. Both this contribute to a higher Energy loss.

When sending a 60 byte Packet, the calculated average Energy is:

With Carrier : 14.21 pJ per bit

With NO Carrier : 12.8 pJ per bit

Figure 3.2 shows the Current Consumption graphic of the Sensor Node, and Figure 3.3 shows the Energy Peaks graphic, both using the Simplex Transmission scheme:

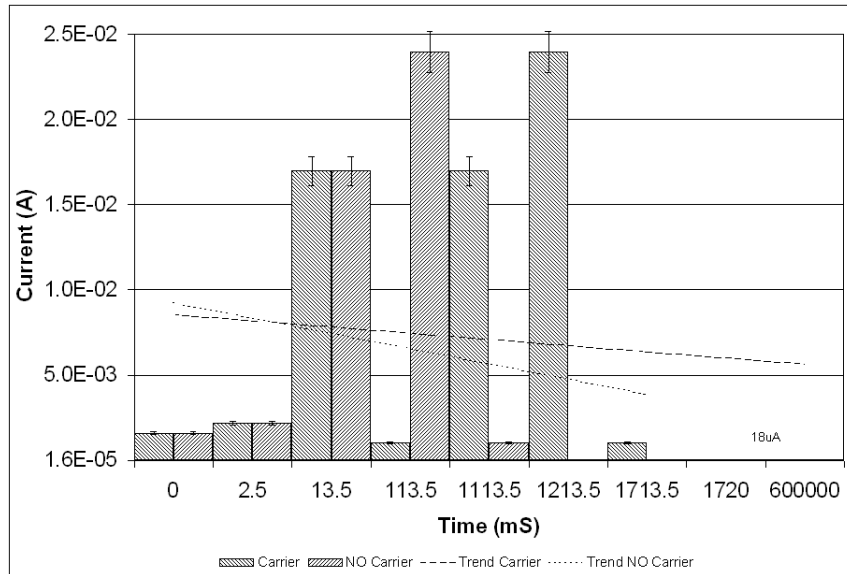


Figure 3.2: Current Consumption for Simplex Scheme

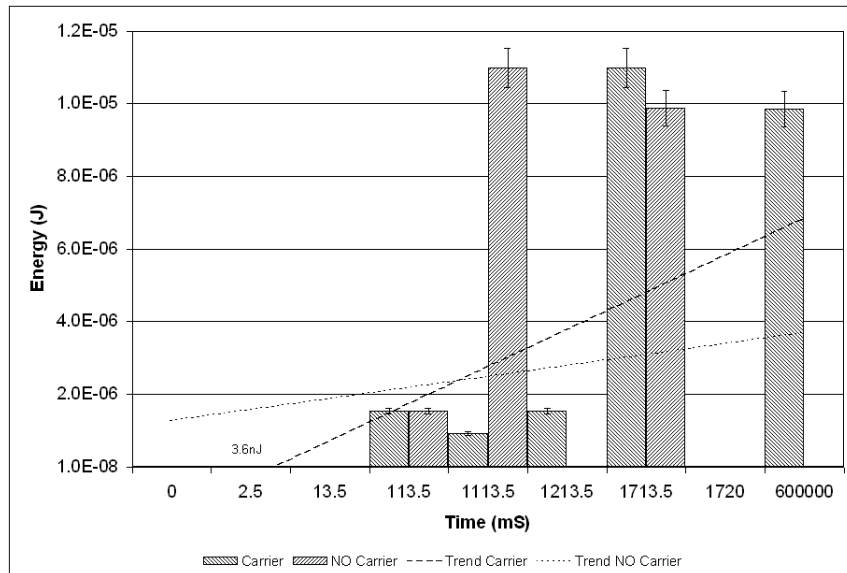


Figure 3.3: Energy Peaks for Simplex Scheme

3.1.2 Half-Duplex Transmission

Half-Duplex Transmissions are those where the Sensor Node both Transmit Data at predetermined intervals, and Receive Commands, from Base Station, to execute. These Commands can be of any kind, and preprogrammed by the designer. increase/decrease Transmit Power, increase/decrease Transmit delay or enter ShutDown mode, are among them.

Figure 3.4 gives a better explanation on how the Half-Duplex Transmission Scheme works:

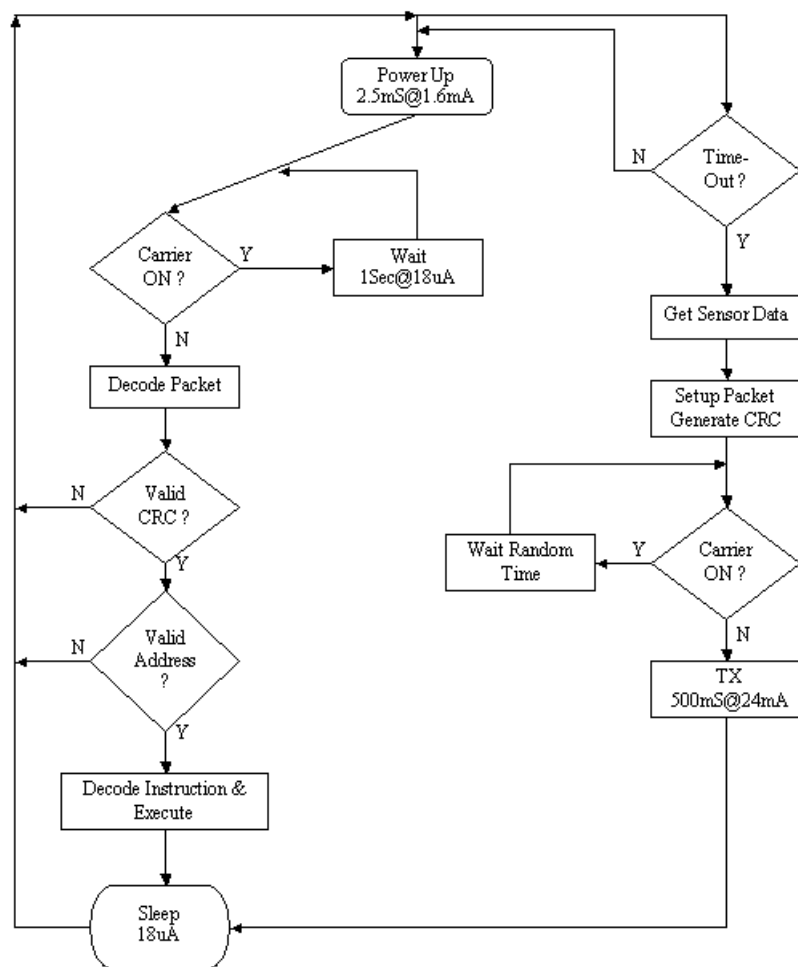


Figure 3.4: Half-Duplex Scheme Fluxogram

Fluxogram, Current and Energy analysis graphics were drawn, according to components used on the Sensor Node Hardware. The Data for these graphics were taken from the manufacturers components datasheets.

These graphics try to show the Current Consumption behavior, and Energy Peaks, against Time, from the moment the Sensor Node wakes up, until Node enters Sleep mode. The Time shown is a sum of the Time needed to perform all the Tasks programmed, as per Figure 3.4. They show the worst case analysis, where there is a Timer Time-out, for new Data to be gathered and transmitted, followed by a Channel with Carrier present before Transmitting.

Trend lines are also shown to evaluate the Average Tendency for both graphics. It can be seen that although there is a high Energy consumption when in Sleep mode, towards the end of the Work Period, it is this average Sleep Current that imposes the Overall Energy consumption of the Sensor Node. Work period is of 600 seconds or 10 minutes.

When sending a 60 byte Packet, the calculated average Energy is:

With Carrier : 15.5 pJ per bit

Figure 3.5 shows the Current Consumption graphic of the Sensor Node, and Figure 3.6 shows the Energy Peaks graphic, both using the Half-Duplex Transmission scheme:

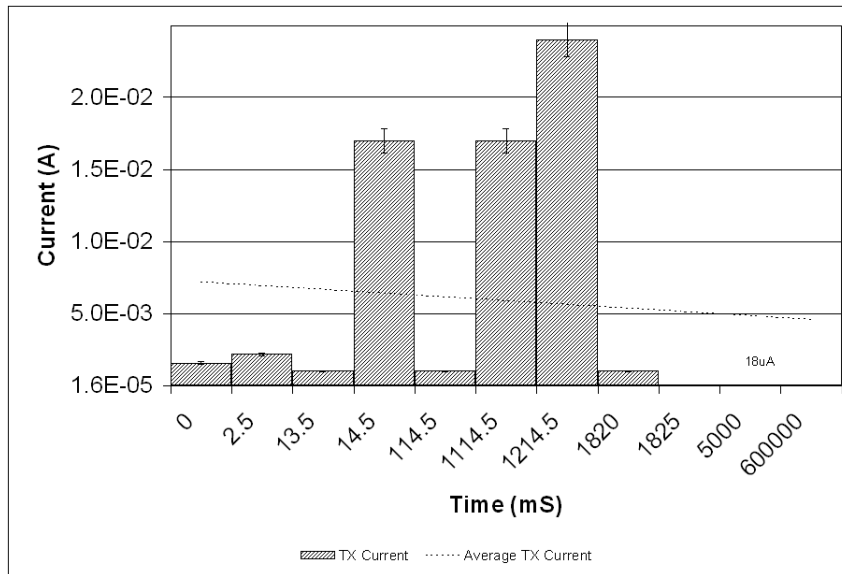


Figure 3.5: Current Consumption for Half-Duplex Scheme

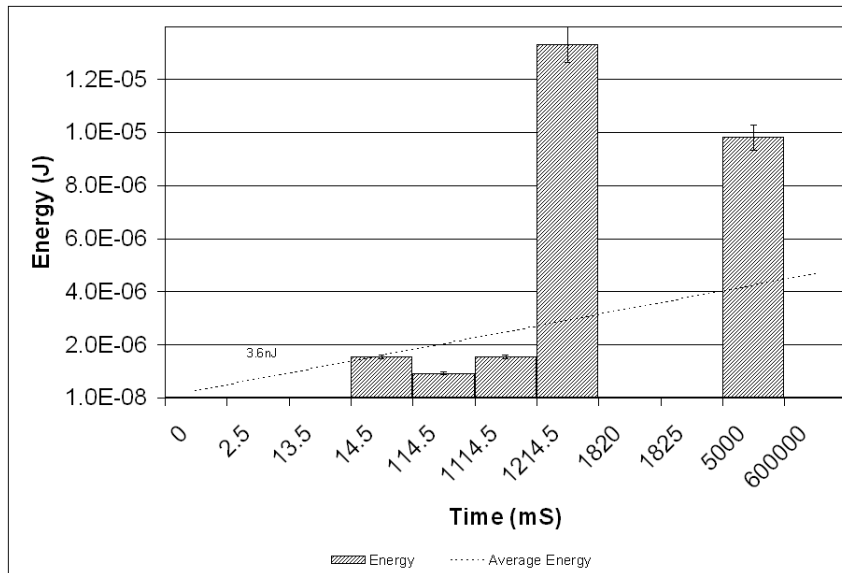


Figure 3.6: Energy Peaks for Half-Duplex Scheme

3.1.3 Polled Transmission

Polled Transmissions are those where the Sensor Node only Transmits when commanded by the Base Station. Base Station sends addressed commands, and if the Address matches the ID of any Sensor Node, that Node will respond to the command sent.

Figure 3.7 gives a better explanation on how the Polled Transmission Scheme works:

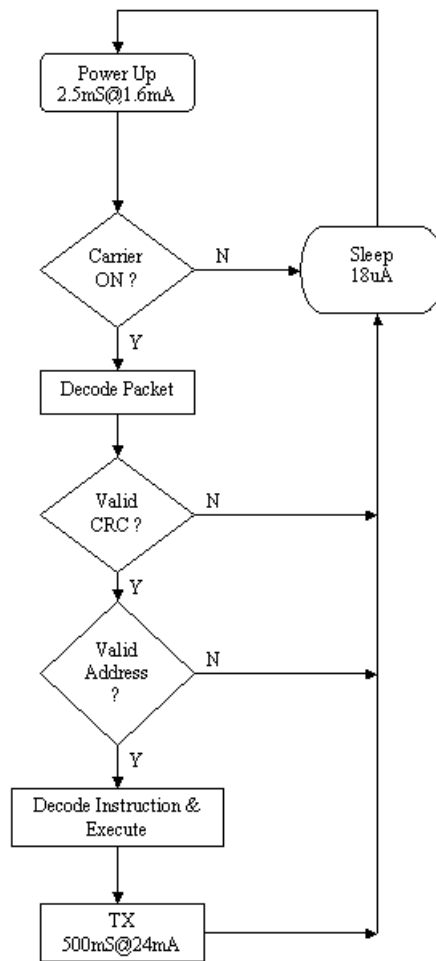


Figure 3.7: Polled Scheme Fluxogram

Fluxogram, Current and Energy analysis graphics were drawn, according to components used on the Sensor Node Hardware. The Data for these graphics were taken from the manufacturers components datasheets.

These graphics try to show the Current Consumption behavior, and Energy Peaks, against Time, from the moment the Sensor Node wakes up, until Node enters Sleep mode. The Time shown is a sum of the Time needed to perform all the Tasks programmed, as per Figure 3.7. Comparison is made between Match ID and NO Match ID situations.

Trend lines are also shown to evaluate the Average Tendency for both graphics. It can be seen that the Average Current spent is much lower when there is NO Match ID, for the Node enters in Sleep mode right after it rejects the NO Match ID packet. It can also be seen that the Energy consumption increases when a Match ID occurs. Therefore, processing power is spent on Decoding the Packet and Processing Data to Transmit. Work period is of 600 seconds or 10 minutes.

When sending a 60 byte Packet, the calculated average Energy is:

With Match ID : 171.5 pJ per bit

With NO Match ID : 78.3 pJ per bit

Figure 3.8 shows the Current Consumption graphic of the Sensor Node, and Figure 3.9 shows the Energy Peaks graphic, both using the Polled Transmission scheme:

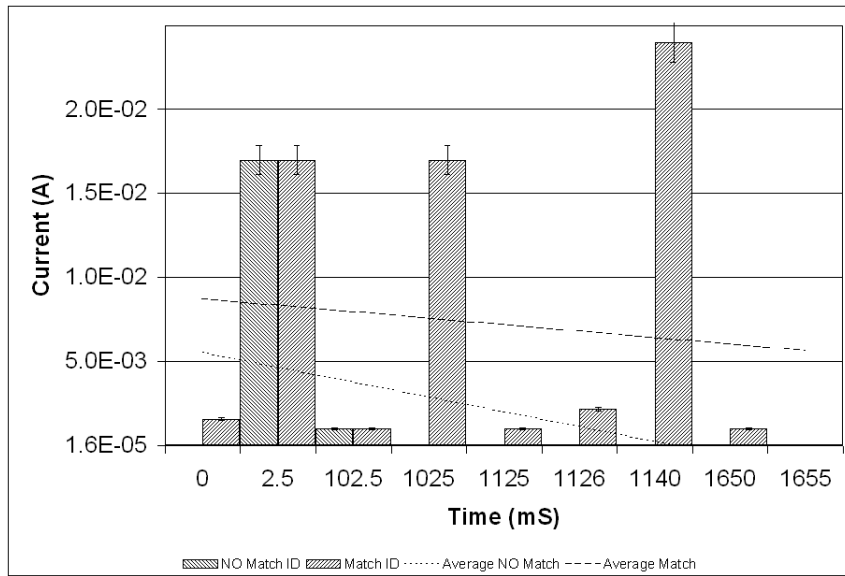


Figure 3.8: Current Consumption for Polled Scheme

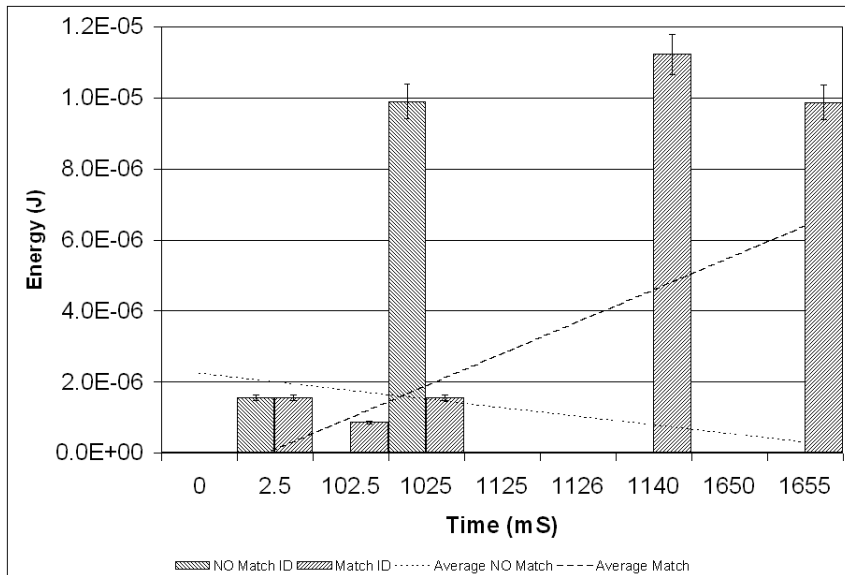


Figure 3.9: Energy Peaks for Polled Scheme

3.2 Base Station

The Base Station is the heart of all Data between the PC (End User) and the Network. It acts as a relay. It constantly monitors the RF Channel for any Carriers, tries to Decode them and, if one of them is a valid Packet from a Sensor Node, it Processes the Data and sends it to the PC (End User).

Figure 3.10 gives a better explanation on how the Polled Transmission Scheme works:

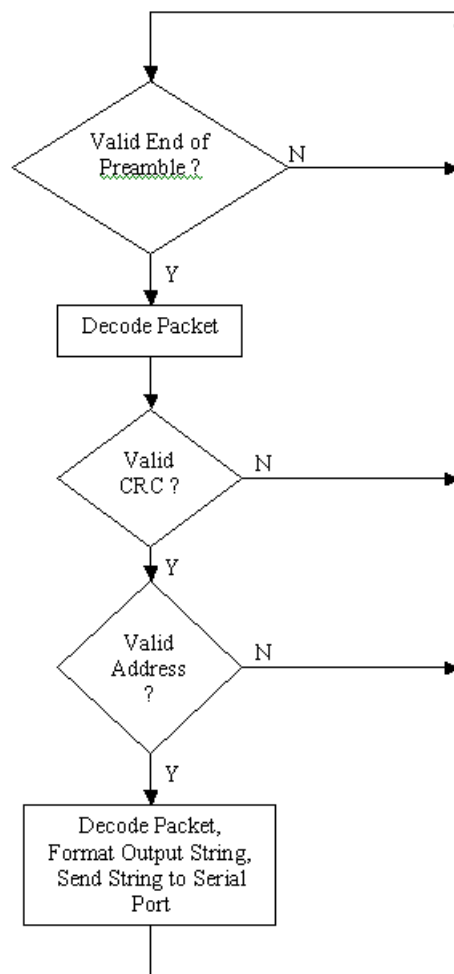


Figure 3.10: Base Station Fluxogram

No Energy Analysis has been made to the Base Station, for it does not have Power Constraints. Hence, it can be in continuous Reception Mode, to have the RF Channel always being monitored.

The Algorithm implemented for the Base Station, only allows for a Sensor Node Simplex Scheme (Fig. 3.1) to be used.

Chapter 4

Practical Analysis

The Practical Analysis chapter is where all of the Practical work will be analyzed. This analysis spans from the choices made when choosing Components to build the Sensor Node and Base Station, to the final Energy and Life Expectancy Analysis.

This chapter will explain in detail:

- Components used on both Sensor Node and Base Station
- Hardware Current Consumption Behavior
- Energy Analysis and Life Expectancy
- Firmware for each Component used

4.1 Sensor Nodes

The Sensor Node is the most fragile piece of Hardware on the Network System. The fulfilling of the job proposed, on numerous situations, depends on the good Working Condition, Reliability and Life Period of the Node. If the Batteries reach to an end on a critical phase, or event, where the Node is needed the most, the whole purpose of the Wireless Sensor Network will be a failure, and would compromise the future of the system, regarding to its extendability to further areas and markets.

4.1.1 Components Used

In order to keep the Power Consumption as Low as possible, the Components were carefully chosen to work on a supply voltage of 3.3V DC. This is the lowest allowed, for some Components were impossible to find for lower Voltages.

Refer to Appendix B for the Sensor Node Schematic and PCB.

The list of Components used for the Sensor Node:

- Microchip PIC 18LF252, Microcontroller [15]
- Chipcon CC1020, RF Transceiver Chip [1]
- Sensirion SHT11, Temperature/Humidity sensor [18]
- Maxim MAX1724, DC/DC Switch Mode Power Supply Chip [12]
- 2xAA 2100mA NiMH Batteries [5]
- ISM UHF whip/rubber antenna

4.1.1.1 PIC 18LF252 Microcontroller

This microcontroller was chosen for its special characteristics of Low Power Modes, among a multitude of useful Peripherals that would later allow the Sensor Node performance to be escalated:

- 32kBytes of Flash Memory & 1k5Bytes of RAM
- 8x8 Hardware Multiplier, useful for Mathematical Data Processing
- Secondary Oscillator Clock Options, to run the CPU at different Speeds
- Several Serial Communication Methods, enables many different peripherals to be used
- 10-bit A/D Converter, allowing analysis of "Real-World" signals on Chip
- Extremely Low Sleep Supply Current of $\leq 1\mu\text{A}$
- Low Operating Current of $\leq 1\text{mA}$

4.1.1.2 CC1020, RF Transceiver Chip

Small size and Easy of Operation were the key points when choosing this RF Transceiver Chip.

- Single Chip UHF RF Transceiver
- High Sensitivity
- Programable Output Power
- Low Supply Voltage
- Digital RSSI¹ and Carrier Sense Indicator
- $\leq 1\mu\text{A}$ in Power Down Mode

4.1.1.3 SHT11, Temperature/Humidity sensor

This Sensor was chosen for it has a combined Temperature/Humidity sensing capability on the same package:

- A/D Converter capable of 14-bit for Temperature, and 12-bit for Humidity

¹Receive Signal Strength Indicator

- Calibrated Digital output, 2-Wire Serial output
- Extremely Low Supply Current of $\leq 28\mu\text{A}$
- Extremely Low Sleep Supply Current of $\leq 0.3\mu\text{A}$

4.1.1.4 MAX1724 DC/DC Switch Mode Power Supply Chip (SMPS)

One of the most important Components in the Sensor Node System, is the SMPS Chip. Besides its thin small package, the easiness of use was a must:

- $\leq 90\%$ Efficiency
- $\leq 1.5\mu\text{A}$ Quiescent Supply Current
- $\leq 0.1\mu\text{A}$ Logic Controlled ShutDown
- 0.8VDC to 5.5VDC input Voltage Range

With an Input Voltage of around 2.4VDC², the SMPS has around $\leq 80\%$ Efficiency. The Quiescent Current is a very important parameter, for it contributes to the Sleep/Standby Current spent by the System.

4.1.1.5 2100mA NiMH Batteries

This High Capacity NiMH Batteries were chosen, for the good performance of this Battery Technology in Slow Discharge conditions, while allowing continuous Top-Chargings with Solar Panel used and having much less Memory-Effect than a regular NiCd Battery Technology.

4.1.1.6 ISM UHF antenna

This is a $\frac{1}{4}\lambda$ High Gain rubber Antenna, capable of the ISM bands. It will be used to Transmit&Receive on the UHF 433.92MHz Channel.

²With 2xAA NiMH Batteries

4.1.2 Hardware Behavior

After assembling the Hardware PCB³, loading the Sensor Node Firmware, the Current Consumption Behavior was analyzed and measured with a Digital Oscilloscope. A 10 Ω Resistor was placed in series with the Low Side of the circuit: from the Circuit Ground, to the Battery Ground. The Digital Oscilloscope was connected in Parallel to this Resistor.

The following were Measured and Captured:

- 1st part of Power Up, Fig. 4.1
- 2nd part of Power Up, Fig. 4.2
- Complete Transmission Cycle, Fig. 4.3
- Portion of Sleep/Standby Period, Fig. 4.4

The graphics bellow show the Captured Measurements made by the Digital Oscilloscope:

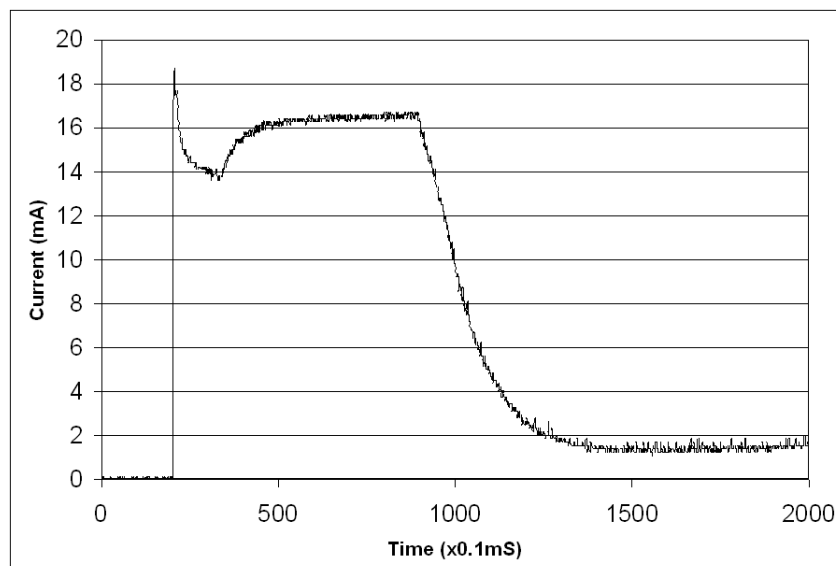


Figure 4.1: Current Consumption for the 1st part of Power Up

³Printed Circuit Board

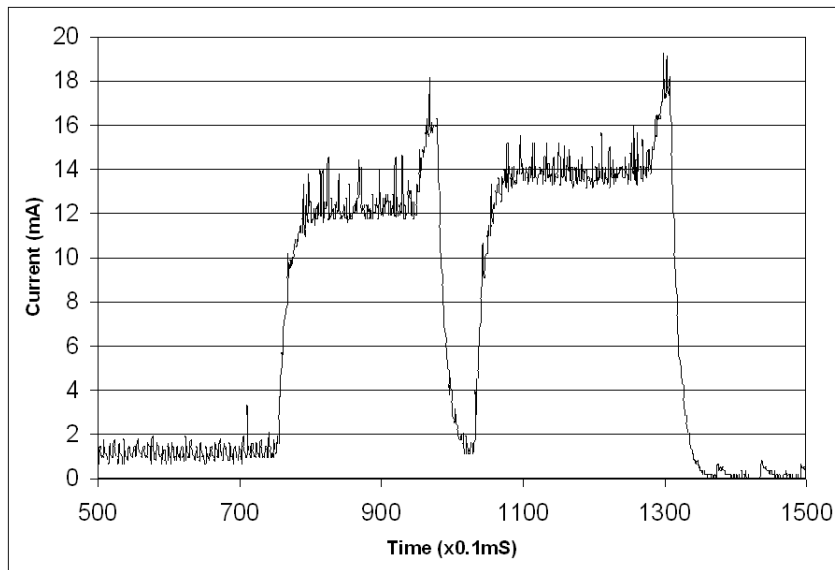


Figure 4.2: Current Consumption for the 2nd part of Power Up

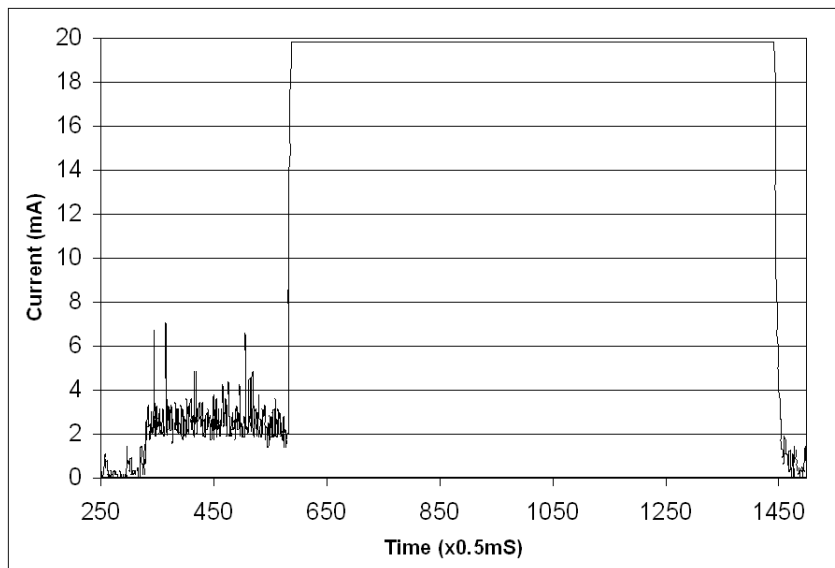


Figure 4.3: Current Consumption for the Transmission Cycle

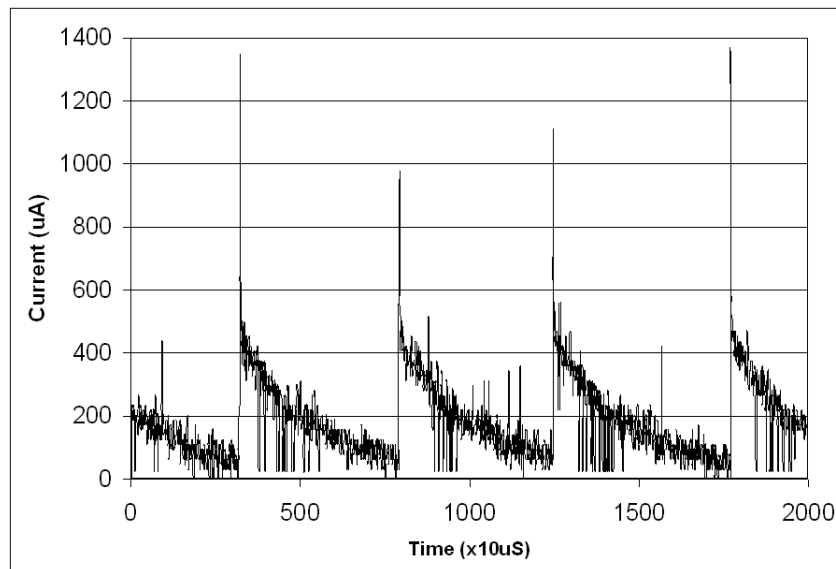


Figure 4.4: Current Consumption for the Sleep Period

Fig. 4.1 corresponds to the initial Power Up consumptions of SMPS, CPU and unconfigured RF Transceiver Chip. Stops in Standby Current around 1.4mA.

Fig. 4.2 details the start Standby Current, TX PLL Calibration and RX PLL Calibration, stopping in Sleep Current. The end peaks on each PLL Calibration, are most certainly from the final stage of Calibration, though no reference is made to this on the CC1020 datasheet [1].

Fig. 4.3 shows the start Sleep Current, the CPU wakes up, gets Data from Sensor, and then a Long High Peak of Transmitting Current, stopping in Sleep Current.

It can be seen, in Fig. 4.4, that the major Power Consumption comes from the SMPS Chip. The Ripple Current is product of the PFM⁴ Control Scheme and the Inductor Peak Current of the SMPS Chip. At this Low Load Currents, the SMPS Chip Load-Transient Response is a fairly clean Ripple Waveform. [12]

The Noise on the Waveform shown in Fig. 4.4, is from the SMPS Chip's PFM Control Scheme and the Noise coupled by the Oscilloscope. Peaks at the beginning are well seen and translate the Current Peak on the Inductor, followed by its Discharge to the Circuit. [12]

⁴Pulse Frequency Modulation

4.2 Base Station

Along with the Sensor Node, the Base Station needs to be a Robust, and Reliable, Link between the Sensor Network and the PC (End User). Thus, it is not allowed to fail in any circumstances. Its Firmware must be prepared to deal with self-crashes, in order to Resume normal operation as fast as it can, so it wont loose any Incoming Packets from any Sensor Node Transmitting at that time. It must forward the Incoming Data to the PC, at a high rate, and must not loose any Incoming Packets while doing this.

With the condition of Robustness in mind, the Base Station Hardware was kept to a minimum, in order not to decrease the MTBF⁵ of the System.

Refer to Appendix A for the Base Station Schematic and PCB.

4.2.1 Components Used

The list of Components used for the Base Station:

- Microchip PIC 18F252 [15]
- Chipcon CC1020, RF Transceiver Chip [1]
- MAX3232 RS/232 interface & DLP/USB232 USB DIP Module Interface [13] [6]
- ISM UHF antenna

4.2.1.1 PIC 18F252 Microcontroller

This microcontroller was chosen for its large Flash Memory, and a multitude of useful Peripherals that would later allow the Base Station performance to be escalated:

- 32kBytes of Flash Memory & 1k5Bytes of RAM
- 8x8 Hardware Multiplier, useful for Mathematical Data Processing

⁵Mean Time Between Failure

- Several Serial Communication Methods, enabling many different peripherals to be used

4.2.1.2 CC1020, RF Transceiver Chip

Small size and Easy of Operation were the key points when choosing this RF Transceiver Chip.

- Single Chip UHF RF Transceiver
- High Sensitivity
- Programable Output Power
- Digital RSSI and Carrier Sense Indicator

4.2.1.3 MAX3232, RS/232 interface

Easy of Operation and RS/232 \Leftrightarrow 3.3V CMOS Buffering, were the key points when choosing this Buffer Transceiver Chip.

4.2.1.4 DLP/USB232 USB DIP Module Interface

Easy of Operation and USART \Leftrightarrow USB , were the key points when choosing this Buffer Transceiver Chip.

- Single Chip USB Asynchronous Serial Data Transfer
- Integrated level converter on UART and control signals for interfacing to 5V and 3.3V logic
- Integrated 3.3v regulator for USB IO
- USB 1.1 and USB 2.0 compatible
- Virtual COM Port (VCP) Drivers

4.2.1.5 ISM UHF antenna

This Antenna must be capable of the ISM bands. It will be used to Transmit&Receive on the UHF 433.92MHz Channel.

4.3 Energy Analysis and Life Expectancy

After analyzing the Current Consumption Behavior for the Sensor Node, this section will Explain and Analyze the Energy Usage of it, and present some Life Expectancy calculations, according with the Capacity of the Batteries used.

4.3.1 Energy Analysis

The Energy Efficiency of the Overall System is focused where most of the Current is spent, for different Sensor Node Work Periods: The SMPS Chip.

As it was seen in the Theoretical Analysis Chapter, the Sleep Current is a dominant factor that conditionalizes the Overall Performance of the Sensor Node, hence the Battery Life⁶. The Table of Fig. 4.5 shows the Energy Analysis after all the Measurements were done.

With special attention to the calculations for 10 min and 60 min periods, it can be seen that the Average Current difference between them both is of $\approx 17\mu\text{A}$, and ≈ 20 more days if using 60 min periods. It's an increase of $\approx 7.5\%$. This can be particularly useful, for there may be special Events that need increased monitoring, and the Sensor Node Algorithm can change the Transmitting Periods, according with the needs of the User and Event, without significantly undercharging the Batteries.

Also, there is not much Energy Saving if the Periods are bigger than 60min, and even 30min.

The Table of Fig. 4.5 was produced with the following Specifications and Measured Values:

- Duty Cycle Current: 24mA
- Average Sleep Current given by Fig. 4.4: $200\mu\text{A}$
- Supply Voltage: 3.3V
- Battery Voltage: 2.55V ⁷ (2xAA 1.2V NiMH)

⁶Appendix F gives an overview of Battery Discharge Characteristics

⁷For fully charged Batteries

- Battery Capacity: 2100mA
- SMPS Efficiency: 80% [12]
- Battery Efficiency: 80%⁸
- Total Bits Sent: 488

The use of a SMPS Chip has pros and cons, and hence some drawbacks. It is known [12] that the SMPS Chip maintains the 3.3V Output, regardless the Input it has⁹. And though it guarantees an Output of 40mA at 1.0V Input, the Current Drawn from the Batteries will be much higher, to compensate for the PFM Control Scheme and Synchronous Rectification. Thus, the lower the Input voltage, the Higher the Current needed to maintain the Output voltage.

So, the Current Drawn from the Batteries is Inversely Proportional to Batteries' Voltage, and the Battery Discharge Characteristic changes due to the increased Discharge Rate. Hence, this constantly changing Behavior has an Impact Effect on the Batteries' Capacity. [5] [3] and Appendix F

As the Battery Discharge Characteristic is hard to accommodate on the calculations for Battery Life Expectancy, the Battery's Efficiency was extrapolated to be 80%, trying to make this changing Behavior as close as possible to Real Results.

4.3.2 Life Expectancy

As said before in Section 4.3.1, it is very difficult to predict a correct Discharge Characteristic for the Batteries used on the Sensor Node. That's why the use of an Empirical value, trying to match the Results as close as possible to the Real Characteristic.

The Battery Life calculations, were based on key aspects that dictate the Longevity of the System, taken from Table of Fig. 4.6:

⁸Value extrapolated from Capacity Loss due to Temperature Spans and Self-Discharge

⁹Input has to be between range

- Average Current [AvgC]
- Battery Capacity [BatC]
- Supply efficiency [η_1]
- Battery Efficiency [η_2]

And also:

- Sleep Time [ST]
- Sleep Current [SC]
- Period Time [PT]
- Duty Cycle Time [DCT]
- Duty Cycle Current [DCC]

The Average Current was calculated using the formula:

$$\text{Average Current} = \frac{\text{ST [s]} \times \text{SC [A]} \times \text{DCT [s]} \times \text{DCC [A]}}{\text{PT [s]}} [\mu\text{A}]$$

Putting some numbers in:

$$\text{Average Current} = \frac{599.5 \times 200\mu \times 0.5 \times 24m}{600} = 220\mu\text{A}$$

When writing the Mathematical formula for calculating the Battery Life, the number of hours that add to a full Day were used, hence 24. This total number of Hours, will cancel with the Total Hours Battery Capacity calculated for the Average Current, giving the total Days for Battery Life.

$$\text{Battery Life} = \frac{\frac{\text{BatC [mAh]} \times \eta_1 \times \eta_2}{\text{AvgC [A]}}}{\text{Hours in a Day}} [\text{Days}]$$

Putting some numbers in:

$$\text{Battery Life} = \frac{\frac{2100 \times 0.8 \times 0.8}{220\mu}}{24} = 254.7 \text{ [Days]}$$

Another example, to find the Battery Life in Months, using the number of Hours that add to a full Month were used, hence 730.5¹⁰. This total number of Hours, will cancel with the Total Hours Battery Capacity calculated for the Average Current, giving the total Months for Battery Life.

$$\text{Battery Life} = \frac{\frac{\text{BatC [mAh]} \times \eta_1 \times \eta_2}{\text{AvgC [A]}}}{\text{Hours in a Month}} \text{ [Months]}$$

Putting some numbers in:

$$\text{Battery Life} = \frac{\frac{2100 \times 0.8 \times 0.8}{220\mu}}{730.5} = 8.4 \text{ [Months]}$$

¹⁰Equivalent to the average of Days in a Month in a Year

Sleep Time (sec)	Sleep Occupancy (%)	Duty Cycle (sec)	Duty Cycle Occupancy (%)	Period (sec)	Average Current (mA)	Average Power (mW)	Average Energy per bit (mJ)	Battery Life (days)	Battery Life (months)	Battery Life (years)
9.5	95.0000	0.5	5.0000	10	1.390	4.587	0.0261	40.3	1.3	0.11
29.5	98.3333	0.5	1.6667	30	0.597	1.969	0.0336	93.9	3.1	0.26
59.5	99.1667	0.5	0.8333	60	0.398	1.315	0.0449	140.6	4.7	0.39
599.5	99.9167	0.5	0.0833	600	0.220	0.725	0.2478	254.7	8.5	0.70
1799.5	99.9722	0.5	0.0278	1800	0.207	0.682	0.6986	271.0	9.0	0.74
3599.5	99.9861	0.5	0.0139	3600	0.203	0.671	1.3748	275.4	9.2	0.75
7199.5	99.9931	0.5	0.0069	7200	0.202	0.665	2.7273	277.7	9.3	0.76

Figure 4.5: Energy Analysis table and Battery Life Expectancy

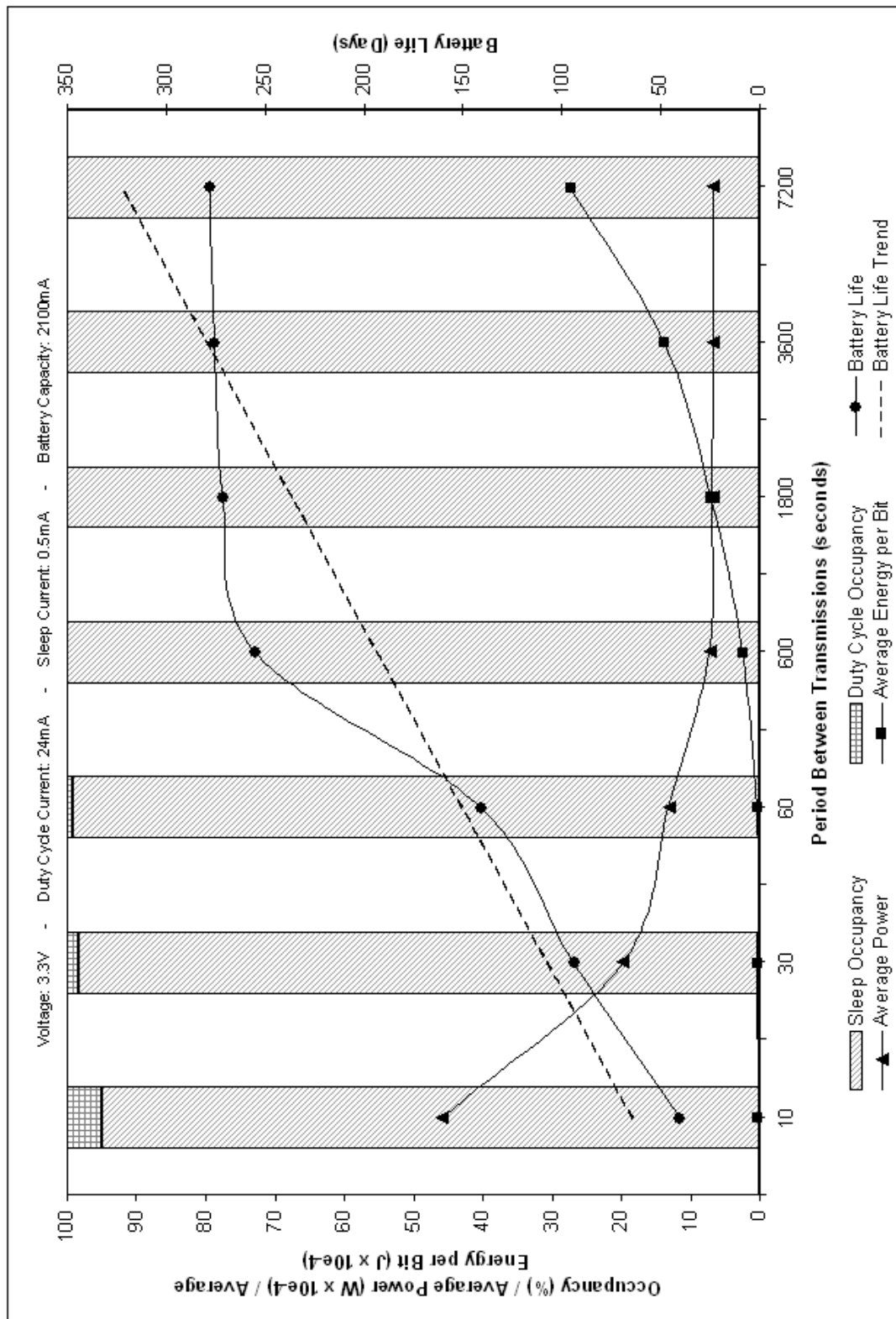


Figure 4.6: Energy Analysis and Battery Life Expectancy graphic

4.4 Firmware

Complete and detailed Source Code for all Hardware of the Project is in Appendix E. The Firmware was written with C18 Compiler. [14]

This section will focus on particularly excerpts of Source Code, that directly relates to the Energy Savings on how Firmware was written to deal with each Hardware Component of the Sensor Node.

4.4.1 System Configuration

During physical programming of the PIC chip, the internal Fuses are set to enable/disable the Peripherals and other adjustments that need to be configured for the Task proposed.

The Code that does this is:

```
_CONFIG_DECL ( _OSCS_OFF_1H & _OSC_XT_1H,  
              /* XT osc */  
              _PWRT_ON_2L & _BOR_OFF_2L & _BORV_45_2L,  
              /* BrownOut 2.5V OFF */  
              _WDT_OFF_2H & _WDTPS_128_2H,  
              /* WDOG controlled by software, Postscaler of 128 */  
              (...)
```

XT Oscillator is used, to support Lower Speed Crystals, BrownOut is Disabled to Save Power and WatchDog is enabled/disabled by Software with its Postscaler set to divide by 128.

This Postscaler gives a WatchDog Timeout of $\approx 2.3s$. The WatchDog is a Free-Running RC¹¹ Oscillator, and may drift with Temperature, thus changing Timings.

4.4.2 Main Program

Refer to Appendix E and Section E.1.

When Hardware is Powered Up, the CPU first executes the `CPU_config()` and then `System_config()`:

¹¹Resistor/Capacitor Network

```
void main (void) {  
  
    // CPU configuration and calibration  
  
    CPU_config();  
  
    // System configuration and calibration  
  
    System_config();  
}
```

The `CPU_config()` (Page 68) configures the CPU I/O Ports and Disables Analogue Comparators.

The `System_config()` (Page 68) configures the SHT11 Sensor for the Lower Resolution, Saving power, and Calibrates the CC1020 Transceiver PLL. Both this Peripherals are left in Power Down mode, to Save Power.

After CPU and System are Up and Ready to Run, the Main program sits in an Endless WHILE loop:

```
while (1)  
{  
    for ( var1=5 ; var1 != 0 ; var1--)  
    {  
        WDTCONbits.SWDTEN = 1; //WatchDog ON  
  
        Sleep();           //CPU Sleep, WakeUp when WatchDog overflow's  
        Nop();             //do nothing on the next cycle  
    }  
  
    WDTCONbits.SWDTEN = 0; //WatchDog OFF, to prevent Reset  
  
    process();           //Execute sensing and data transmission  
  
} //end while
```

Inside the WHILE loop, the program now coordinates all the Hardware functions. The PIC WatchDog is RESET each $\approx 2.3s$, Waking Up the CPU, and then goes back to sleep. This happens for VAR1 times, inside the FOR loop, adding up to the desired Interval between Transmissions. The `Nop()` makes sure that the instruction loaded while the CPU is Sleeping and the next one to be executed, is a NOP instruction.

After the FOR loop, the WatchDog is disabled to prevent further RESET's of the CPU, and the `process()` function is executed, performing the Sensor Node Jobs.

For this written Firmware Function (Page 67), the `process()` reads Data from the SHT11 Sensor, and sends all the Data it may have in the Array, through the RF Transceiver. Peripherals are left in Power Down Mode.

After the `process()` Function had done all the Jobs programmed, the Firmware returns to the FOR loop, and WatchDog Enabling.

The program repeats.

Chapter 5

Practical Analysis Discussion

During the course of the Project, some unforeseen problems had arise that conditionalized the Primary Goal of the Project. All of them related to the Hardware of the Radio Transceiver.

The Link between the Sensor Node and the Base Station, or Network, is the Radio Transceiver. As the most vital part of the System, it has to be dealt with great care. It should be Designed and Assembled, minding that it is a sensitive RF part, and if tight rules are not followed, the System may work erratically or may not work at all. Appendix C details the Radio Transceiver Schematics and PCB.

The next sections will detail the problems that were detected, and further discussions on how to solve them.

5.1 Machine Soldering

The earliest problem found was on the soldering of the CC1020 Chip [1] to the PCB.

Being a QFN¹ package type, the handling must be done by a Pick-and-Place² machine, and soldered in an Specialized Soldering Oven, after applying solder paste with the Solder Mask³ of the PCB.

As all the work that should have been done by the automatic machines would have been very expensive, all of it was done by hand on a Manual Pick-and-Place, and soldered with an automated process of Hot Air. No solder paste was used, for the PCB had Solder remaining on the pads, left by the Manufacturing Process.

¹Page 74 of [1]

²Automated Robotic Arm that Picks component and places it on the PCB

³A mask that sits on top of the PCB and only lets the solder paste go through to the soldering points

Some Chips had all pins soldered, but others did not. Corrections were made with a Soldering Iron, but with negative results. Two boards were discarded.

5.2 PLL Lock

As a sensitive part itself, the Radio Transceiver has in its core a Radio Chip that is the brain of all System. Adding some very few external passive components, and the control from the Sensor Node CPU, all the radio Operations are done without the use of a Power Hungry Radio Modem, or even a Radio itself.

The next problem detected was on the Radio Transceiver PCB's, when after all three prototypes were fully assembled, only two were working. One was working flawlessly, and one at 50%.

After some investigations with measuring tools, and reading back the Internal Status Register [1] of the Chip, the problem was found to be on the Chip's PLL Network. There is a random behavior when Calibrating the PLL, that some times Calibrate OK, and others not.

Thus, one Radio Transceiver board is working 100% as a Transceiver, and the other only works as a Transmitter, for the Receiving PLL Calibration fails.

5.3 Receiver Sensitivity

After the Simplex Scheme was put to work on the bench, some field distance tests were made, trying to map the distance with the Output Transmitted Power used.

Using the Antennas⁴ listed in Section 4.1.1, the first tests were very disappointing. With Maximum Output Transmitted Power⁵, the Received Signals using an equal Antenna, from one room to another with less than 10mts and with a concrete wall in between, were almost

⁴ ≥ 2dBi stated by manufacturer

⁵ ≥ 9dBm measured with a Spectrum Analyzer

on the Noise Level. In some cases, the Signal was Over-Modulated⁶ with local Noise, and Decoding was a failure.

5.4 PCB Temperature

The last problem discovered has to do with the Temperature surrounding the Sensor Node Radio Transceiver PCB.

It was seen that when the Temperature is increased above a certain level⁷, the Receiving End is able to decode $\geq 95\%$ of all Packets that are Received, even when Signal Level is near the Noise floor.

Decreasing Temperature would increase the BER⁸, and only some Random Packets were Properly Decoded.

5.5 Practical Analysis Discussion Summary

All the problems stated in this Chapter are thought to come from two distinct factors:

- Bad Radio Transceiver PCB Design
- Poor Crystal Characteristics

The PCB was Designed with a Powerful Auto-Router, and in spite of proper configuration of the software, the PCB should have been designed by hand, for the output stage from the CC1020 Chip to the Antenna has to follow strict track widths and lengths to maintain the 50Ω throughout. Also, the PLL Network PCB Tracks should have been Hand routed to specifications.

⁶When signal A Over-Modulates signal B, the Receiving End would not be able to Decode either of them

⁷E.g. Sensor Node System Standing in Sunlight

⁸Bit Error Rate

The CC1020 Chip Manufacturer [1] states that Crystals with ≤ 10 ppm⁹ should be used, to limit the Drifts in Frequency Oscillation.

⁹Part Per Million

Chapter 6

Conclusions

Throughout the Design of this Project, from the early stages of Writing Specifications, to choosing components, doing the last Measurements on the Hardware and Tune-Ups on the Simplex Scheme Firmware, the Main Purpose and Original Idea was strictly followed, to ensure that the Prototypes and Algorithms would end as initially thought.

Due to unforeseen Problems, the Design had to change course a few times, adding some difficulty to the written Specifications, that turned out to have slight changes at the end.

The Problems are analyzed in Chapter 5, and Solutions are also discussed. These are Solutions to these analyzed Problems, although some changes to the Hardware are preferred to Overcome the Problems.

6.1 Project Changes

With the advent of New Technology being launched every month, there is already a better solution to substitute the CC1020 Radio Transceiver Chip. This new solution [16] wins in almost every field, for the new Chip has in its Core some features like:

- Data Ready signal when a valid data package is received or transmitted
- Address Match for detection of incoming package
- Automatic retransmission of data packages
- Automatic CRC and preamble generation
- Extremely small Bill of Materials

Hence, this would be a good choice for a Future Radio Transceiver Chip, that would solve almost all Problems that arose in the Project, together with compact and easy to build Prototype PCB's.

A second change would be to do all Component Assembly using Automated Machines, to guarantee 100% Yield of the Boards.

6.2 Project Limitations

This project Limitations are where the Hardware fails. The failure of the Main functions results in the impossibility of Developing a deeper Research on the Network Schemes Proposed in Chapter 3, Section 3.1. Hence a deeper Investigation on the Energy Savings/Consumption of the Simplex Scheme, Section 3.1.1, related to the Energetic behavior of the Simplex Node, was done.

6.3 Project Future Modifications

Interesting and Useful Modifications were thought from the beginning when Designing the Schematics for both Sensor Node and Base Station.

Both are prepared to have more Sensors and/or Peripherals connected to them, enabling higher Throughput and Performance of the System as a whole.

Sensor Node is prepared with, or to have:

- Serial EEPROM for large capacity Data storage
- Battery Monitor Chip, with Gauge, to Monitor the Battery Status
- Extra Analogue and Digital Inputs available, to add external Sensors or Peripherals
- Solar Panel Input

Base Station is prepared with, or to have:

Conclusions

- Serial EEPROM for large capacity Data storage
- Onboard Temperature Sensor, to measure abnormal surrounding conditions
- Extra Digital Inputs available, to add external Peripherals
- USB and RS232 port availability

Also, improving the Sustainability and Environmentally Friendly Design, the Batteries could be substituted by a Solar Panel, or even run in Parallel with it, enabling Lower Capacity Batteries to be used, improving the Size and Portability of the Sensor Node, specially if the Hardware is to be in Protected Environments.

Further Investigations on the Impact that a Working System like this would have in some Habitats, would be valuable, presenting new solutions and a good backbone for Scientists studying Earth Life Sciences.

Appendix A

Base Station Schematic and PCB

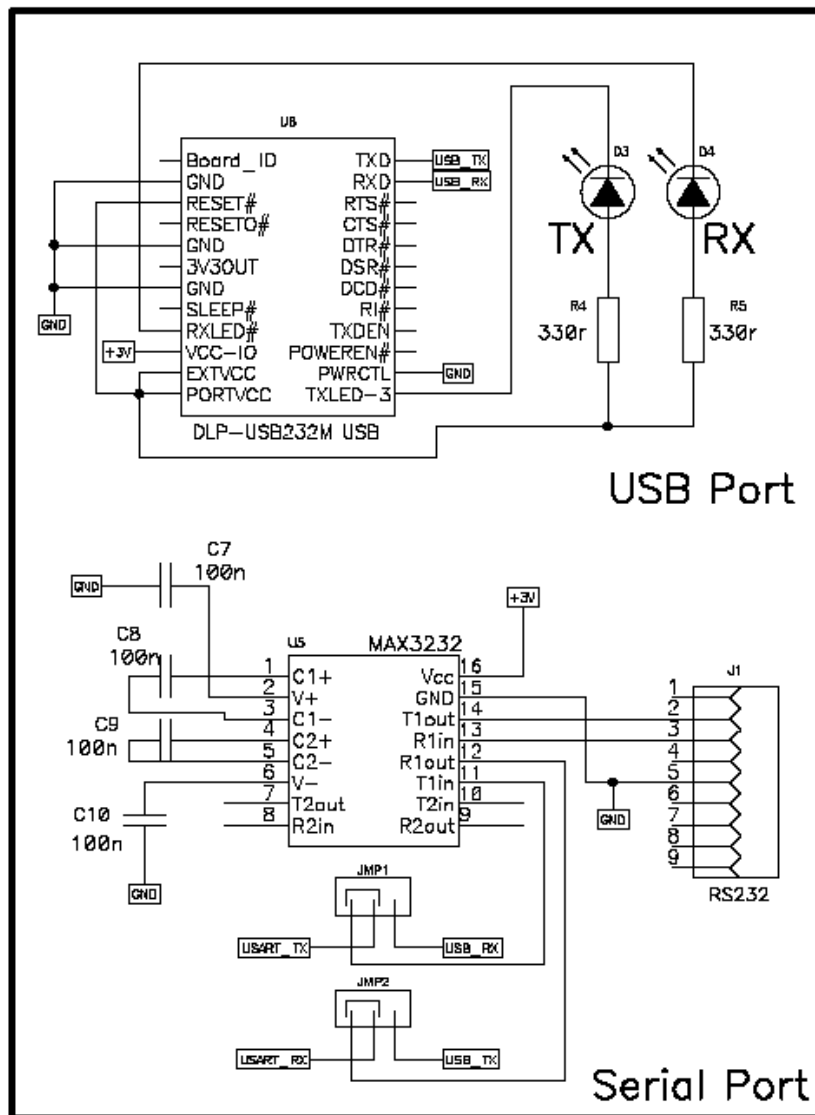


Figure A.1: Base Station PC Communications

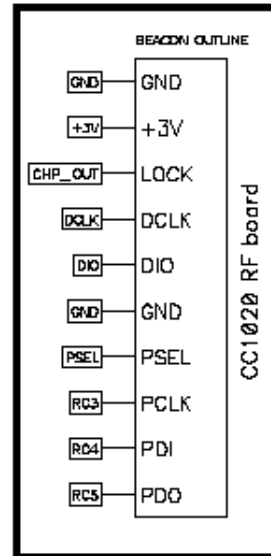
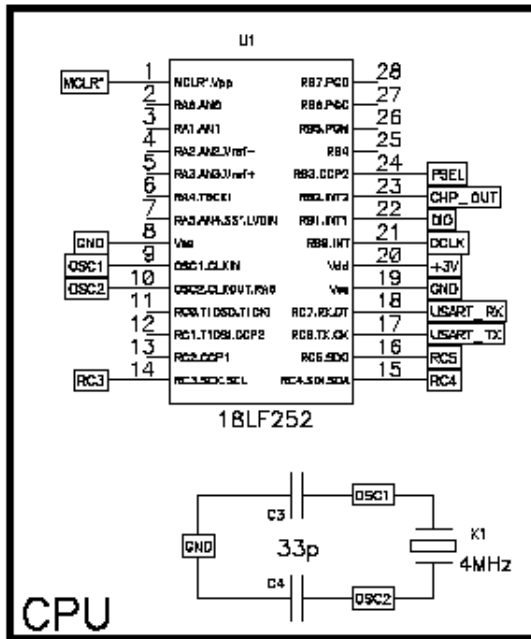
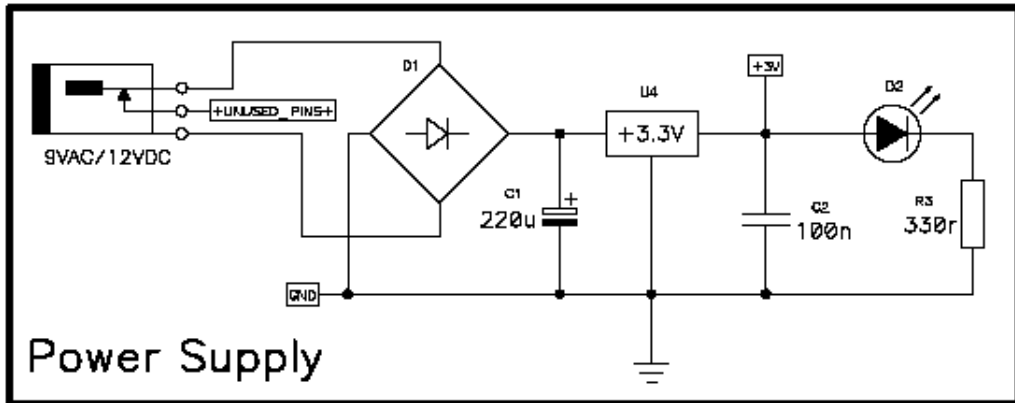


Figure A.2: Base Station CPU, Transceiver Socket and Power Supply

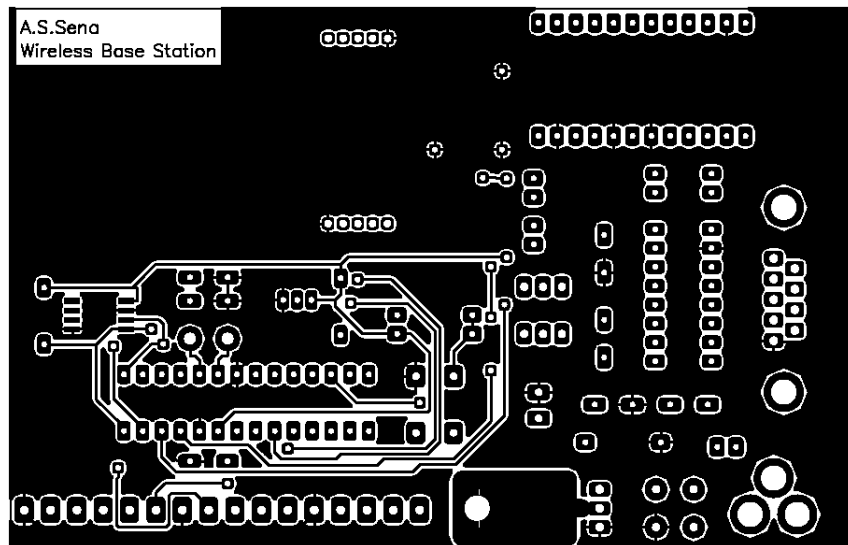


Figure A.3: Base Station Top PCB

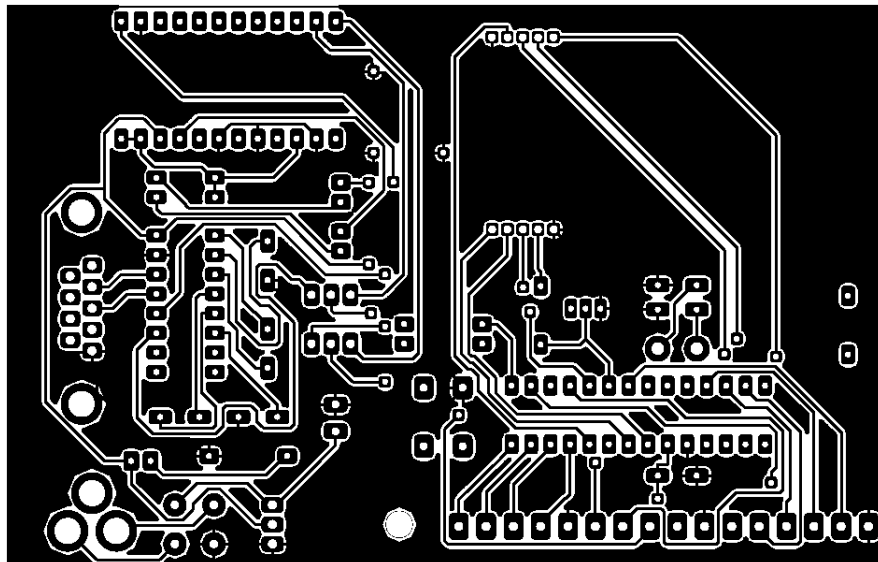


Figure A.4: Base Station Bottom PCB

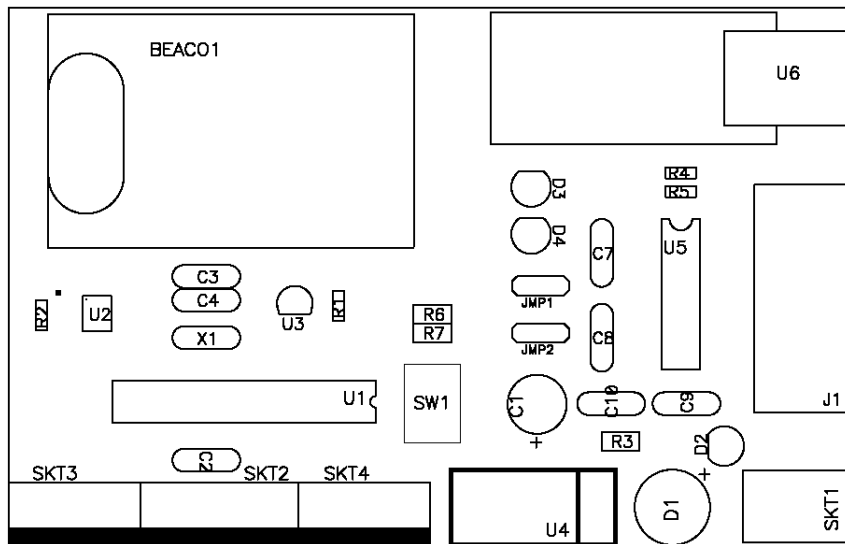


Figure A.5: Base Station Top Silk PCB

Appendix B

Sensor Node Schematic and PCB

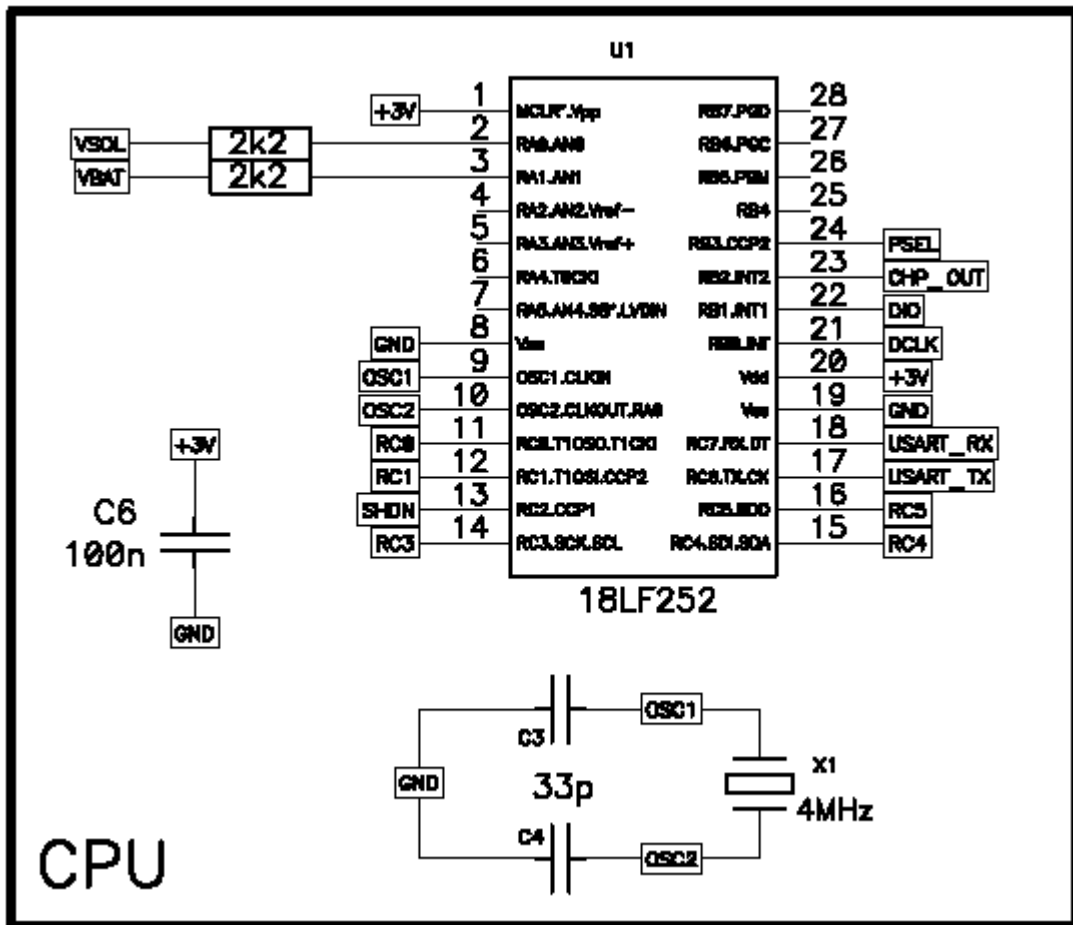


Figure B.1: Sensor Node CPU

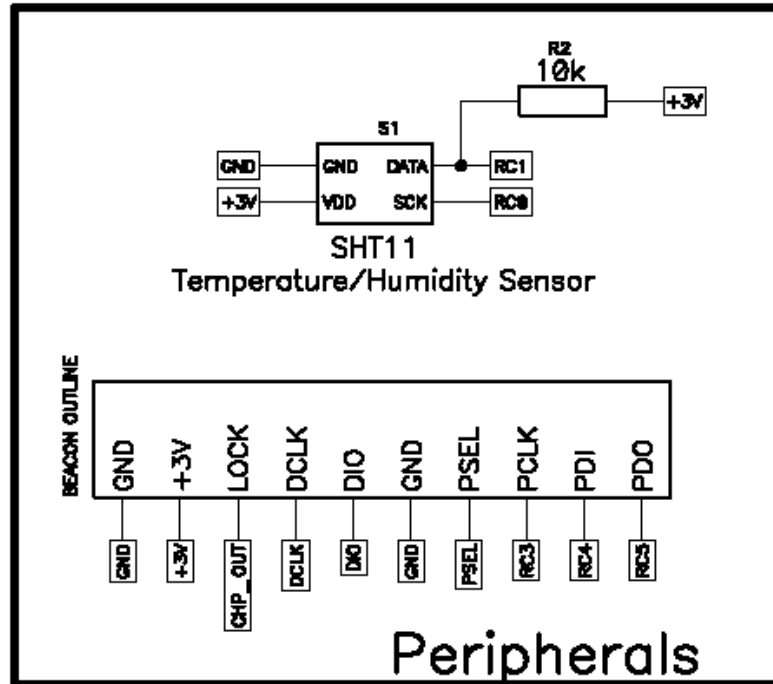


Figure B.2: Sensor Node Sensor and Transceiver Socket

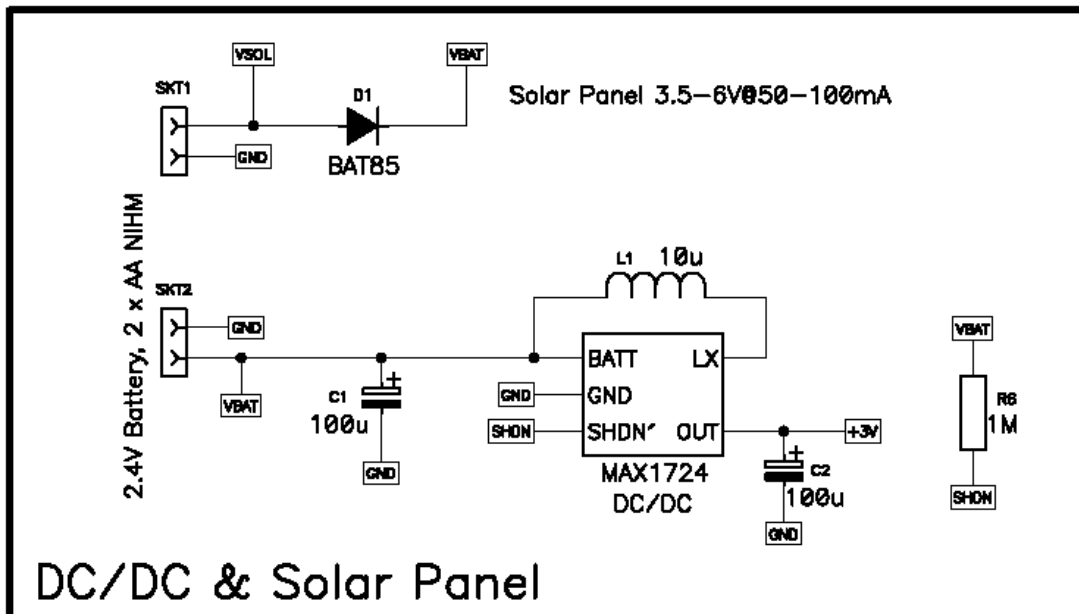


Figure B.3: Sensor Node Power Supply

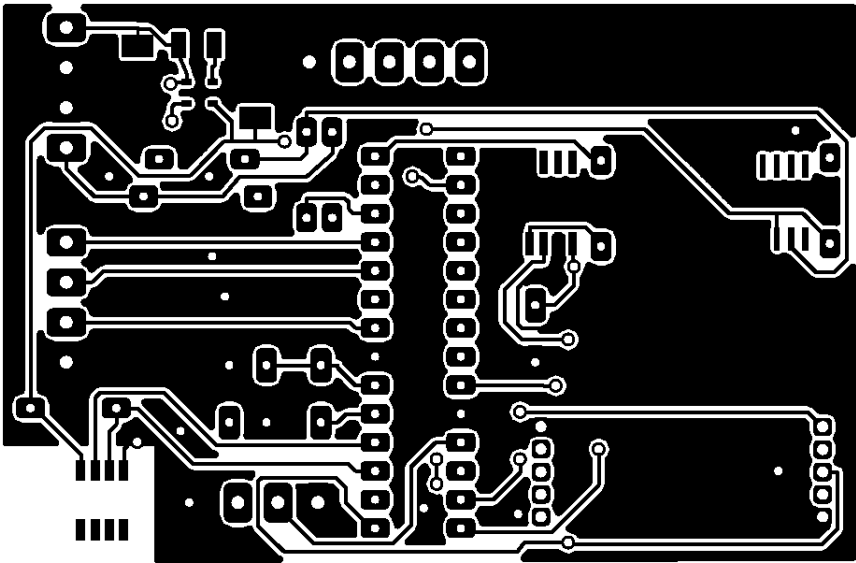


Figure B.4: Sensor Node Top PCB

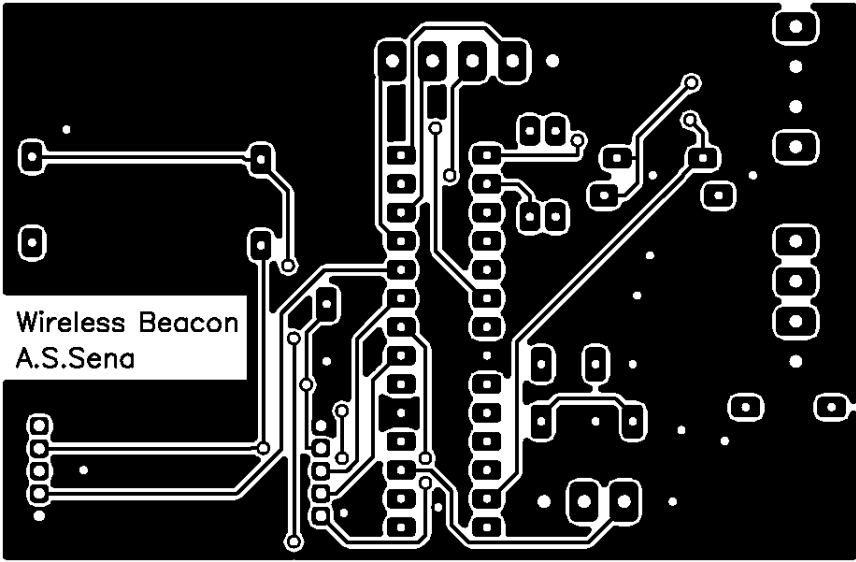


Figure B.5: Sensor Node Bottom PCB

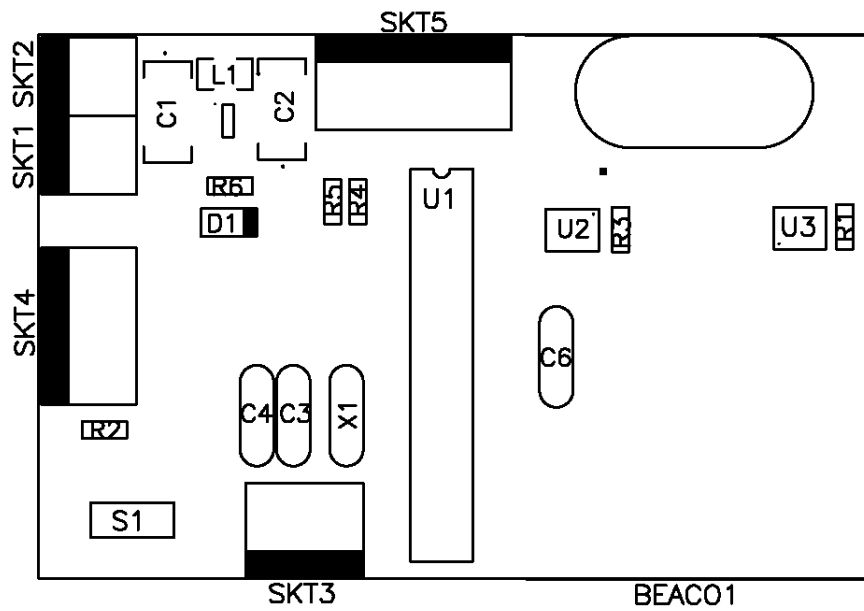


Figure B.6: Sensor Node Top Silk PCB

Appendix C

Radio Transceiver Schematic and PCB

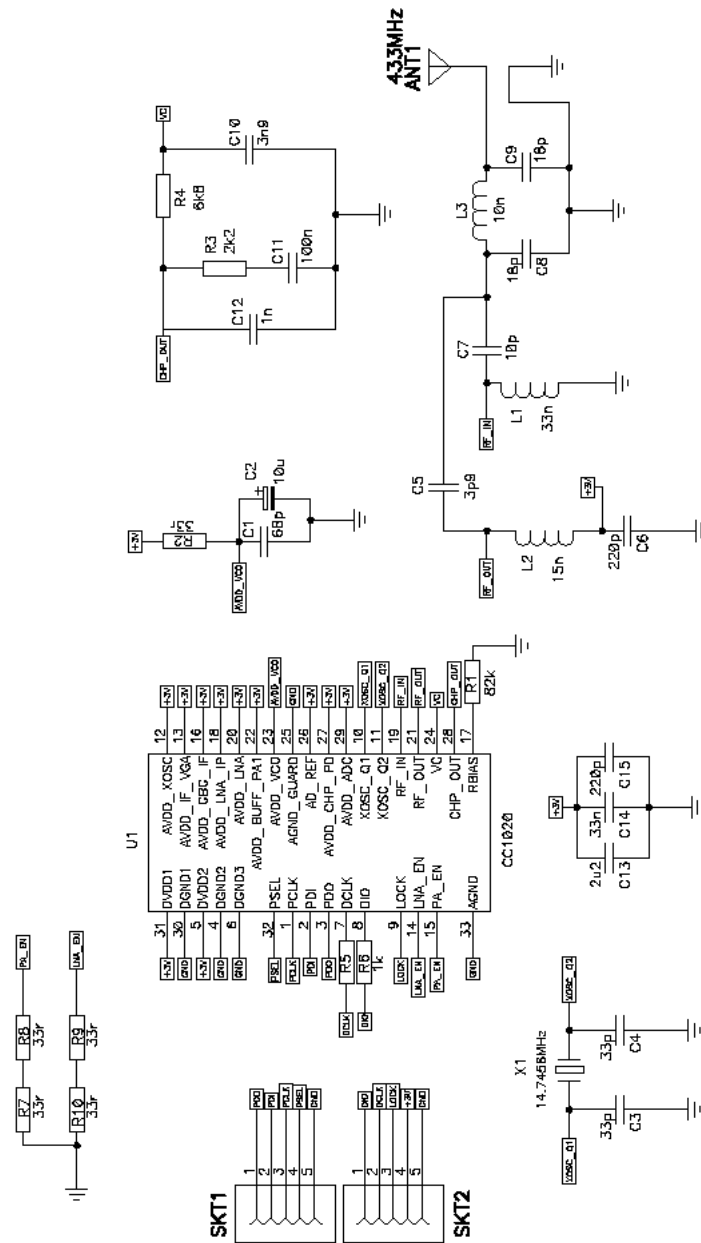


Figure C.1: Radio Transceiver Schematic

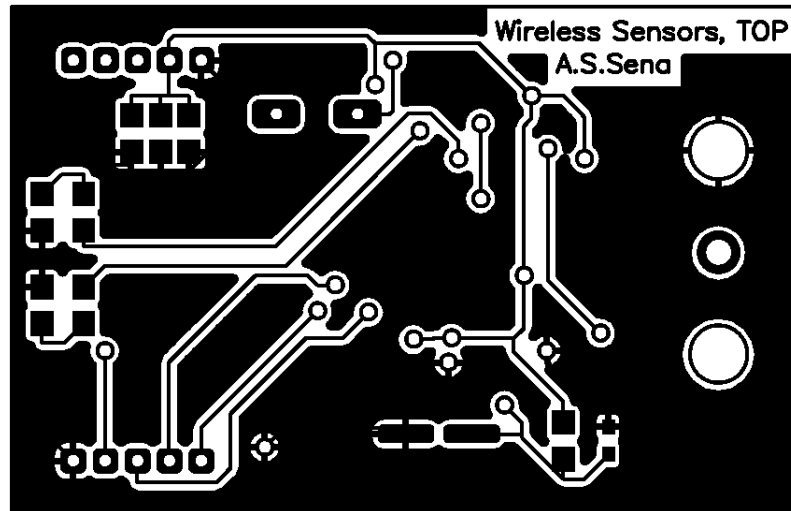


Figure C.2: Radio Transceiver Top PCB

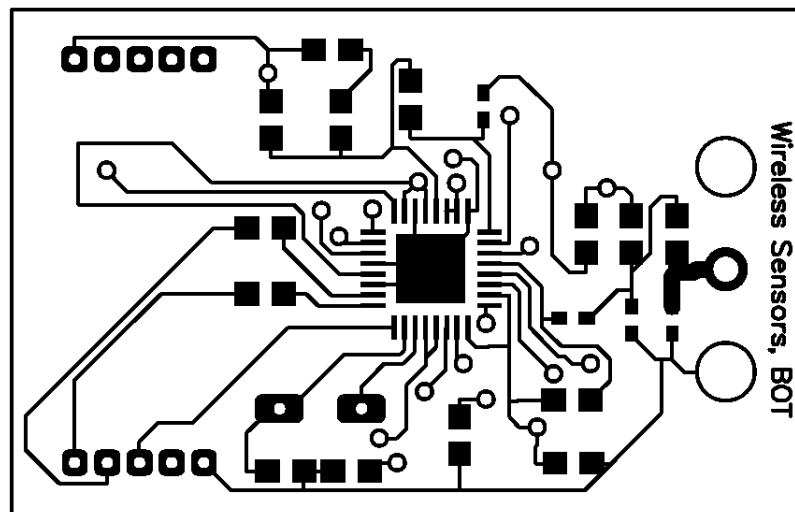


Figure C.3: Radio Transceiver Bottom PCB

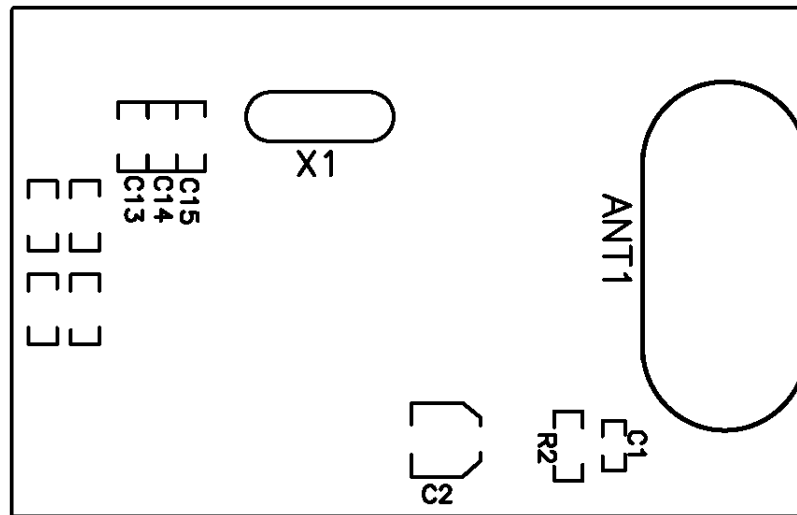


Figure C.4: Radio Transceiver Top Silk PCB

Appendix D

Bill of Materials

Base Station

Count	ComponentName	RefDes	Value
1	DB9F_TRAS	J1	RS232
1	MAX232	U5	MAX3232
1	LM1117-XX	U4	+3.3V
1	18F252	U1	18LF252
1	BEACON OUTLINE	BEAC01	
2	CAP200_FAB	C3	33p
		C4	
5	CAP200_FAB	C2	100n
		C7	
		C8	
		C9	
		C10	
1	CAP_ELEC_3.5MM	C1	220u
1	DLP-USB232M USB	U6	USB <-> RS232
2	J3	JMP1	
		JMP2	
3	LED	D2	
		D3	
		D4	

Bill of Materials

1	POWER_DC		SKT1	9VAC/12VDC
1	RECTIFIER	D1		
2	RES300_FAB		R4	330r
			R5	
1	RES400_FAB		R3	330r
1	XTAL	X1		4MHz

Sensor Node

Count	ComponentName	RefDes	Value
1	DIODE	D1	BAT85
1	18F252	U1	18LF252
1	BEACON OUTLINE	BEAC01	
2	CAP200_FAB	C3	33p
		C4	
1	CAP200_FAB	C6	100n
2	CAP_SMD_7343_43	C1	100u
		C2	
1	CRISTAL	X1	4MHz
1	IND_SMD_1210	L1	10u
1	MAX1724	PSU1	
1	RES300_FAB	R6	1M
2	RES300_FAB	R4	2k2
		R5	
1	RES300_FAB	R2	10k
1	SHT11	S1	

Bill of Materials

1	SKT_PARAF_2_3.51	SKT2	2.4V Battery, 2 x AA
1	SKT_PARAF_2_3.51	SKT1	Solar Panel 3.5-6V@5

Radio Transceiver

Count	ComponentName	RefDes	Value
1	ANTENNA_SMA_PCB	ANT1	433MHz
1	CAP_ELECTRO_4X55	C2	10u
1	CAP_SMD_603	C10	3n9
1	CAP_SMD_603	C5	3p9
2	CAP_SMD_603	C8	18p
		C9	
1	CAP_SMD_603	C1	68p
1	CAP_SMD_805	C12	1n
1	CAP_SMD_805	C13	2u2
1	CAP_SMD_805	C7	10p
1	CAP_SMD_805	C14	33n
2	CAP_SMD_805	C3	33p
		C4	
1	CAP_SMD_805	C11	100n
2	CAP_SMD_805	C6	220p
		C15	
1	CC1020	U1	
1	CRISTAL	X1	14.7456MHz
2	DIL_5X1_2MM	SKT1	
		SKT2	

Bill of Materials

1	IND_SMD_805	L3	10n
1	IND_SMD_805	L2	15n
1	IND_SMD_805	L1	33n
2	RES_SMD_805	R5	1k
		R6	
1	RES_SMD_805	R3	2k2
1	RES_SMD_805	R4	6k8
5	RES_SMD_805	R2	33r
		R7	
		R8	
		R9	
		R10	
1	RES_SMD_805	R1	82k

Appendix E

Source Code

E.1 Sensor Node Source Code

```
#include <p18f252.h> //Processor header
#include <delays.h> //Function headers
#include <usart.h>
#include <timers.h>
#include "cc1020.h" //CC1020 headers
#include "sht11.h" //SHT11 headers

#define configSD TRISCbits.TRISC2 //ShutDown pin IO setting
#define SD PORTCbits.RC2 //ShutDown pin IO

#pragma code

unsigned char i; //Assign RAM registers
unsigned int var1;
volatile unsigned char DataString[16] = {0x00,0x01,0x02,0x03,0x04,0x05,
                                         0x06,0x07,0x08,0x09,0x0a,
                                         0x0b,0x0c,0x0d,0x0e,0x0f};//

void process(void); //function prototypes
void CPU_config(void); //
void System_config(void); //

/*****
/*****
/* Main program starts here
/*****
/*****

void main (void) {

    // CPU configuration and calibration

    CPU_config();

    // System configuration and calibration

    System_config();

/*****
/* Main program code
/*****

    while (1)
    {
```

Source Code

```
/* FOR cycle to generate the necessary delay.
   WatchDog used with:: VAR1 variable, Postscaler of 128 (2.3seg)
   -> VAR1 x 128 X WATCHDOG Time, giving always random values
   around like ::
   VAR1 = 4 = ~9.2s
   VAR1 = 13 = ~30s
   VAR1 = 131 = ~5min
   VAR1 = 783 = ~30min
   VAR1 = 1565 = ~60min
*/

for ( var1=5 ; var1 != 0 ; var1--)
{
    WDTCNbits.SWDTEN = 1; //WatchDog ON

    Sleep();           //CPU Sleep, WakeUp when WatchDog overflow's
    Nop();             //do nothing on the next cycle
}

WDTCNbits.SWDTEN = 0; //WatchDog OFF, to prevent Reset

process();           //Execute sensing and data transmission

} //end while

}

/*****
/* End of Main program
*****/

#pragma code

/*****
/* Code to execute for sensing and data transmission
*****/

void process(void)
{
    /*** SHT11 action *****/

    s_transstart(); //transmission start
    s_write_byte(MEASURE_TEMP); //measure Temperature
    while (DATA); //wait until SHT11 ends measurements
    //Save highest byte read, in RAM
    DataString[11] = (unsigned char) s_read_byte(ACK);
    //Save lowest byte read, in RAM
    DataString[12] = (unsigned char) s_read_byte(noACK);

    s_connectionreset(); //Reset Sensor BUS
    s_transstart(); //transmission start
    s_write_byte(MEASURE_HUMI); //measure Humidity
    while (DATA); //wait until SHT11 ends measurements
    //Save highest byte read, in RAM
    DataString[13] = (unsigned char) s_read_byte(ACK);
    //Save lowest byte read, in RAM
    DataString[14] = (unsigned char) s_read_byte(noACK);
}

```

Source Code

```
/** CC1020 action *****/
//Prepare CC1020 to transmission
WakeUpCC1020ToTX(TXANALOG_VALUE);
SetupCC1020TX(TXANALOG_VALUE, PA_POWER);

PREAMBLE(); // send PREAMBLE
SendDataString(&DataString); //send data string

SetupCC1020PD(); //power down CC1020
}
/*****
/* End of sensing and data transmission routine
*****/

/*****
/*****
/* Code to execute on CPU Configuration
/*****
/*****
void CPU_config(void)
{
    TRISA = 0x10; //RA4 as input (1-wire)
    TRISB = 0x05; //RB0 and RB2 as input (CC1020 DCLK and LOCK)
    TRISC = 0x14; //RC4 and RC2 as input (CC1020 PDO and SMPS ShutDown)

    PORTA = 0; //Clear all ports
    PORTB = 0;
    PORTC = 0;

    ADCON0 = 0; // A/D converter module shut off
    ADCON1 = 0x07; // All pins as Digital IO's
}
/*****
/* End of Code to execute on CPU Configuration
*****/

/*****
/*****
/* Code to execute on System Configuration
/*****
/*****
void System_config(void)
{
    // SHT11 sensor configuration

    Delay1KTCYx(13); //wait 13ms before Sensirion's RESET, start up
    s_connectionreset(); //reset sensor
    s_write_statusreg(0x03); //low resolution data

    // CC1020 configuration and calibration

    SetupCC1020ForSPI(); //setup PIC for SPI comms

    SetupCC1020PD(); //PowerDown CC1020
    ResetCC1020(); //Reset CC1020
```

Source Code

```
ConfigureCC1020(); //Configure CC1020 internal registers

WakeUpCC1020ToTX(TXANALOG_VALUE); //Calibrate PLL for TX
CalibrateCC1020(PA_POWER);

WakeUpCC1020ToRX(RXANALOG_VALUE); //Calibrate PLL for RX
CalibrateCC1020(PA_POWER);

SetupCC1020PD(); //PowerDown CC1020
}
/*****
/* End of Code to execute on System Configuration
*****/

#pragma code
/*****
*****/
/* Main program - END
*****/
*****/

#pragma romdata CONFIG
_CONFIG_DECL ( _OSCS_OFF_1H & _OSC_XT_1H,
              /* XT osc */
              _PWRT_ON_2L & _BOR_OFF_2L & _BORV_45_2L,
              /* BrownOut 2.5V OFF */
              _WDT_OFF_2H & _WDTPS_128_2H,
              /* WDOG controlled by software, Postscaler of 128 */
              _CCP2MUX_OFF_3H,
              _STVR_OFF_4L & _LVP_OFF_4L & _DEBUG_OFF_4L,
              _CONFIG5L_DEFAULT,
              _CONFIG5H_DEFAULT,
              _CONFIG6L_DEFAULT,
              _CONFIG6H_DEFAULT,
              _CONFIG7L_DEFAULT,
              _CONFIG7H_DEFAULT
              );
#pragma romdata
```

E.2 Base Station Source Code

```
#include <p18f252.h> //Processor header
#include <delays.h> //Function headers
#include <usart.h>
#include "cc1020.h" //CC1020 headers

unsigned char i;
volatile unsigned char DataString[20];

#pragma code

/*****
/*****
/* Main program starts here
/*****
/*****

void main (void)
{

/*****
/* CPU configuration and calibration
/*****

    TRISA = 0x01; //
    TRISB = 0x05; //RB0 and RB2 as input (CC1020 DCLK and LOCK)
    TRISC = 0x10; //RC4 as input (CC1020 PDO)

    PORTA = 0; //Clear ports
    PORTB = 0;
    PORTC = 0;

    ADCON0 = 0; // AD converter module shut off
    ADCON1 = 0x07; // All pins as Digital IO's

    // Open USART
    // spbrg = 25 -> 9600bps, 4MHz Crystal
    // spbrg = 64 -> 9600bps, 10MHz Crystal
    // spbrg = 12 -> 19200bps, 4MHz Crystal
    OpenUSART( USART_TX_INT_OFF & USART_RX_INT_OFF &
               USART_ASYNCH_MODE & USART_EIGHT_BIT &
               USART_CONT_RX & USART_BRGH_HIGH, 12);

/*****
/* CC1020 configuration and calibration
/*****

    SetupCC1020ForSPI(); //setup SPI on the PIC

    SetupCC1020PD(); //put CC1020 in Power Down
    ResetCC1020(); //reset CC1020
    ConfigureCC1020(); //configure CC1020 internal registers

// The following routines calibrate PLL for both Transmit and Receive
//
    WakeUpCC1020ToTX(TXANALOG_VALUE); //wake up CC1020 to Transmit
    if (!CalibrateCC1020(PA_POWER)) //Calibrate and return status
        putsUSART("TX Calibration failed", 21);
```

Source Code

```
else
    putsUSART("TX Calibration success",22);

WakeUpCC1020ToRX(RXANALOG_VALUE); //wake up CC1020 to Receive
if (!CalibrateCC1020(PA_POWER)) //Calibrate and return status
    putsUSART("RX Calibration failed",21);
else
    putsUSART("RX Calibration success",22);

/*****
/* Put CC1020 in RX mode
*****/

SetupCC1020RX(RXANALOG_VALUE, PA_POWER); //Put CC1020 in Receiving Mode

/*****
/* Main program
*****/

while (1)
{

    Delay100TCYx(9); //wait before requesting new RSSI value

    //For RSSI values bigger or equal than 0x25, decode Packet
    if ((ReadFromCC1020Register(CC1020_RSSI)) >= 0x25)
    {
        WaitPREAMBLE(); //Wait for packets with Preamble
        PORTBbits.RB7 = 1; //monitor light on
        ReadDataCC1020(&DataString); //get Data String, load it into
        // the array
        PORTBbits.RB7 = 0; //monitor light off

        for (i=0;i<20;i++) //monitor received array
        // out serial port
        {
            while(BusyUSART());
            WriteUSART(DataString[i]);
        }
    }

}

} //end while

}

#pragma code
/*****
*****/
/* Main program - END
*****/
/*****
*****/

#pragma romdata CONFIG
_CONFIG_DECL ( _OSCS_OFF_1H & _OSC_HS_1H,
    /* XT osc */
    _PWRT_ON_2L & _BOR_ON_2L & _BORV_27_2L,
    /* BrownOut 2.5V OFF */
    _WDT_OFF_2H & _WDTPS_1_2H,
    /* WDOG controlled by software, Postscaler of 128 */
    _CCP2MUX_OFF_3H,
    _STVR_OFF_4L & _LVP_OFF_4L,
```

Source Code

```
        _CONFIG5L_DEFAULT,  
        _CONFIG5H_DEFAULT,  
        _CONFIG6L_DEFAULT,  
        _CONFIG6H_DEFAULT,  
        _CONFIG7L_DEFAULT,  
        _CONFIG7H_DEFAULT  
    );  
#pragma romdata
```

E.3 RF Transceiver CC1020 Source Code

```
#ifndef __cc1020_H
#define __cc1020_H

/*****
/* Pin usage definitions */

#define PDO          PORTCbits.RC4    //SPI bus with CC1020
#define PDI          PORTCbits.RC5
#define PCLK         PORTCbits.RC3
#define PSEL         PORTBbits.RB3
#define DIO          PORTBbits.RB1    //Serial IO bus with CC1020
#define DCLK         PORTBbits.RB0
#define TRIS_DIO     TRISBbits.TRISB1
#define TRIS_DCLK    TRISBbits.TRISB0
#define CHP_OUT      PORTBbits.RB2    //Monitor out from CC1020

/*****
/* Register addresses definitions */

#define CC1020_MAIN          0x00
#define CC1020_INTERFACE    0x01
#define CC1020_RESET        0x02
#define CC1020_SEQUENCING   0x03
#define CC1020_FREQ_2A      0x04
#define CC1020_FREQ_1A      0x05
#define CC1020_FREQ_0A      0x06
#define CC1020_CLOCK_A      0x07
#define CC1020_FREQ_2B      0x08
#define CC1020_FREQ_1B      0x09
#define CC1020_FREQ_0B      0x0A
#define CC1020_CLOCK_B      0x0B
#define CC1020_VCO          0x0C
#define CC1020_MODEM        0x0D
#define CC1020_DEVIATION    0x0E
#define CC1020_AFC_CONTROL  0x0F
#define CC1020_FILTER       0x10
#define CC1020_VGA1         0x11
#define CC1020_VGA2         0x12
#define CC1020_VGA3         0x13
#define CC1020_VGA4         0x14
#define CC1020_LOCK         0x15
#define CC1020_FRONTEND     0x16
#define CC1020_ANALOG       0x17
#define CC1020_BUFF_SWING   0x18
#define CC1020_BUFF_CURRENT 0x19
#define CC1020_PLL_BW       0x1A
#define CC1020_CALIBRATE    0x1B
#define CC1020_PA_POWER     0x1C
#define CC1020_MATCH        0x1D
#define CC1020_PHASE_COMP   0x1E
#define CC1020_GAIN_COMP    0x1F
#define CC1020_POWERDOWN    0x20
#define CC1020_TEST1        0x21
#define CC1020_TEST2        0x22
#define CC1020_TEST3        0x23
#define CC1020_TEST4        0x24
#define CC1020_TEST5        0x25
#define CC1020_TEST6        0x26
```

Source Code

```
#define CC1020_TEST7          0x27
#define CC1020_STATUS        0x40
#define CC1020_RESET_DONE    0x41
#define CC1020_RSSI          0x42
#define CC1020_AFC           0x43
#define CC1020_GAUSS_FILTER  0x44
#define CC1020_STATUS1       0x45
#define CC1020_STATUS2       0x46
#define CC1020_STATUS3       0x47
#define CC1020_STATUS4       0x48
#define CC1020_STATUS5       0x49
#define CC1020_STATUS6       0x4A
#define CC1020_STATUS7       0x4B

// LOCK status definitions
#define LOCK_NOK              0x00
#define LOCK_OK               0x01
#define LOCK_RECAL_OK         0x02

// PA power setting
#define PA_POWER 0xff

/* Functions for accessing the CC1020 */

/*****
*****
*****
*****
*****
*****/

const short Config433[32] = {
    0x010F, // 0x01, INTERFACE
    0x02FF, // 0x02, RESET
    0x038F, // 0x03, SEQUENCING
    0x043A, // 0x04, FREQ_2A
    0x050e, // 0x05, FREQ_1A
    0x069d, // 0x06, FREQ_0A
    0x073A, // 0x07, CLOCK_A   Manchester @ 4800bps
    0x083A, // 0x08, FREQ_2B
    0x0919, // 0x09, FREQ_1B
    0x0A47, // 0x0A, FREQ_0B
    0x0B3A, // 0x0B, CLOCK_B   Manchester @ 4800bps
    0x0C44, // 0x0C, VCO
    0x0D51, // 0x0D, MODEM      Manchester
    0x0E1B, // 0x0E, DEVIATION  FSK
    0x0FCC, // 0x0F, AFC_CONTROL Manchester @ 4800bps
    0x102F, // 0x10, FILTER
    0x1165, // 0x11, VGA1
    0x1257, // 0x12, VGA2
    0x132F, // 0x13, VGA3
    0x141e, // 0x14, VGA4   Carrier Sense Normal and RSSI level
                // to activate CARRIER SENSE
    0x1540, // 0x15, LOCK   Carrier Sense Pin Enable
    0x1678, // 0x16, FRONTEND
    0x1746, // 0x17, ANALOG
    0x1854, // 0x18, BUFF_SWING
    0x1922, // 0x19, BUFF_CURRENT
    0x1AAE, // 0x1A, PLL_BW
    0x1B35, // 0x1B, CALIBRATE
    0x1C0E, // 0x1C, PA_POWER
    0x1D00, // 0x1D, MATCH
```

Source Code

```
0x1E00, // 0x1E, PHASE_COMP
0x1F00, // 0x1F, GAIN_COMP
0x2000 // 0x20, POWERDOWN
};

const char TXANALOG_VALUE = 0x46;
const char RXANALOG_VALUE = 0x46;

/*****
/*****
/*****

/*****
/* This routine sets up the CC1020 for SPI transfer */
/*****

void SetupCC1020ForSPI(void)
{
    SSPSTAT=0x40;          //Setup PIC for SPI comms
    SSPCON1=0x20;
}

/*****
/* This routine writes to a single CC1020 register */
/*****

void WriteToCC1020Register(char addr, char data)
{
    char dummy;

    PSEL=0;          //Select Chip

    dummy=SSPBUF; //Clear Buffer
    SSPBUF=(addr<<1)|0x01; // Write address to CC1020, write bit is always 1

    // Wait until data is written
    while (SSPSTATbits.BF==0); //While receive not complete

    dummy=SSPBUF; //Clear Buffer
    SSPBUF=data; //Write to SPI bus
    while (SSPSTATbits.BF==0); //While receive not complete

    PSEL=1;          //Unselect Chip
}

/*****
/* This routine writes to a single CC1020 register, with data and address */
/* given in the same variable */
/*****

void WriteToCC1020RegisterWord(short addranddata)
{
    char dummy;

    PSEL=0;          //Select Chip

    dummy=SSPBUF; //Clear Buffer
    SSPBUF = (((char)((addranddata>>8)&0x00FF))<<1)|0x01; //Write to SPI bus

    // Wait until data is written
    while (SSPSTATbits.BF==0); //While receive not complete
```

Source Code

```
dummy=SSPBUF; //Clear Buffer
SSPBUF = (char)(addr&data&0x00FF); //Write to SPI bus

// Wait until data is written
while (SSPSTATbits.BF==0); //While receive not complete

PSEL=1; //Unselect Chip
}

/*****
/* This routine reads from a single CC1020 register */
*****/

char ReadFromCC1020Register(char addr)
{
    char Value;

    PSEL=0; //Select Chip
    Value=SSPBUF; //Clear Buffer
    SSPBUF=(addr<<1)&0xFE; // Write address to CC1020, write bit is always 0

    // Wait until data is written
    while (SSPSTATbits.BF==0);
    SSPCON1bits.SSP0V=0;

    SSPBUF=0xFF; // Dummy write

    while (SSPSTATbits.BF==0); //While receive not complete
    Value=SSPBUF;

    PSEL=1; //Unselect Chip
    return Value; //Return read Data
}

/*****
*****/

/*****
/* This routine sends new configuration data to the CC1020 */
*****/

void ConfigureCC1020(void)
{
    short val;
    char i;

    for (i=1;i<=32;i++) //Send 32 configuration values to CC1020
    {
        val=Config433[i-1];
        WriteToCC1020RegisterWord(val);
        Delay1KTCYx(1);
    }
}

/*****
/* This routine resets the CC1020, clearing all registers. */
*****/

void ResetCC1020(void)
{
```

Source Code

```
// Reset CC1020
WriteToCC1020Register(CC1020_MAIN, 0x0F&~0x01);

// Bring CC1020 out of reset
WriteToCC1020Register(CC1020_MAIN, 0x1F);
}

/*****
/* This routine calibrates the CC1020 */
/* Returns 0 if calibration fails, non-zero otherwise. Checks the LOCK */
/* to check for success. */
*****/

char CalibrateCC1020(char PA_VALUE)
{
    volatile int TimeOutCounter;

    // Turn off PA to avoid spurs during calibration in TX mode
    WriteToCC1020Register(CC1020_PA_POWER,0x00);

    // Start calibration
    WriteToCC1020Register(CC1020_CALIBRATE,0xB5);

    // Monitor calibration
    while ((ReadFromCC1020Register(CC1020_STATUS)&0x80) == 0);

    // Monitor Lock
    while ((ReadFromCC1020Register(CC1020_STATUS)&0x40) != 0);
    Delay100TCYx(25); //2.5msec

    // Restore PA setting
    WriteToCC1020Register(CC1020_PA_POWER,PA_VALUE);

    // Return state of LOCK_CONTINUOUS bit43x
    return ((ReadFromCC1020Register(CC1020_STATUS)&0x20)==0x20);
}

/*****
/* This routine puts the CC1020 into RX mode (from TX). When switching to */
/* RX from PD, use WakeupC1020ToRX first */
*****/

char SetupCC1020RX(char RXANALOG, char PA_VALUE)
{
    volatile int TimeOutCounter; //Temporal registers
    char lock_status;

    // Switch into RX, switch to freq. reg A
    WriteToCC1020Register(CC1020_MAIN,0x11);

    // Setup bias current adjustment
    WriteToCC1020Register(CC1020_ANALOG,RXANALOG);

    // Wait for 5 msec
    Delay1KTCYx(5); //5msec

    // Monitor LOCK
    while ((ReadFromCC1020Register(CC1020_STATUS)&0x40) != 0);
    Delay100TCYx(25); //2.5msec

    // If PLL in lock
    if((ReadFromCC1020Register(CC1020_STATUS)&0x10)==0x10){
```

Source Code

```
// Indicate PLL in LOCK
lock_status = LOCK_OK;
// Else (PLL out of LOCK)
}else{
// If recalibration ok
if(CalibrateCC1020(PA_VALUE)){
// Indicate PLL in LOCK
lock_status = LOCK_RECAL_OK;
// Else (recalibration failed)
}else{
// Indicate PLL out of LOCK
lock_status = LOCK_NOK;
}
}

// Switch RX part of CC1020 on
WriteToCC1020Register(CC1020_MAIN,0x01);

// Return LOCK status to application
return (lock_status);
}

/*****
/* This routine puts the CC1020 into TX mode (from RX). When switching to */
/* TX from PD, use WakeupCC1020ToTX first */
*****/

char SetupCC1020TX(char TXANALOG, char PA_VALUE)
{
volatile int TimeOutCounter;
char lock_status;

// Turn off PA to avoid frequency splatter
WriteToCC1020Register(CC1020_PA_POWER,0x00);

// Setup bias current adjustment
WriteToCC1020Register(CC1020_ANALOG,TXANALOG);

// Switch into TX, switch to freq. reg B
WriteToCC1020Register(CC1020_MAIN,0xC1);

// Wait for 5msec
Delay1KTCYx(5); //5msec

// Monitor LOCK
while ((ReadFromCC1020Register(CC1020_STATUS)&0x40) != 0);
Delay100TCYx(25); //2.5msec

// If PLL in lock
if((ReadFromCC1020Register(CC1020_STATUS)&0x10)==0x10){
// Indicate PLL in LOCK
lock_status = LOCK_OK;
// Else (PLL out of LOCK)
}else{
// If recalibration ok
if(CalibrateCC1020(PA_VALUE)){
// Indicate PLL in LOCK
lock_status = LOCK_RECAL_OK;
// Else (recalibration failed)
}else{
// Indicate PLL out of LOCK
lock_status = LOCK_NOK;
}
}
}
```

Source Code

```
}

// Restore PA setting
WriteToCC1020Register(CC1020_PA_POWER,PA_POWER);

// Turn OFF DCLK squelch in TX
WriteToCC1020Register(
    CC1020_INTERFACE,ReadFromCC1020Register(CC1020_INTERFACE)&~0x10);

// Return LOCK status to application
return (lock_status);
}

/*****
/* This routine puts the CC1020 into power down mode. Use WakeUpCC1020ToRX */
/* followed by SetupCC1020RX or WakeupCC1020ToTX followed by SetupCC1020TX */
/* to wake up from power down */
*****/

void SetupCC1020PD(void)
{
    // Put CC1020 into power-down
    WriteToCC1020Register(CC1020_MAIN,0x1F);

    // Turn off PA to minimise current draw
    WriteToCC1020Register(CC1020_PA_POWER,0x00);
}

/*****
/* This routine wakes the CC1020 up from PD mode to RX mode */
*****/

void WakeUpCC1020ToRX(char RXANALOG)
{
    volatile int i;

    // Turn on xtal oscillator core
    WriteToCC1020Register(CC1020_MAIN,0x1B);

    // Setup bias current adjustment
    WriteToCC1020Register(CC1020_ANALOG,RXANALOG);

    Delay1KTCYx(5); //5msec

    // Turn on bias generator
    WriteToCC1020Register(CC1020_MAIN,0x19);

    Delay10TCYx(15); //150usec

    // Turn on frequency synthesier
    WriteToCC1020Register(CC1020_MAIN,0x11);
}

/*****
/* This routine wakes the CC1020 up from PD mode to TX mode */
*****/

void WakeUpCC1020ToTX(char TXANALOG)
{
    // Turn on xtal oscillator core
    WriteToCC1020Register(CC1020_MAIN,0xDB);
```

Source Code

```
// Setup bias current adjustment
WriteToCC1020Register(CC1020_ANALOG, TXANALOG);

Delay1KTCYx(5); //5msec

// Turn on bias generator
WriteToCC1020Register(CC1020_MAIN, 0xD9);

Delay10TCYx(15); //150usec

// Turn on frequency synthesiser
WriteToCC1020Register(CC1020_MAIN, 0xD1);
}
/*****
/*****

/*****
/* This routine writes data to CC1020, serial IO */
/*****

void WriteDataCC1020(unsigned char buffer)
{
    unsigned char BitCounter;

    for (BitCounter=0 ; BitCounter<8 ; BitCounter++) //Writes 8 bits
    {
        if (DCLK == 0) //Wait for Low Clock to finish
        {
            if ((buffer & 0x80) == 0x80) //If last bit
                DIO = 1;
            else
                DIO = 0;

            while (DCLK == 0); //Wait for Low Clock to finish
        }
        else
        {
            while (DCLK == 1); //Wait for High Clock to finish

            if ((buffer & 0x80) == 0x80) //If last bit
                DIO = 1;
            else
                DIO = 0;

            while (DCLK == 0); //wait for the 0 to end
        }

        buffer = buffer << 1; //Rotate Buffer to the Left
    }
}
/*****

/*****
/* This routine reads data of CC1020, serial IO bitbang */
/*****

void ReadDataCC1020(volatile unsigned char *a)
{
    unsigned char BitCounter, buffer, i;
```

Source Code

```
TRIS_DIO = 1; //DIO as input

for (i=0;i<20;i++) //read N arriving bytes
{
    buffer = 0;

    //read arriving byte from CC1020
    for (BitCounter=0 ; BitCounter<8 ; BitCounter++)
    {
        if (DIO == 1) //Execute if bit is 1
        {
            buffer |= 0x01; //Load Buffer with another 1 bit
        }

        if (BitCounter != 7) //Dont do this on the last bit
        {
            buffer = buffer << 1; //Rotate Buffer to the Left
        }

        while (DCLK == 1); //Wait for High Clock to finish
        while (DCLK == 0); //Wait for Low Clock to finish
    }

    *(a+i) = buffer; //Save received Buffer into Array
}

while (!CHP_OUT); //Wait until carrier fades out, neglects extra data

TRIS_DIO = 0; //DIO as output
}
/*****/

/*****/
/* This routine checks for PREAMBLE, and waits for it to finish */
/*****/

void WaitPREAMBLE (void)
{
    unsigned int buffer=0;

    TRIS_DIO = 1; //DIO as input

    while (buffer != 0x33)
    {
        while (DCLK == 1); //wait for the 1 to end
        while (DCLK == 0); //wait for the 0 to end

        if (DIO == 1) //Execute if bit is 1
        {
            buffer |= 0x80;
        }

        buffer = buffer >> 1; //Rotate Buffer to the Right
    }

    TRIS_DIO = 0; //DIO as output
}
/*****/
```

Source Code

```

/*****
/* This routine performs PREAMBLE */
*****/

void PREAMBLE (void)
{
    unsigned char i;

    for (i=0 ; i < 40 ; i++)
    {
        WriteDataCC1020(0xaa); //writes X bytes of "10101010"
    }

    WriteDataCC1020(0x33); //writes last byte and synchronizer
    WriteDataCC1020(0x33); //writes last byte and synchronizer
    WriteDataCC1020(0x33); //writes last byte and synchronizer
    WriteDataCC1020(0x33); //writes last byte and synchronizer

}
/*****/

/*****/
/* This routine sends the String of Data */
*****/

void SendDataString (volatile unsigned char *a)
{
    unsigned char i, buffer;

    for (i=0 ; i < 16 ; i++) //Send 16 bytes out
    {
        buffer = *(a+i);
        WriteDataCC1020(buffer); //Write byte to CC1020
    }

    WriteDataCC1020(0xaa); //Writes last byte and synchronizer

}
/*****/

#endif
```

E.4 Temperature/Humidity SHT11 Sensor Source Code

```

#ifndef __SHT11_H
#define __SHT11_H

unsigned char error, checksum, temp1, temp2; //Define RAM registers
unsigned char humi_val, humi_val1, temp_val;

enum {TEMP,HUMI};

#define TRIS_DATA TRISbits.TRISC1
#define TRIS_SCK TRISbits.TRISC0
#define DATA PORTCbits.RC1
#define SCK PORTCbits.RC0
#define noACK 0
#define ACK 1

//adr command r/w
#define STATUS_REG_W 0x06 //000 0011 0
#define STATUS_REG_R 0x07 //000 0011 1
#define MEASURE_TEMP 0x03 //000 0001 1
#define MEASURE_HUMI 0x05 //000 0010 1
#define RESET 0x1e //000 1111 0

/*****
// writes a byte on the Sensibus and checks the acknowledge
*****/
unsigned char s_write_byte(unsigned char value)
{
    unsigned char i, error=0;
    TRIS_DATA=0; //make DATA-line output
    SCK=0; //initial state

    for ( i=0x80 ; i>0 ; i/=2 ) //shift bit for masking
    {
        if ( i & value )
            DATA=1; //masking value with i , write to SENSI-BUS
        else
            DATA=0;

        SCK=1; //clk for SENSI-BUS
        Nop(); //pulsewidth approx. 2 cycles
        SCK=0;
    }

    TRIS_DATA=1; //make DATA-line input
    Nop();

    SCK=1; //clk #9 for ack
    error=DATA; //check ack (DATA will be pulled down by SHT11)
    SCK=0;

    return error; //error=1 in case of no acknowledge
}

/*****
// reads a byte from the Sensibus and gives an acknowledge in case of "ack=1"
*****/
unsigned char s_read_byte(unsigned char ack)

```

Source Code

```
{
    unsigned char i, val=0;

    TRIS_DATA=1;    //make DATA-line input

    for ( i=0x80 ; i>0 ; i/=2 ) //shift bit for masking
    {
        SCK=1; //clk for SENSI-BUS
        if (DATA)
            val=(val | i); //read bit
        SCK=0;
    }

    TRIS_DATA=0;    //make DATA-line output
    DATA = !ack;   //in case of "ack==1" pull down DATA-Line

    SCK=1;         //clk #9 for ack
    Nop();         //pulsewidth approx. 2 cycles
    SCK=0;

    TRIS_DATA=1;   //make DATA-line input

    return val;
}

/*****
// generates a transmission start
*****/

void    s_transstart(void)
{
    TRIS_DATA=0;    //make DATA-line output

    DATA=1;
    SCK=0; //Initial state

    SCK=1;
    Nop();
    DATA=0;
    Nop();
    SCK=0;
    Nop();
    SCK=1;
    Nop();
    DATA=1;
    Nop();
    SCK=0;
}

/*****
// communication reset: DATA-line=1 and at least 9 SCK cycles followed
// by transstart
*****/

void    s_connectionreset(void)
{
    unsigned char i;
    TRIS_DATA=0;    //make DATA-line output

    DATA=1;
    SCK=0; //Initial state

    for(i=0;i<12;i++) //12 SCK cycles
```

Source Code

```
{
    SCK=1;
    Nop();
    SCK=0;
}

s_transstart(); //transmission start
}

/*****
// resets the sensor by a softreset
*****/

char s_softreset(void)
{
    unsigned char error=0;

    s_connectionreset(); //reset communication

    error += s_write_byte(RESET); //send RESET-command to sensor

    return error; //error=1 in case of no response form the sensor
}

/*****
// reads the status register with checksum (8-bit)
*****/

char s_read_statusreg(char *p_value, char *p_checksum)
{
    char error=0;

    s_transstart(); //transmission start

    error = s_write_byte(STATUS_REG_R); //send command to sensor

    *p_value = s_read_byte(ACK); //read status register (8-bit)
    *p_checksum = s_read_byte(noACK); //read checksum (8-bit)

    return error; //error=1 in case of no response form the sensor
}

/*****
// writes the status register with checksum (8-bit)
*****/

unsigned char s_write_statusreg(unsigned char p_value)
{
    char error=0;

    s_transstart(); //transmission start

    error+=s_write_byte(STATUS_REG_W); //send command to sensor
    error+=s_write_byte(p_value); //send value of status register

    return error; //error>=1 in case of no response from the sensor
}

#endif
```

Appendix F

Battery Discharge Characteristics

The Battery Efficiency is an Empirical Statement, and very difficult to corroborate. It is known that the NiMH has up to 40% greater Theoretical Efficiency than NiCd batteries, but the Energy Density gain is about 20% to 40% higher. Thus, there is a need to briefly analyze the Discharge Performance of the NiMH batteries, for this technology was used to Power the Sensor Nodes. [5]

”Since cell capacity varies inversely with the discharge rate, capacity ratings depend on the discharge rate used. (...) The discharge voltage profile (...) is affected by environmental conditions, notably discharge temperature and discharge rate. However, under most conditions the voltage curve retains the flat plateau desirable for electronics applications.” [5].

F.1 Effect of Temperature on Capacity

Significantly effects on the Discharge Rate Linearity at $\leq 5^{\circ}\text{C}$ are well detailed and documented, and they are a major concern for applications with a very large Temperature Span over the Energy Cycle¹ of the Battery.

”Typically, optimum performance of the nickelmetal hydride battery is obtained between 0°C and 45°C . The performance characteristics of the battery are affected moderately at higher temperatures. At lower discharge temperatures, performance decreases more significantly, caused primarily by the increase in internal resistance.” Fig. F.1 [3]

¹From fully Charged to Fully Discharged

F.2 Effect of Discharge Rate on Capacity

For rates below 0.5C, there are no significant effects on Battery Capacity, but for rates $0.5C \leq xC \leq 4C$, reductions in Cell Voltage may occur, and this results in Capacity Reductions.

”The capacity of the battery decreases more noticeably as the current increases, particularly at lower temperatures.” [3]

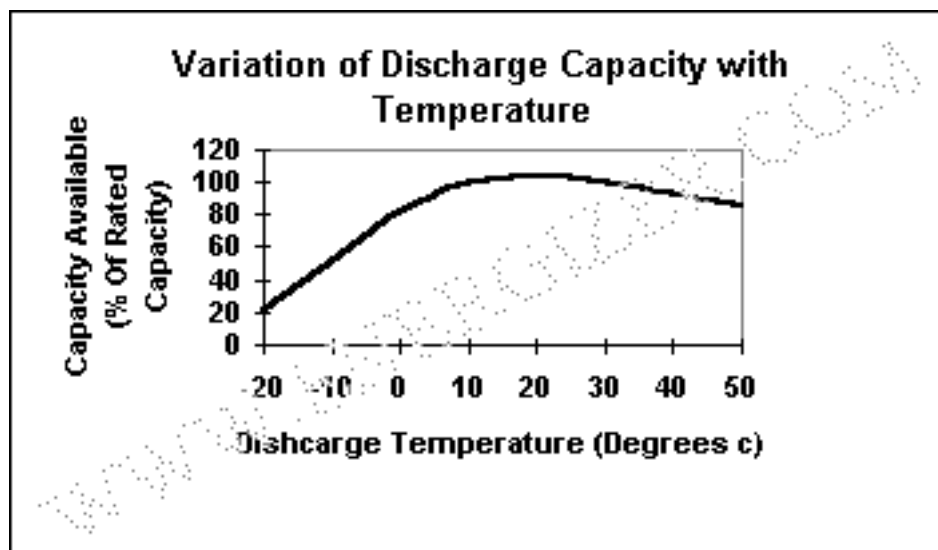


Figure F.1: Variation of Discharge Capacity with Temperature [5]

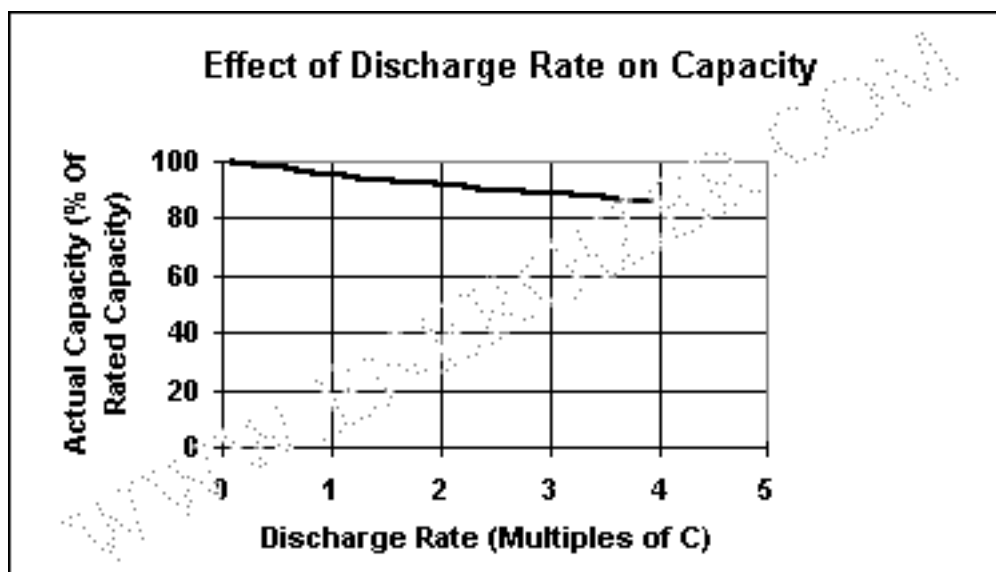


Figure F.2: Effect of Discharge Rate on Capacity [5]

References

- [1] Chipcon: *SmartRF CC1020*. www.chipcon.com.
- [2] Chou, Petrovic, R. A Distributed and Adaptive Signal Processing Approach to Reducing Energy Consumption in Sensor Networks. *University of California at Berkeley* .
- [3] DURACELL: *Ni-MH Technical Bulletin*. www.duracell.com.
- [4] E. Jason Riedy, R. S. Power and Control in Networked Sensors 2000.
- [5] EVEREADY BATTERY COMPANY, I.: *NICKEL-METAL HYDRIDE - Application Manual*. www.energizer.com.
- [6] FTDI: *DLP-USB232M, FT232BM 2nd generation USB UART*. www.ftdichip.com/.
- [7] Ian F. Akyildiz, Weilian Su, Y. S. A Survey on Sensor Networks. *IEEE Communications Magazine* Georgia Institute of Technology, 2002.
- [8] J.L. da Silva Jr., J. Shamberger, M. A. Design Methodology for PicoRadio Networks 2001.
- [9] L. Zhong, J. Rabaey, C. G., and Shah, R. Data Link Layer Design for Wireless Sensor Networks. *Proceedings of IEEE MILCOM 2001, Washington D.C.* :28–31,2001.
- [10] Lizhi Charlie Zhong, Rahul Shah, C. G. An Ultra-Low Power and Distributed Access Protocol for Broadband Wireless Sensor Networks. *IEEE Broadband Wireless Summit, Las Vegas, N.V.* 2001.
- [11] M. Pedram, Q. W. Design Considerations for Battery-Powered Electronics. *Proceedings of the Design Automation Conference* :861–866,1999.
- [12] Maxim: *MAX1722/MAX1723/MAX1724*. www.maxim-ic.com.
- [13] Maxim: *MAX3232, 3.0V to 5.5V, Low-Power True RS-232 Transceivers*. www.maxim-ic.com.

References

- [14] Microchip: *C18 Microchip C Compiler for the 18F series*. www.microchip.com.
- [15] Microchip: *PIC18FXX2 Data Sheet*. www.microchip.com.
- [16] Nordic: *nRF905 - Single chip 433/868/915 MHz Transceiver*. www.nvlsi.com.
- [17] R. C. Shah, J. M. R. Energy Aware Routing for Low Energy Ad Hoc Sensor Networks. *IEEE Wireless Communications and Networking Conf., Orlando, FL 2002*.
- [18] Sensirion: *SHT1x/SHT7x Humidity Temperature Sensor*. www.sensirion.com.
- [19] Wendi Rabiner Heinzelman, Anantha Chandrakasan, H. B. Energy-efficient Communication Protocols for Wireless Microsensor Networks. *Proc. Hawaaiian Int'l Conf. on Systems Science 2000*.

Extra Bibliography

Andrew J. Wixted, Daniel A. James, D. V. T. Low Power Operating System and Wireless Networking for a Real Time Sensor Network. *Griffith University, Australia* .

Carl Falcon, A. S. Adding a Low Data Rate Radio ASSP to an ISM Application. *Microwave Journal* 2003.

J. Hill, R. Szewczyk, A. W. System architecture directions for network sensors. *ASPLOS* 2000.

J. Rabaey, J. Ammer, T. K. PicoRadios for Wireless Sensor Networks: The Next Challenge in Ultra-Low-Power Design. *Proceedings of the International Solid-State Circuits Conference, San Francisco, CA :3–7,2002.*

Jan M, R. Ultra-low Cost and Power Communication and Computation Enables Ambient Intelligence. *sOc* 2003.

Jason Hill, Robert Szewczyk, A. W. System architecture directions for network sensors. *ASPLOS* 2000.

Jason Hill, Philip Bounadonna, D. C. Active Message Communication for Tiny Network Sensors .

Jeong, J., and Ee, C. T. Forward Error Correction in Sensor Networks. *UCB Technical Report* 2003.

M. Kubisch, H. Karl, A. W. Distributed algorithms for transmission power control in wireless sensor networks. *IEEE WCNC 2003, New Orleans, Louisiana :16–20,2003.*

Mainwaring, J. Polastre, R. S. Wireless Sensor Networks for Habitat Monitoring. *ACM International Workshop on Wireless Sensor Networks and Applications* 2002.

Extra Bibliography

Polastre, J.: *Design and Implementation of Wireless Sensor Networks for Habitat Monitoring*. Master's Thesis, University of California, Berkeley. Berkeley, CA., 2003.

Rubin, R. Analysis of Wireless Data Transmission Characteristics. *TinyOS papers* Computer Science Division / University of California, Berkeley, .

Sinha, A., and Chandrakasan, A. Dynamic Power Management in Wireless Sensor Networks. *IEEE Design Test Comp* 2001.

Stojmenovic, I. Position based routing in ad hoc networks. *IEEE Communications Magazine* 40(7):128–134, 2002.

Toh, C.-K. Maximum battery life routing to support ubiquitous mobile computing in wireless ad hoc networks. *IEEE Communications Magazine* :138–147,2001.

Ye, H. An Energy-Efficient MAC Protocol for Wireless Sensor Networks. *Estrin* 2002.

Index

- Base Station Schematic and PCB, 51
- Battery Discharge Characteristics, 86
 - Effect of Discharge Rate on Capacity, 87
 - Effect of Temperature on Capacity, 86
- Bill of Materials, 62
 - Base Station, 62
 - Radio Transceiver, 64
 - Sensor Node, 63
- Conclusions, 48
 - Project Changes, 48
 - Project Future Modifications, 49
 - Project Limitations, 49
- Extra Bibliography, 90
- Introduction, 1
 - Proposed Schemes, 2
- Practical Analysis, 25
 - Base Station, 32
 - Components Used, 32
 - Energy Analysis and Life Expectancy, 35
 - Energy Analysis, 35
 - Life Expectancy, 36
 - Firmware, 41
 - Main Program, 41
 - System Configuration, 41
 - Sensor Nodes, 26
 - Components Used, 26
 - Hardware Behavior, 29
- Practical Analysis Discussion, 44
- Machine Soldering, 44
- PCB Temperature, 46
- PLL Lock, 45
- Practical Analysis Summary, 46
- Receiver Sensitivity, 45
- Project Background, 4
- Relevant Technical Literature and Papers
 - Communication Protocols, 11
 - Efficiency of the System and Battery Capacity, 8
 - Error Correction Modes, 6
 - Power Saving Modes of Operation, 5
 - What is a Wireless Sensor Network and what uses can it have, 4
- Technical Literature and Papers, 4
- Radio Transceiver Schematic and PCB, 59
- Sensor Node Schematic and PCB, 55
- Source Code, 66
 - Base Station Source Code, 70
 - RF Transceiver CC1020 Source Code, 73
 - Sensor Node Source Code, 66
 - SHT11 Sensor Source Code, 83
- Theoretical Analysis, 13
 - Base Station, 23
 - Sensor Nodes, 14
 - Half-Duplex Transmission, 17
 - Polled Transmission, 20
 - Simplex Transmission, 14