TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Rashiduddin Kakar, 165526IVEM

TRANSIENT COMPUTING AND APPROXIMATE COMPUTING ON NON-VOLATILE MICROCONTROLLERS

Master's Thesis

Supervisor:	Yannick Le Moullec
	PhD
Co-supervisor:	Sikandar Khan
	MSc

Tallinn 2019

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Rashiduddin Kakar, 165526IVEM

JUHU- JA LÄHENDARVUTUS PASSIIVSETEL MIKROKONTROLLERITEL

Magistritöö

Juhendaja:	Yannick Le Moullec
	PhD
Kaasjuhendaja:	Sikandar Khan
	MSc

Tallinn 2019

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Rashiduddin Kakar

06/05/2019

Abstract

According to estimates, the future IoT will accommodate around 50 billion nodes by the year 2025. The deployment of billions of these IoT nodes in hardly accessible areas demand for deploy-and-forget type installations where batteries pose a single greatest threat to the vision of a sustainable IoT. Replacing and disposing of billions of batteries is not only impractical and costly in terms of maintenance but is also a severe threat to the resources of our planet. That is why battery-less IoT nodes are considered as the next crucial step towards a sustainable IoT.

Battery-only designs are not an option since their limited lifetimes require expensive maintenance. Energy-Driven Systems (EDS), that are powered from energy harvesting sources, seem to offer an alternative and promising solution for the realization of battery-less IoT nodes. However, since the Harvested Power (HP) from miniaturized harvesting sources is generally low and unstable, an HP-node cannot operate in the same way as a battery-powered system. Hence, it is crucial to optimize the IoT nodes, both from the hardware and software perspectives, so as to efficiently utilize the scarcely available and highly unpredictable harvested energy.

With such an aim to utilize the harvested energy from various ambient energy sources in an efficient manner, this thesis combines the techniques of transient computing, approximate computing and energy/data prediction models so as to reduce the energy consumptions and prolong the life-time of the nodes.

These techniques are applied in a test-bed consisting of two nodes that combine Texas Instruments MSP-EXP430FR5739 kits (FRAM-based micro-controllers) with CC2500 radio module evaluation kits. Building upon the Compute Through Power Loss utility, the various techniques are implemented for different cases, which illustrate their potential in terms of power, memory, and communication efficiency.

Our experimental results show that the accuracy lost due to incorporation of approximations depend on the nature of the data and the number of approximated bits.

In case of data that is represented only in lower order bits, a 1-bit approximation gives a mean percentage error of 11.11%. However, in case of data that can be represented in higher order bits, a 1-bit approximation gives a mean percentage error of 1.06%, and a 2-bits approximation gives a mean percentage error of 6.11%. Our results also show that the number of radio transmissions could be reduced by 10%, 20%, and 30% through the incorporation of approximations.

This thesis is written in English and is 64 pages long, including 5 chapters, 23 figures and 13 tables.

Annotatsioon

Juhu- ja lähendarvutus passiivsetel mikrokontrolleritel

Hinnangute kohaselt sisaldab tuleviku IoT (asjade Internet) aastaks 2025 umbes 50 miljardit sõlme. Nende miljardite IoT-sõlmede kasutuselevõtt raskesti juurdepääsetavates piirkondades nõuab paigalda-unusta (deploy-and-forget) seadmeid, kus akud on suurimaks ohuks jätkusuutliku IoT visioonile. Miljardite akude asendamine ja hävitamine ei ole mitte ainult ebapraktiline ja kulukas, vaid see kujutaks endas ka tõsist ohtu meie keskkonnale. Seepärast peetakse passiivsete IoT-sõlmede arendust järgmiseks oluliseks sammuks jätkusuutliku asjade Interneti suunas.

Ainult akutoitel põhinev disain ei ole vastuvõetav, kuna piiratud kasutusiga nõuab kulukat hooldust. Energiasäästlikud süsteemid (EDS - Energy-Driven Systems), mis töötavad energia lõikuse (energy harvesting) allikatel, näivad pakkuvat alternatiivset ja paljutõotavat lahendust passiivsete IoT-sõlmede realiseerimiseks. Kuna aga väikesest energia lõikuse seadmest pärinev kogutud energiatase (HP-Harvested Power) on üldiselt madal ja ebastabiilne, ei saa HP-sõlm töötada samamoodi nagu akutoitega süsteem. Seega on ülioluline optimeerida IoT-sõlmede riist- ja tarkvara, et tõhusalt ära kasutada raskesti kättesaadavat ja äärmiselt ettearvamatut kogutud energiat.

Käesolev töö ühendab juhu- ja lähendarvutusmeetodeid ning energia ja andmete prognoosimudelite tehnikaid, et vähendada IoT-sõlmede energiatarvet ning pikendada nende eluiga.

Neid meetodeid rakendatakse kahest sõlmest koosneval testplatvormil, mis sisaldab FRAM-põhiseid mikrokontrollereid (Texas Instruments MSP-EXP430FR5739) ja raadio-arendusmoodulit (CC2500). Tuginedes juhuarvutuse võtetele (CPTL - Compute Through Power Loss) rakendatakse mitmeid tõhusaid võimsuse ja mälu kasutamise ning kommunikatsiooni tehnikaid.

Katsetulemused näitavad, et lähendusarvutuste tõttu kaotatud täpsus sõltub andmete iseloomust ja bit-lähenduse sügavusest. Juhul kui andmed on esitatud ainult noorematest bittides, siis 1-bitine lähendus annab keskmiseks veaks 11,11%. Kuid andmete puhul, mis on esitatud vanemates bittides, annab 1-bitine lähendus keskmise vea 1,06% ja 2-bitine lähendus annab keskmiseks veaks 6,11%. Tulemused näitavad, et lähendusvõtteid kasutades saab raadioside aega vähendada 10%, 20% ja 30%.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 64 leheküljel, 5 peatükki, 23 joonist, 13 tabelit.

List of abbreviations and terms

AC	*Approximate Computing*
ADC	*Analog to Digital Converter*
ANN	*Adaptive Neural Network*
API	*Application Program Interface*
ASEA	*Accurate Solar Energy Allocation*
BOR	*Brownout Reset*
CTPL	*Compute Through Power Loss*
CPS	*Cyber Physical Systems*
DC	*Direct Current*
DEBS	*Dynamic Energy Burst Scaling*
DFS	*Dynamic Frequency Scaling*
DP	*Data Prediction*
DRA	*Dynamic Routine Adjustment*
DRAM	*Dynamic Random Access Memory*
EDS	*Energy Driven Systems*
EEPROM	*Electrically Erasable and Programmable Read-Only Memory*
EH	*Energy Harvesting*
EMU	*Energy Management Unit*
ENS	*Energy Neutral Systems*
EPROM	*Electrically Programmable Read-Only Memory*
EWMA	*Exponential Weighted Moving Average*
FPU	*Floating Point Unit*
FPWF	*Fixed Parameter Weighting Factor*
FRAM	*Ferroelectric Random Access Memory*
GND	*Ground*
GPIO	*General Purpose Input Output*
GPR	*General Purpose Register*
GPU	*Graphics Processing Unit*
HP	*Harvested Power*
IoBT	*Internet of Battery-less Things*
IoT	*Internet of Things*
ISR	*Interrupt Service Routine*

LPM	*Low Power Mode*
LSB	*Least Significant Bit*
MOS	*Metal Oxide Semiconductor*
NB-IoT	*Narrowband-Internet of Things*
NVM	*Non Volatile Memory*
NY City	*New York City*
MAE	*Mean Absolute Error*
MRAM	*Magnetoresistive Random Access Memory*
MSE	*Mean Squared Error*
PC	*Program Counter*
PCM	*Pulse Code Modulation*
PNS	*Power Neutral Systems*
PROM	*Programmable Read-Only Memory*
QL-SEP	*Q-Learning based Solar Energy Predictions*
RF	*Radio Frequency*
ROM	*Read-Only Memory*
SP	*Stack Pointer*
SRAM	*Static Random Access Memory*
SVS	*Supply Voltage Supervisor*
TalTech	*Tallinn University of Technology*
TC	*Transient Computing*
TCS	*Transient Computing Systems*
WCMA	*Weather Conditioned Moving Average*
WSN	*Wireless Sensor Network*

Table of contents

List of figures

List of tables

1 Introduction and Motivation

Thanks to rapid advances in the design processes of semiconductor electronics, the power consumption of embedded IoT devices has reduced significantly. However, the demand for long-term deployments and virtually unlimited lifetimes of IoT devices is still at large and the energy consumption of existing IoT devices is considered as one of the most crucial issues in today's IoT and WSN (Wireless Sensor Networks) applications.

Today's battery-powered IoT devices are resource limited; they cannot sustain their power for longer periods of time and need battery recharges and/or replacements so as to sustain themselves for longer periods of time. The power management of such battery-powered IoT nodes is a challenge so as to efficiently utilize the power of batteries for longer periods of time without any need for recharge or replacements. However, battery-only designs do not appear to be sustainable on the long term since their limited lifetimes require expensive maintenance.

As an alternative to battery-powered systems, Energy-Driven Systems (EDS) are powered from ambient energy sources and offer promising solutions for the realization of battery-less IoT nodes [1]. EDS can be used in conjunction with existing battery-powered systems, or they can replace the existing battery-powered systems with stand-alone Energy Harvesting (EH) sources. Replacing the battery with the EH source leads to battery-less or energy autonomous devices also referred to as Internet of Battery-less Things (IoBT).

EH sources transform the energy absorbed from the ambient sources (e.g. solar, thermal, air) in the environment to electrical power, which is then converted to direct current (DC) for powering up the sensor/IoT nodes. EH sources usually produces irregular power due to time-based deviations in the environment such as wind speed, time of the day, availability of light, changes of temperature and weather conditions [2]. This uncertainty in the availability of energy from the ambient sources raises new challenges in the development of reliable energy efficient sensor nodes. For example, solar cell power production variations are not only limited to the 24-hour cycle in the solar energy but also depends on the sensitivity, position and orientation of the cells; similarly, wind power source deviations occur because of the changes in the speed and direction of wind. As per the variations and even temporary unavailability of the EH sources, there can be substantial interruptions in the execution of the IoT end device and will not be able to operate as a battery-powered system where the power is constant for as long as the battery is alive.

Various techniques such as Transient Computing (TC) [2], Approximate Computing (AC) [3], and Energy/Data Prediction models (E/DP) [4] have been proposed in the literature not only to optimize the Energy-driven (battery-less) IoT nodes both from the hardware and software perspectives, but also to cover the intrinsic issues of EH sources. Also from the hardware perspective, Non-Volatile Memories (NVM) based architectures (i.e., Ferroelectric RAM (FRAM) and Magnetoresistive (MRAM)) are currently in use to achieve power and performance efficiency as compared to flash memory based approaches [5].

Flash memory uses bit-cells that need to be erased in order to rewrite a logic 0 or 1 to it and this erase operation requires higher voltage due to which it is more energy consuming and also presents asymmetric read and write operation timings, which complexifies latency in the system. On the other hand, FRAM uses polarization on the ferroelectric capacitor to distinguish between the logic states and does not require erase operation. FRAM also uses destructive read with an instantaneous write operation due to which there is no latency or overhead in the system. Additionally, FRAM has 10 orders of magnitude higher endurance than the approximate endurance of 10^5 erase/write cycles of the flash memory [6].

With a motivation to design an FRAM-based IoT node that could possibly sustain the frequent power losses and variations of ambient energy sources and efficiently utilize the available energy, this thesis combines the techniques of TC and AC along with prediction/DP models to prolong their life time and paves the way towards self-sustainable and maintenance free battery-less IoT nodes.

1.1 Problem Statement

As discussed earlier, improvements at the software and hardware ends are required to gain more energy efficiency. This becomes more challenging due to limited hardware resources of the IoBT devices and the irregular power supply from the ambient energy source.

In this thesis several techniques such as TC, AC and DP are used to increase the lifetime of the IoT end devices. This MSc thesis is part of a research effort conducted at Thomas Johann Seebeck Department of Electronics. In [7], a working prototype combining an FRAM-based node, TC, and EP has been implemented on a Texas Instruments MSP430FR5739 microcontroller and CC2500 radio module. The key idea is illustrated in Figure 1.

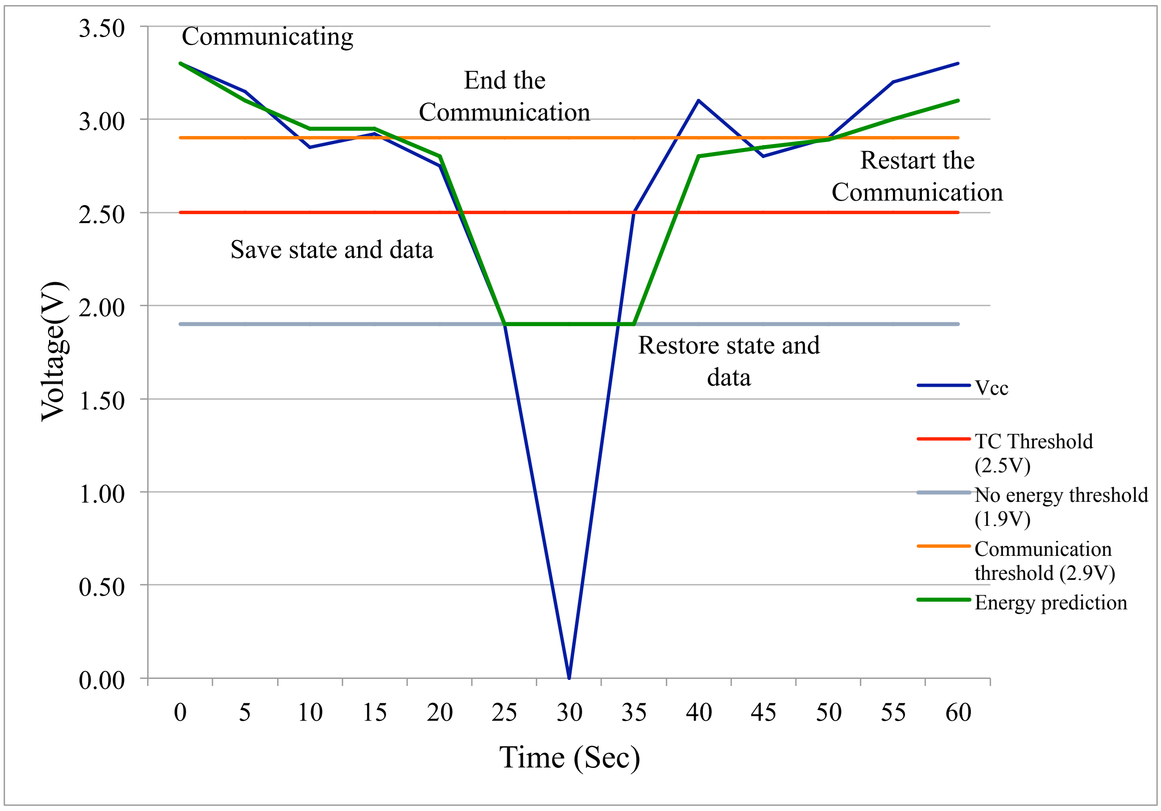


Figure 1. Illustration of the available energy and communication states at different Vcc (voltages). The communication states are decided based on the predicted energy; saving and restoring state and data of the microcontroller is enabled thanks to the FRAM technology and the CTPL library. Figure inspired from [7].

Figure 1 illustrates how an EH source powers a node comprising a Texas Instrument MSP430FR5739 microcontroller combined with a CC2550 RF module. The next energy level is predicted and the node communicates with its peer (which has the same hardware combination) until the voltage prediction level drops to a configurable threshold level (here 2.9 V). Below the 2.9V threshold, the node stops communicating. The node then takes a snapshot (in the non-volatile FRAM) of the current state of the microcontroller and hibernates when V_{CC} further drops to the 2.5V transient computing threshold level. If the voltage level further drops to the so-called no energy threshold level of 1.9 V, the node is not operating. When the energy is back and the voltage rises above 1.9 V, the processor restores the state where it was when TC triggered. Finally, above 2.9 V the node turns on its radio again and proceeds with the communication. EP is used to anticipate significant power losses (to properly terminate the communication between the nodes) and very short power losses (to avoid unnecessary save and restore steps).

The existing setup provides a suitable platform towards battery-less nodes. However, there is room for further improvements; thus, the main purpose of this thesis is to augment the existing setup with techniques that would allow increasing the power, memory, and communication efficiency of the system. As a novelty of this thesis, software based approximate computing (AC) techniques [3] is introduced in the system to increase the amount of information that can be transmitted within the available bandwidth. Moreover, Line-P prediction model, previously used for EP, is now used for DP (in particular detecting if information is changing); this, in turns, allows making decisions regarding whether or not to transfer information.

To achieve the above, the following goals have been defined:

- Analyze the existing system architecture and source code;
- Identify AC opportunities in the existing system and implement them;
- Define and implement a data prediction mechanism;
- Define a transmission control strategy based on data prediction and implement it;
- Analyze the results in terms of power, memory, and communication efficiency.

1.2 Organization of the thesis

The rest of this thesis is organized as follows. Chapter 2 presents the background information on NVMs, EDS, TC, AC and DP. Chapter 3 presents the experimental setup, including the hardware platform and the various software modules used and developed. Chapter 4 presents the results in terms of current consumption, TC, DP, AC, and radio control. Chapter 5 concludes the work and suggest a few perspectives.

2 Background

This chapter aims to study the different techniques to comprehend the range of approaches for EH IoT nodes. It also shows the overview of the state of the art literature of different techniques including the challenges towards the sustainable EH IoT technologies.

As discussed earlier, several techniques such as TC, AC, and EP can be used to reduce the cost and increase the energy efficiency of the IoT devices. In TC, the energy buffer is removed, and the load is directly connected to the energy supply but as the source is subject to variations, it makes the system more unstable, meaning that if the power is not sufficient for the load then it will turn off and when the power is restored again then the system starts back from the beginning. As a solution to this problem, snapshots of the state of the system are taken at periodic checkpoints to NVM; they can restore the system from the same checkpoint at a later time when the energy will be available [8]. However, inflexible, periodic checkpoints in the system creates an overhead. In order to remove the overhead created by the unused checkpoints, EP can be used to predict the energy using historical data and thus reducing the number of unnecessary checkpoints. Moreover AC can used to reduce the energy consumed for the communication between the nodes and introduce error tolerant systems.

2.1 Non-Volatile Memories

NVMs are a type of memories that retains data after losing power. In computers, a Read-Only Memory (ROM) is such a NVM, but as the name suggests, it is only readable, and the data can be written only once. In embedded systems, the embedded code and data records can be stored in the ROM. ROM can be categorized into mask read only memories, which are PROM (Programmable Read-Only Memory), EPROM (Electrically Programmable Read-Only Memory), and EEPROM (Electrically Erasable and Programmable Read-Only Memory).

The other main type of memory is volatile memory, the most common one being called Random Access Memory (RAM) and as the name suggests, RAM loses its data if the power is turned off. RAM can perform both read and write operations and can be further divided into two sub categories called SRAM (Static RAM) and DRAM (Dynamic RAM).

The main difference between SRAM and DRAM is that after each write operation, SRAM keeps data as long as power is available, while the data needs to be refreshed (i.e. overwritten) in DRAM.

2.1.1 Available NVMs

There are different types of NVMs but in most of them, an NVM cell is based on a MOS (Metal-Oxide Semiconductor) transistor having a source, drain, control gate and a floating gate. Charges are stored and retained in the floating gate when the power is removed. All floating gate memories have same cell structure [9].

Due to the data retaining strength of NVMs, they are used in most of the consumer products. Flash memory is the mostly used NVM; flash memory uses a memory cell allowing high reliability and low power consumption. As NVM is developing, new technologies and types of NVMs are emerging day by day.

2.1.1.1 Flash memory

In 1980, Toshiba's then factory manager Dr Fujjo Masuoka invented Flash memory [9] and proposed it for the first time in 1984 [10]. It is commonly used in computers and other electronic devices for the storage of important data that needs to be saved if the power is removed. Flash memory has the advantage of removing the unwanted clusters of data without removing the memory chip.

Flash memories uses large memory cells which are grouped in blocks comprising of memory arrays. Charging the floating gate in the transistor can program these blocks.

2.1.1.2 FeRAM/FRAM (Ferro electric Random-Access Memory)

FRAM is another kind of widely used NVM as they preserve data even in the absence of power supply signal. A ferroelectric capacitor is used as a storage element in each memory cell that stores a logic state based upon electric polarization of the capacitor. The non-volatility nature of the FRAM comes due to using ferroelectric material as the dielectric of the cell's capacitor. The memory cell is composed of ferroelectric capacitor and an access transistor, which stores logical data '0' or '1' depending on the electrical polarization state of the ferroelectric capacitor. Ferroelectricity is a physical property where a spontaneous polarization of the electric dipoles is generated by applying external voltage to electric dipoles arranged in the ferroelectric material [9].

FRAMs are generally classified into two types; one type operates by detecting a change in charge amount stored in the ferroelectric capacitor and the second type operates by detecting a change in the resistance of a semiconductor due to spontaneous polarization of the ferroelectric material.

The applied voltage across the ferroelectric capacitor polarizes the ferroelectric material according to the direction of an electric field. FRAM features are high speed, low power consumption and high rewriting. A general comparison of the common RAM types is given in Table 1.

Table 1. Comparison of FRAM with other memory products. Table inspired by [9].

Memory Products	***1 SRAM**	***2 DRAM**	***3 EEPROM**	***4 FLASH**	***5 FeRAM**
Memory type	Volatile backup	Volatile	Non-volatile	Non-volatile	Non-volatile
Read cycle (ns)	12	70	200	70	110
Internal write voltage (V)	3.3	3.3	20 (supply voltage 3.3V)	12 (supply voltage 3.3V)	3.3
Write cycle	12 ns	70 ns	3 ms	1 s	110 ns
Data write	Overwrite	Overwrite	Erase + Write	Erase + Write	Overwrite
Data erase	Unnecessary	Unnecessary	Byte (64 byte page)	Sector (8K / 16K /32K /64K)	Unnecessary
Endurance (# of rewrites)	Infinite	Infinite	1E5	1E5	1E10 to 1E12
Stand-by current (uA)	7	1000	20	5	5
Read operation current (mA)	40	80	5	12	4
Write operation current (mA)	40	80	8	35	4[1]

[1] Note *1: 512K x 8bit, *2: 2M x 8bit, *3: 8K x 8bit, 4: 1M x 8bit, *5: 8K x 8bit

As shown in Table 1, all the other available memories have drawbacks: for instance, DRAM and SRAM are fast but volatile, also SRAM are comprised of large cell sizes. EEPROM and Flash are non-volatile but slow and have lower write speeds. On the other hand, FRAM is faster and consumes less energy than the other available and low-cost memories, which makes it more suitable for low power applications as targeted in this work.

2.1.2 Emerging Non-Volatile Memories

Due to the limitations of the aforementioned memories, there is a dire need for high speed, high write endurances and small size. Some of the emerging NVMs are Magnetoresistive Random Access Memory (MRAM), Conductive Bridge Random Access Memory (CBRAM), Phase Change Random Access Memory (PRAM), Silicon-Oxide-Nitride-Oxide-Silicon (SONOS), Resistive Random-Access Memory (RRAM), Racetrack memory, Nano Random Access Memory (NRAM). These memories are under research to make it more feasible to implement them in IC technologies. MRAM is non-volatile, fast, consume very little power and are not prone to write cycle limitations. MRAM uses magnetic orientations to retain data in its cells. Nowadays MRAM is also available on the market; as shown in Table 2, it has higher speed than FeRAM but due to higher price it is not considered feasible for low cost, low power application devices. Table 2 shows the differences and comparison of the available memories in the industry.

Table 2 Comparison of available NVMs. Table inspired by [9].

	MRAM	SRAM	DRAM	Flash	FeRAM
Read Speed	Fast	Fastest	Medium	Fast	Fast
Write Speed	Fast	Fastest	Medium	Low	Medium
Array Efficiency	Med/High	High	High	Med/Low	Medium
Future Scalability	Good	Good	Limited	Limited	Limited
Cell Density	Med/High	Low	High	Medium	Medium
Non-Volatility	Yes	No	No	Yes	Yes
Endurance	Infinite	Infinite	Infinite	Limited	Limited
Low Voltage	Yes	Yes	Limited	Limited	Limited
Complexity	Medium	Low	Medium	Medium	Medium

Please note that this MSc thesis does not seek to gain a deep understanding of these new memories but rather presents an overview; the work presented later is based on already commercially available FRAM-based microcontroller from Texas Instrument.

2.2 Energy Driven Systems

EDSs can be categorized broadly into the following classes and subclasses.

2.2.1 Transient Computing Systems (TCS)

The word transient means lasting only for a short time and computing means to calculate or compute; so, combining both words gives us transient computing that means temporary or non-lasting computations. In the computing world, TCS is a generic term and refers to all those systems that are capable of providing correct operation despite experiencing frequent power losses [1] and is often used in combination with EH and NVM.

To remove the hustle of charging or replacing the battery, interest is growing for powering the IoT devices from the ambient sources available in the surroundings. As discussed earlier, EH techniques have their challenges due to constrained and irregular supply and this irregular supply is the due to the variations in the surroundings. However, as a traditional approach, this issue can be resolved by adding an energy buffer such as super-capacitor between the source and the load and provides stable supply to the system. The disadvantage associated to the energy buffer is that it adds additional cost, weight and runtime for its charging and discharging [8].

As a solution to the aforementioned problem, TC can be used to directly power the system with the EH method and saves a snapshot of the microcontroller state when the system senses instability in the power. The system usually takes snapshot of the system after certain intervals of time and if the node dies due to power loss, it retains the snapshot and once the power is available, and the system resumes from the previous saved checkpoint.

Scientists and researchers have developed several techniques in the recent years. Works done on TC have not been limited to the implementation processes but also includes the combination of other techniques, such as EP and AC. Several types of TC techniques have been proposed so far which are used to optimize the available energy and take a snapshot of the system at certain stage before the power loss so it can be retained later at the availability of power source. TC techniques can be classified into generic TC systems, Power Neutral Systems (PNS) and Energy Neutral Systems (ENS).

Power Neutral Systems (PNS): Directly coupled systems that adjust their performance and adapt themselves to the harvested power are called Power Neutral Systems (PNS). In Figure 2, the presence of a harvesting-aware computational load brings power-neutrality into the system where a power conversion unit may or may not be present. Systems using Dynamic Routine Adjustment (DRA), Dynamic Frequency Scaling (DFS), or a combination of both, are examples of PNS [1].

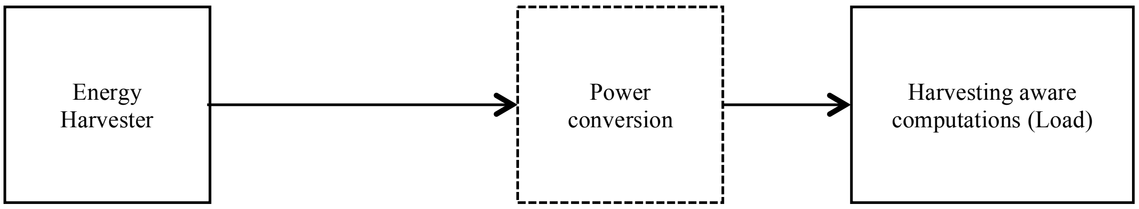


Figure 2. PNS's energy subsystem architecture in TC. Figure inspired by [1].

Energy Neutral Systems (ENS): These systems add intermediate components such as power regulators, power conversions and energy storage between the harvester and the load to make it look like a battery to the load. The added components increase the complexity, cost, mass and volume of the system. All Energy-Management-Unit (EMU) based systems are ENS [1], as shown in Figure 3.

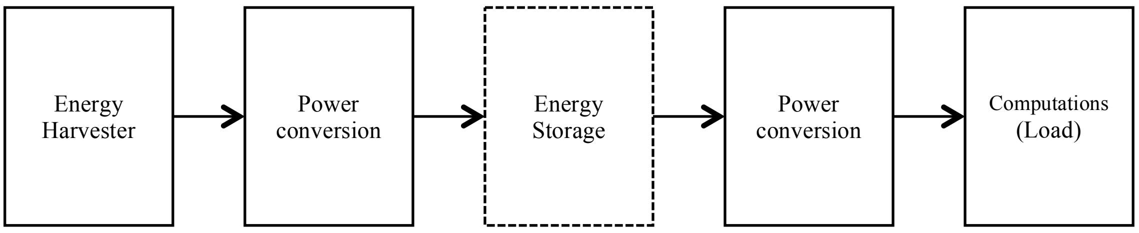


Figure 3. ENS's energy subsystem architecture in TC. Figure inspired by [1]

These techniques are summarized in Table 3 and generic TC techniques are explained further.

Table 3. Classification of recent studies in EH powered IoT nodes

Type	Notable work	Platform	Main Idea	Advantages	Disadvantages	Contribution to further development
Generic	**Mementos**[11]	Software	- Check-pointing at strategic points. Interruptible - Computations for running over EDS. - A step toward battery-less computations.	- Improving the reliability of EH nodes - Adding flexibility. - Increasing the integrity of EH- nodes	- No energy consumption evaluation - High overhead in energy and storage - Manual entry of Trigger points	- Prototype - Technique
	Hibernus [12]	Hardware	- One snapshot before power failure. - Switching between Active and Hibernate state	- Simplicity to TCS - No manual placement of trigger points	- No energy consumption evaluation - Frequent capturing of snapshots -Manual input of threshold voltage for Active and Hibernate state	- Technique - Platform
	Compute through power loss (CTPL) utility by TI [13]	Hardware	- Hibernus like system	- Open source - Simple - Available	- No energy consumption evaluation -Only three levels available - Only works with MSP430FR microcontrollers	- Prototype - Platform
	QuickRecall [6]	Hardware	- Using unified memory system. - FRAM is used as RAM while	- High throughput for taking snapshot	- Slower due to usage of FRAM - No usage of	- Technique - Platform

			RAM is not used.	and restore.	RAM - Usage of inflexible fixed voltage threshold.	
	Hypnos [14] **(Not complete TC)**	Hardware	-Uses ultra low power mode. - No NVM is used and instead SRAM and super-capacitor is used - Extreme voltage scaling	- Ultra low power sleep mode. - SRAM is used	- Loss of data if no power is available for longer period	- Prototype - Technique
Power Neutral Systems (PNS)	**Dynamic Frequency Scaling (DFS)** [15]	Hardware	-Change the frequency of the microcontroller with respect to the available power	- High throughput - High productivity - Increased execution time	- Only compatible with processors having frequency scaling capability	- Prototype - Platform
	Dynamic Routine Adjustment (DRA) [16]	Software	- Modulating the sleep and wake up routine of the microcontroller with respect the available power	- High throughput - High charging time of the capacitor	- Increased overhead - Increased sleep times	- Technique
	Hibernus ++ [17]	Software	- Adaptive system to take the snapshot only once before the power failure. - Characterise the system properties and behaviour - Intelligently adapts hibernate and restore thresholds.	- Autonomous set up of thresholds for hibernate and restore. - Low snapshot overhead	- Higher computation overhead than Hibernus	- Prototype - Technique

	Dynamic Tasks Scheduling (Enhanced Hibernus++) [18]	Software	- Scheduling tasks based on the available power	- Reducing communication workload	- Only simulations available - Higher overhead than Hibernus++	- Technique
	ENZYME [19]	Software	- Frequency modulation and routine adjustment	- High throughput - High productivity	Combined overhead of DRA and DFS.	- Technique
Energy Neutral Systems (ENS)	**Energy Management Unit (EMU) based EDS** [20]	Hardware	-Useful when tasks operate at different voltage/energy levels. -Execution continues in parts	-Bridging the gap between operating levels of source/load	-Much efforts on the design of EMU rather than original problem -No evaluation of the EMU based energy consumptions	-Prototype -Architecture -Platform
	Dynamic Energy Burst Scaling (DEBS) [21]	Hardware	-Execution of program in chunks of tiny tasks	-Useful when tasks operate at different voltage/energy levels. -Execution continues in parts	-Hardware overhead -Sustainability issues -No evaluation of the EMU based energy consumptions	-Platform - Architecture
	EMU+DEBS+EDS [22]	Hardware/Software	-Speed estimation application based on an EMU based EDS	-A reliable platform -Battery less System -No maintenance -Green application	-Accuracy is power dependent -Lack of energy models	- Application

The most “fundamental” techniques are explained further in what follows.

2.2.1.1 Mementos

Mementos is the oldest and basic technique of TC in which checkpoints at strategic points are taken during execution, meaning that it regularly saves snapshots of the system state to the NVM that allows it to return to a previous checkpoint after power failure. Checkpoints are taken at strategic points, but periodically, which is considered as the drawback for the technique, as it increases time and overhead due to unnecessary checkpoints.

This technique has two parts, one is a set of program transformation passes and the other one is a so-called compact library. The former is used to insert energy-measurement code at control points and the latter provides state check pointing and recovery functions. It can be integrated to the application using standard means. The goal of the Mementos design is to automatically resume and suspend the program without external intervention. The main principles of Mementos includes that it can be developed in any NVM hardware (although it has been evaluated using only Flash memory for experiments) and does not require any special hardware except voltage measurement in order to measure the voltage of the energy buffer. Voltage measurement circuitry is generally available on all devices operated on energy storage devices.

The next principle is to argue about energy maximally during the run-time and minimally during the compile time. Reasoning about run-time energy availability at compile time was impossible due to variations in the energy sources. So, Mementos designs estimates energy at run time and insert energy checks at compile time.

Mementos places trigger points and estimates the available energy, then it covers the *main()* function of the program with a code that restores execution from an available checkpoint.

Mementos offers three different instrumentation strategies enabling it to instrument common structures to be compatible with any program. The following modes are used in order to support the instrumentation strategies.

- Loop-latch mode: Loop-latch is a back edge that runs from the bottom to the top of the loop. Trigger points are placed at each loop-latch and check the input voltage level for each iteration of every loop in the program.

- Function-return mode: Trigger points are inserted after each function call, where Mementos checks the input voltage level each time the program returns from a function.
- Timer-aided mode: This mode is used to reduce the frequency of check-pointing operations. This mode works with either the loop-latch or function-return mode. A hardware timer interrupt is inserted to set a flag at predefined intervals. Each trigger point checks the flag, if the flag is set then it proceeds for checking the voltage level. This mode is used to reduce excessive check-pointing and save energy. This energy saving make sure less energy is used in checking the voltage level that the run-time execution.

A programmer can insert trigger points manually, instead of using the above mentioned instrumentation strategies, simply by including the header of the API provided by Mementos and placing the function calls without checking the input voltage level. An ADC is used to compare the input voltage against a predefined threshold [11]. If the input voltage is deemed to be failing (Vcc < threshold V_{TH}), a snapshot of the system is saved to NVM. Regular polling of Vcc is required for this purpose that results in multiple snapshots being saved if Vcc fluctuates near the threshold and increases the time overhead of the performance of the system [23]. The frequency of the trigger points enables the ability of Mementos to precisely complete a checkpoint.

The drawback of this technique is that it uses (slow) flash memory and has a very high overhead due to excessive check-pointing [11].

2.2.1.2 Hibernus

Hibernus is the second technique proposed for TC. It is a refinement of Mementos. This approach saves the snapshot of the complete system to the NVM only once immediately before the system power failure. This technique uses FeRAM, as it is faster than Flash memory and easily integrated into low power application. FeRAM is exploited upon a power failure: it is used to store the microcontroller's state snapshot by using the energy left in the microcontroller's decoupling capacitance. Contrary to Mementos, Hibernus does not use any trigger points and has two states, i.e. active and hibernating states. Active state is when the input voltage is higher than the restore voltage (V_R) and hibernating state is that when the threshold voltage (V_H) is higher than the input voltage.

In the original paper [23], Hibernus is implemented on a Texas Instruments FRAM-based MSP430 microcontroller and the internal comparator of the microcontroller is used to continuously check the input voltage and sends hardware interrupt if it crosses either hibernate or restore thresholds. Inside the interrupt service routine, a function is called to store the snapshot of the volatile memory and the registers into the non-volatile FeRAM, the check-pointing is set, and the system will enter into deep sleep mode. The energy stored in the decoupling capacitance of the microcontroller is used to save the snapshot of the system before going to deep sleep. This allows it to have low threshold voltage (V_H), resulting in increased active period of the main program. While determining V_H, the time needed to charge the decoupling capacitor is considered in order to have enough energy for saving the snapshot before a complete power loss. The flowchart shown in Figure 4 illustrates the Hibernus approach.

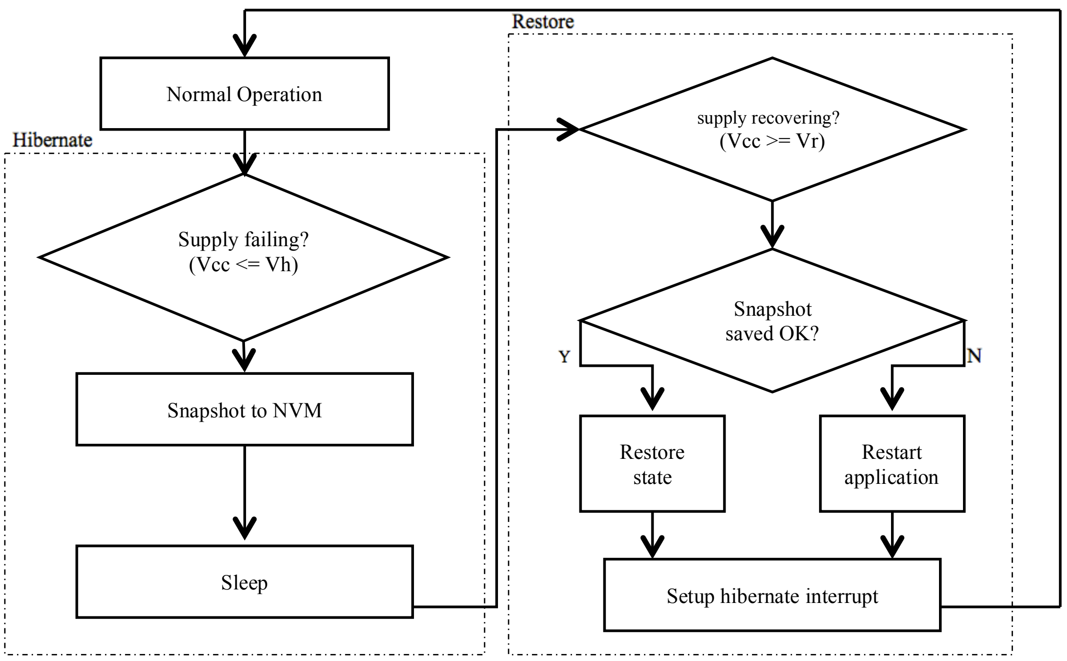


Figure 4. Flowchart illustrating the principle of Hibernus. Flowchart inspired by [23]. Hibernus stops the program normal operation if the supply is lower than Vh and goes to sleep after taking a snapshot. Once power is again available, it checks the snapshot if taken successfully and restore the state.

Hibernus avoids excessive check-pointing and as a result, this technique is more energy and time efficient. This technique can make the system hibernate at any point during the execution of an application and is transparent to the programmer. The content of all registers and volatile memory are copied to the NVM in order to save a snapshot. The energy required, E_σ, to save a snapshot is calculated using Equation 1.

$$E_\sigma = n_\alpha E_\alpha + n_\beta E_\beta \quad [24] \qquad (1)$$

where n_α is the size of the RAM in bytes and n_β is the number of bytes used by the registers, E_α and E_β are the required energies for copying each RAM and register byte to the NVM, respectively. The microcontroller active voltage range lies between V_{min} and V_{max}. Given the total capacitance $\sum C$, Equation 2 can be used to determine the energy E_δ stored in the microcontroller's decoupling capacitor between a given voltage V and V_{min} [24].

$E_\delta = ((V^2 - V_{min}^2)/2). \sum C$ [24] (2)

After determining the stored energy in the decoupling capacitor, the threshold voltage (V_H) can be determined in order to ensure that the snapshot is saved completely.

According to the authors of [12], the capacitance of the microcontroller is 16μF, the size of RAM in bytes is 1024, and the size of registers is 512 bytes. Moreover, E_α is 4.2 nJ and E_β is 2.7 nJ in case of FeRAM. Substitution of the given data in Equation 1 gives that 5.7 μJ (E_σ) is consumed to store a snapshot of the system. To make sure that a complete snapshot is taken then the expression $E_\sigma \leq E_\delta$ needs to be true. As the MSP430FR5739 works from V_{min} = 1.9 V to V_{max} = 3.6 V, using $E_\sigma = E_\delta$ and using Equation 2 gives the voltage as 2.17 V; to allow $V_{cc,}$ an hysteresis is added by setting V_R higher than 2.17 V.

2.2.1.3 QuickRecall

The next TC technique used as a solution to the problems of EH is QuickRecall [2]. This technique allows the FRAM to be utilized as RAM and thus the system works as a "unified memory system", i.e. the RAM itself is not used and only FeRAM is used as a unified memory.

In order to take a snapshot, the program needs to save the states of the processor and program to the NVM before a power failure. Previously, checkpointing was introduced and the application checked the input voltage level either periodically or at desired locations, which delays program execution and introduces additional overhead to the application. QuickRecall is designed in such a way that a checkpoint is triggered if there is a drop in the input voltage, meaning that the checkpoint is taken only if there is a forthcoming power loss; this results in the removal of the delay in the normal execution of program due to checkpointing. It is crucial for this kind of technique that the checkpointing needs to be successfully completed before the complete power loss. For this, QuickRecall chooses an appropriate trigger voltage (V_{trig}) to interrupt the program and start the checkpointing.

To check the input voltage level, the microcontroller's General Purpose Input Output (GPIO) pins are connected to an external comparator, which is then configured with a predetermined trigger voltage (V_{trig}). The comparator checks the input voltage (V_{cc}) and compares it with the trigger voltage (V_{trig}). If V_{cc} is smaller than $V_{trig,}$ it sends the signal output to the microcontroller [25].

To take a complete snapshot, the system needs to store the state of the program, processor and configuration of the registers of various peripherals. All the mentioned states need to be retained and recalled as a function of the availability of power. The system needs to checkpoint periodically and store the contents of the RAM and the registers, which introduces checkpointing overhead. By definition, checkpointing overhead is the time needed to the snapshot of the system before a power-loss and wake-up overhead is the time taken by the system to restore on the availability of power. According to [25], the checkpointing and wake-up overhead of the techniques presented in the previous subsections are higher as compared to that of QuickRecall.

MSP430FR5739 has internal non-programmable supply voltage supervisor (SVS) that monitors V_{cc} using comparator and its output proctors the program execution window [25]. For the MSP430FR5739, the value of V_{trig} is required to be higher than values for SVS_{off} and SVS_{on}. The typical values for SVS_{off} and SVS_{on} are 1.88V and 1.93V, respectively. Moreover, 2.0V is required for the safe operation of FeRAM. The capacitor discharge equation and the storage time enabled the calculation of $V_{trig.}$ V_{trig} was calculated as 2.0003V for the successful completion of checkpoint [25].

In the software, after taking the snapshot, a flag is set to check later if a checkpoint is stored or not. During the booting process, the program checks the flag. If the flag is set, then it initialises all the peripherals and checks if V_{cc} is higher than V_{trig}, then the system restores the core registers, clears the flag and executes the main program. The flowchart shown in Figure 5 shows the software structure of this technique [25].

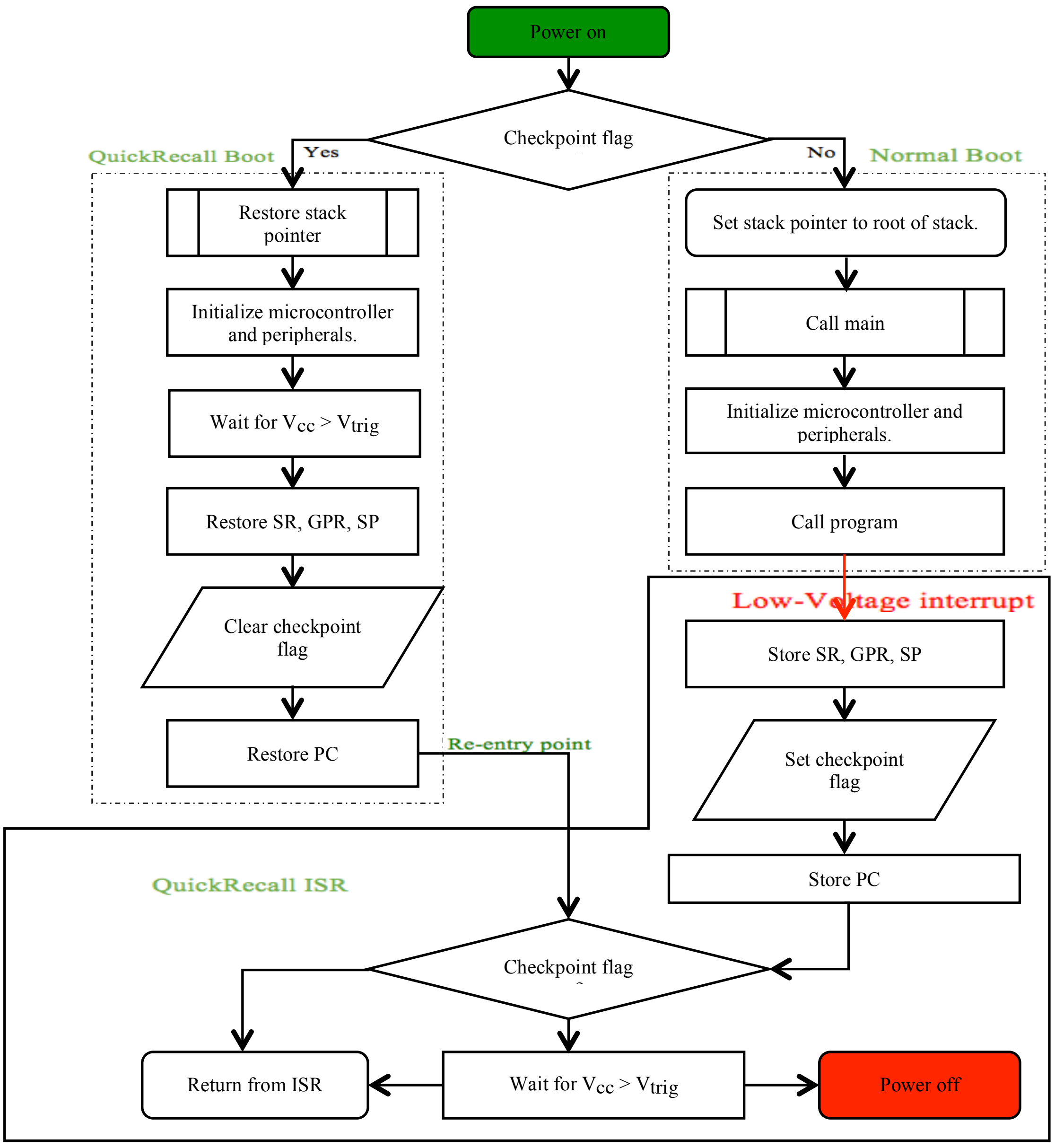


Figure 5. Flowchart illustrating the software flow of QuickRecall. Flowchart inspired by [25]. After the start-up, QuickRecall checks the checkpoint flag and if the flag was set, it boots the processor and initializes the microcontroller. It waits for Vcc to go above Vtrig and then restores SR (Stack Register), GPR (General Purpose Register), SP (Stack Pointer) and if the flag is not set then the system goes through normal boot.

A disadvantage of QuickRecall is that it relies on the processor for the use of FeRAM as a unified memory and by doing so, it introduces significant timing overhead and consumes more energy in write operation than with SRAM. Moreover, it uses a fixed and inflexible threshold voltage (V_H) to take the snapshot. An additional overhead varies depending on the application due to initialization of the microcontroller and the peripherals [26].

2.2.1.4 Hibernus++

After considering the drawbacks of the previous techniques, a new technique was proposed to increase the efficiency of the system. The main goal of this technique was to reduce the overhead and increase the accuracy of system. Hibernus++ is the advanced version of Hibernus [17]. In particular, it has the ability to self calibrate and adapt the threshold and restore voltages in response to the load properties and source dynamics. In this technique the system characterises the hardware in order to set the threshold voltage (V_H). This characterisation is done to increase the active time and reduce energy wastage.

The flowchart shown in Figure 6 shows the working mechanism of the Hibernus++ technique.

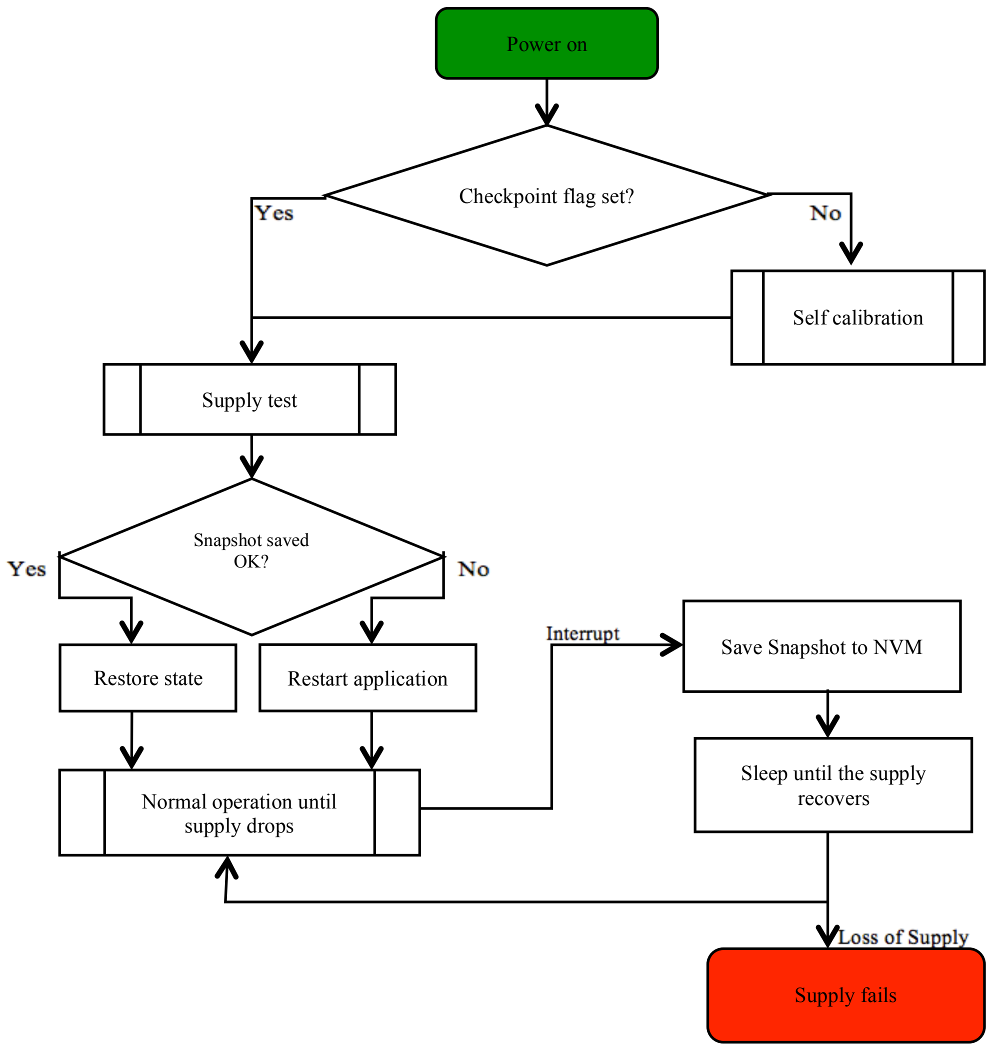


Figure 6. Working mechanism of the Hibernus++. Figure inspired from [17]. Hibernus++ self calibrates to find the threshold voltage if a snapshot was not saved previously, otherwise it restores the previous state.

When power is available, the system checks if it was calibrated or not. If the system was not calibrated previously, then it evaluates the rate of voltage drop in case of sudden power loss by running the calibration routine. During this routine, the system sets the threshold voltage (V_H) to sleep. The system then checks the input power and tests if the power is sufficient for the sustainability of the system in active mode. If the input power is not sufficient then the system goes to sleep until the voltage reaches a sustainable value. The system checks if there was a previous attempt to take a snapshot; if the previous attempt failed, then it increases V_H by 0.1 V and starts the execution from the beginning. If the system snapshot was taken successfully, then the system restores and continues normal operation until the input voltage drops below the threshold voltage (V_H). If the input voltage goes below the threshold voltage then the system stores the snapshot and goes to sleep. If the input voltage recovers before going below the operating voltage of the microcontroller (V_{min}), and the contents of the volatile memory are not lost, then the system resumes its operation without the need to restore the data from the snapshot, otherwise the system restores from the previous successful snapshot.

As discussed earlier, check-pointing and polling increases the overhead. To reduce the overhead caused by checkpointing, Hibernus++ uses an adaptive approach, i.e. the system stores a snapshot only if the power loss is imminent. This approach involves the risk of losing all the data if a snapshot is not successful and the system needs to start computation from the beginning.

Hibernus++ uses self-calibration routine to determine the threshold voltage at run-time in order to enable the system to take a snapshot using the energy stored in the decoupling capacitance of the microcontroller. Similar to Hibernus, an interrupt is configured to save a snapshot if the voltage goes below the threshold voltage.

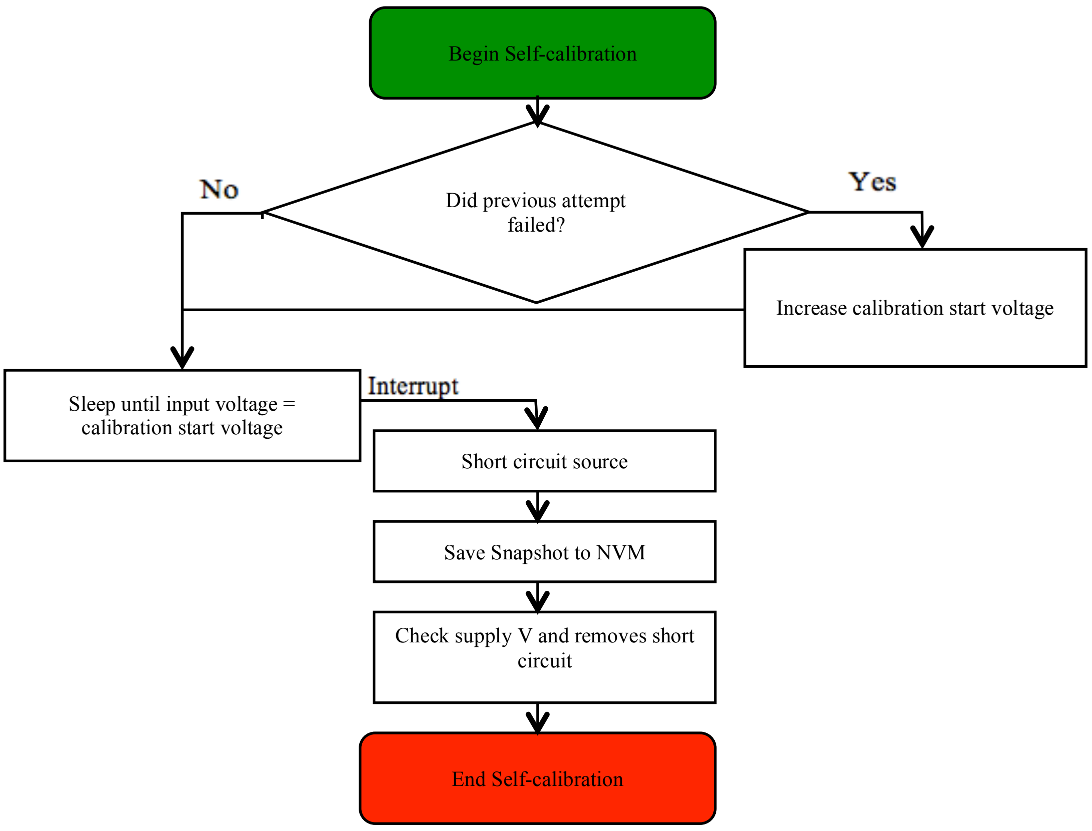

Begin Self-calibration
Did previous attempt failed?
No
Yes
Increase calibration start voltage
Sleep until input voltage = calibration start voltage
Interrupt
Short circuit source
Save Snapshot to NVM
Check supply V and removes short circuit
End Self-calibration

Figure 7. Self-calibration of the threshold voltage V_H. Flowchart inspired by [17]. Hibernus ++ begins self-calibrating during the start up and tries to save the snapshot if Vcc is at the initial level. If the attempt fails, the program increases the threshold voltage until a snapshot is taken successfully.

The flowchart shown in Figure 7 shows the process of determining VH using self-calibration routine. As shown, the system waits for the input voltage to reach the calibration voltage. Once both voltages are matched then the system disconnects the source using a switch and saves a snapshot the FeRAM. Then the system checks the input voltage and reconnects the source and closes the self-calibration routine.

2.2.1.5 Compute Through Power Loss (CTPL)

Texas Instruments (TI) provides the Compute Trough Power Loss (CTPL) software utility library as FRAM utilities for its low power FRAM-based MSP430FRXXXX microcontroller series. CTPL is a Hibernus-like approach and uses voltage thresholds to save the state of the CPU and its peripherals to the FRAM NVM before the loss of power. Once power is back, it restores the CPU and its peripherals and the program resumes from the point where it executed last. Comparator D of the microcontroller is used to trigger the interrupt in the program. This library can be integrated into the program and sometimes avoids the intensive start-up routine when it starts from the resume state.

CTPL stores the CPU states, key peripherals and stack from RAM to FRAM. After the availability of power, CTPL checks if an image was taken successfully and the program shall resume from the image or execute the C start-up routine. The C start-up routine is the C compiler runtime library which executes prior to the *main()* function. If the program needs to execute from the image, then the state of the CPU, key peripherals and stack are restored, which avoids the C start-up routine re-initialisation and saves significant amount of energy. The flowchart of CTPL is shown in Figure 8.

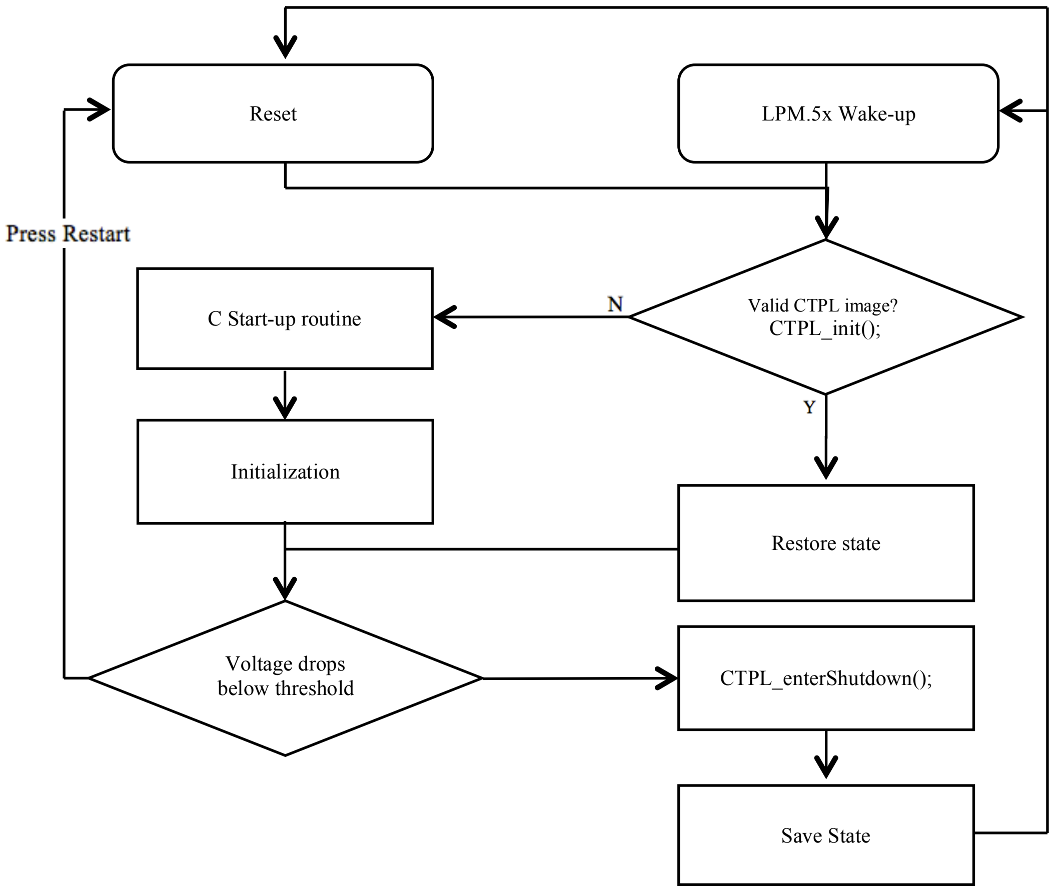


Figure 8. Working mechanism of CTPL. Figure inspired by [13]. CTPL checks if a snapshot was taken and then restores the state; if a snapshot was not taken then it will go through initialisation process and once the voltage drops below threshold then it takes the snapshot

The voltage thresholds for saving the state of the program are configurable. This can be chosen based on the system requirement of the application to provide enough energy for safe shutdown. The voltage measurement ability of the ADC10_B is used to measure V_{CC} with a sampling rate of 1 kHz. Also, the built-in comparator in the MSP430FRXXXX can be used with an external voltage divider to provide a reference voltage on Pin P1.5 and trigger the CTPL in case the voltage drops below the threshold. When V_{CC} goes below the threshold level, it saves the state and enters the low-power mode “LPM.5x”. After entering LPM.5x, it waits for the device to enter a Brownout reset (BOR) if there is loss of power [27].

The main disadvantage of CTPL is that it is designed for TI MSP430 FRAM-based microcontrollers and is not compatible with other platforms [27]. Moreover, its weakness is that if the voltage decrease rate is higher than 4.8 V/s, then the system will not detect the power loss and the snapshot will not be taken [7].

2.3 Energy Prediction

Unlike battery operated IoT systems, it is impossible for a battery-less system to quantify the amount of available energy at a given time. As of today’s research, the energy consumption of the radio module in a node is higher than other operations including sensing or computing. EP of the EH is new paradigm to in order to control the radio or take other decisions based on the predicted energy. Researchers have proposed several EP models in the recent years for EH [4]. These models depend on different parameters such as time of the day, weather and historical data. One of the EP models proposed in our department is LINE-P EP model.

2.3.1 LINE-P EP Model

Line-P is a lightweight and near to accurate energy prediction model. This model has three classes and uses results of approximation and sampling theory [4]. Line-P takes into account the smooth variations and also the rapid fluctuations of the historic data and then predicts the short term and long term energy. In this model, sampling operators in Equation (26) of [4] are used to define three predictors.

According to [4] LINE-P has three cases which are divided based on different levels of memory overheads and accuracy. It has been claimed by the author of [4] that LINE-P predictions are more accurate if the fluctuations on the source are smoother.

The first and third cases of LINE-P use energy information from one of the previous days and take the previous samples from the same day to predict the next value. However, unlike other cases and models, the second case works only with the previous samples from the same day. This case is beneficial if the data from previous days are not available. As this case does not require the data from the previous days, it is more memory efficient than the other prediction cases or models. However, according to [4], the Mean Square Error (MSE) and Mean Average Error (MAE) of Case II is higher than the other two cases of the same model.

The equation that represents class II of Line-P EP model is as is as per Equation 3.

$$(S_{PREDII;a}f)\ (j) = \sum_{k=1}^{m} a_k\, f(j-k)\ [4] \tag{3}$$

Where j is the next time slot, m is the number of historic slots and a_k is the coefficient and its values are derived in [4] as per (4),

$$a = \left\{0,0,0,0,0,0,0,\frac{3}{8},\frac{15}{16},\frac{1}{2},-\frac{3}{8},-\frac{3}{8},-\frac{1}{16}\right\}\ [4] \tag{4}$$

To make a single prediction, six samples are taken at different time intervals. For example, to predict the value for the next second, then the data from the previous six seconds are considered with different weightage to give the prediction value. The weightage of the intervals are derived in [4] and remain constant.

This prediction model was given for the prediction of energy, but if the variations of the data are smooth, then this could possibly be used for the prediction of other types of data.

2.3.2 Other Models

Other EP models include fixed parameter weighting factor (FPWF) and Adaptive Neural Network (ANN) based EP models. FPWF based energy prediction models are Exponential Weighted Moving Average (EWMA) [28], Weather Conditioned Moving Average (WCMA) [28], Accurate Solar Energy Allocation (ASEA) [29], Q-learning-based solar energy prediction (QL-SEP) [30], Pro-Energy (PROfile Energy Prediction Model for solar and wind energy harvesters) [31].

EWMA uses the information from previous days combined with historical average of the data to predict the energy. This model is suitable for consistent weather and longer slots.

WCMA is the advanced form of EWMA and uses the mean value of the energy of the current and previous day to predict the energy for short term. This model is more accurate and less computationally complex.

ASEA is proposed considering short-term conditions and unpredictable weather. This is based on the ratio between the real harvested energy and the predicted value. Although this model does not require as many historical data, according to its authors it produces less accurate predictions than WCMA [4].

Pro-Energy (PROfile Energy prediction model) uses the historic data from the past days to predict energy. This prediction model is designed for solar and as well as for wind energy harvesting sources. This model compares the current conditions with the past days and predicts the energy using the most similar day from the stored data. 30 minute data interval time slots are taken for Pro-Energy [31]. The results obtained from Pro-Energy predictions are 60% better than that of EWMA and WCMA. Later on, Pro-Energy with variable-length timeslots (Pro-Energy VLT) of 30, 60, and 90 minutes was introduced to increase the accuracy and reduce the memory and energy overhead [32].

QL-SEP also uses the data of the past days and the most recent weather conditions from the current day. In this model, the day is divided equally into slots. This model also uses EWMA. A daily ratio (DR) parameter is also introduced which is the energy average and shows if the energy increases or decreases in the slots. According to its authors, QL-SEP produces better prediction values than EWMA, ASEA and Pro-Energy [30]. However, this model is designed for longer slots and if the condition changes rapidly, then the model will not produce accurate predictions and involve higher computations than the other models [4].

IPro-Energy is also based on Pro-Energy and compensates the inconsistencies in the weather conditions [32]. This model has low requirement for the storage of data and is less complex. Based on the results provided by its author, the predictions are more accurate for IPro-Energy as compared to Pro-Energy [33].

Several ANN based models are also available for the short-term predictions of energy. One of these models uses three to five months of sliding window for training the network and it was shown by its authors that the five months window produced the most accurate results [34]. However, this model is not suitable for low performance IoT nodes due to its requirement of minimum 3 months historic data. Moreover, it is not adaptive and less reliable than EWMA and WCMA algorithms, as it needs a large sliding window for training [32].

2.4 Approximate Computing

Another approach associated with the energy consumption is AC. In the AC approach, systems trade-off accuracy and energy efficiency. AC comprises a wide and open range of hardware and software based techniques to make the system energy efficient at the cost of accuracy. This approach exploits the tolerance level of the degradation in quality of the application. Several research papers proposing different types of AC techniques have been published recently, showing that AC is (again) a hot topic for research.

A taxonomy of proposed AC techniques is presented in [3] which compared the AC techniques in terms of visibility, determinism and granularity.

Visible AC techniques are those in which the errors are introduced during the execution of any specific instruction. These errors are architecturally visible, whereas invisible AC techniques introduce errors to the system silently which make the error detection and correction more challenging.

Deterministic AC techniques are those for which if there is a constant error for every input at the same initial state, and if the error is not constant and there is more than one error. Due to unknown error, debugging and testing of non-deterministic techniques can be challenging.

AC techniques can be termed as coarse-grained or fine-grained. If the number of dynamic instructions and data footprint are reduced, then the system is coarse-grained otherwise the technique is fine grained.

Table 4 inspired from [3] compares different AC techniques based on visibility, determinism and granularity.

Table 4. Taxonomy of Approximate Computing, inspired by [3].

Software Technique	Visibility	Deterministic	Coarseness
Approximate GPU kernels	Yes	Yes	Yes
Approximate Synthesis	Yes	Yes	Yes
Algorithm Selection	Yes	Yes	Yes
Code Perforation	Yes	Yes	Yes
Lossy Compression/Packing	Yes	Yes	Yes
Parallel Pattern Replacement	Yes	Yes	Yes
Bit-width reduction	Yes	Yes	No
Float to fixed conversion	Yes	Yes	No
Approximate Parallelization	Yes	No	Yes
Statistical Query	Yes	No	Yes
Synchronisation Elision	Yes	No	Yes
Hardware Technique	**Visibility**	**Deterministic**	**Coarseness**
Digital Neutral Acceleration	Yes	Yes	Yes
Interpolated memorisation	Yes	Yes	Yes
Approximate Warp Deduplication	Yes	Yes	Yes
Bit-width reduction (Voltage scaling)	Yes	Yes	No
Clock Overgating	Yes	Yes	No
Load value Approximation	Yes	Yes	No
Approximate Cache Coherence	Yes	Yes	No
Instruction Memorisation	Yes	Yes	No
Precision Scaling	Yes	Yes	No
Logical Simplifications	Yes	Yes	No
Reduced-Precision FPU	Yes	Yes	No
Analog Neural Acceleration	Yes	No	Yes
Approx Processors	Yes	No	No
Voltage Overscaling	Yes	No	No
Stochasting Logic	Yes	No	No
Approx. PCM Multi Level Cells	Yes	No	No
SRAM Soft Error exposure	Yes	No	No
Approximate Value Deduplication	No	Yes	Yes

Approx. PCM Failed Cells	No	No	No
Low Refreshed DRAM	No	No	No

AC has been used in different applications and it depends on the level of error resiliency of the application. The authors of [35] proposed an AC based method for addressing the security challenges in IoT. In this method, LSBs of 32 bit segments in IEEE754 format are replaced to hide information and produce tolerable error while MSBs is used for precise information, as illustrated in Figure 10.

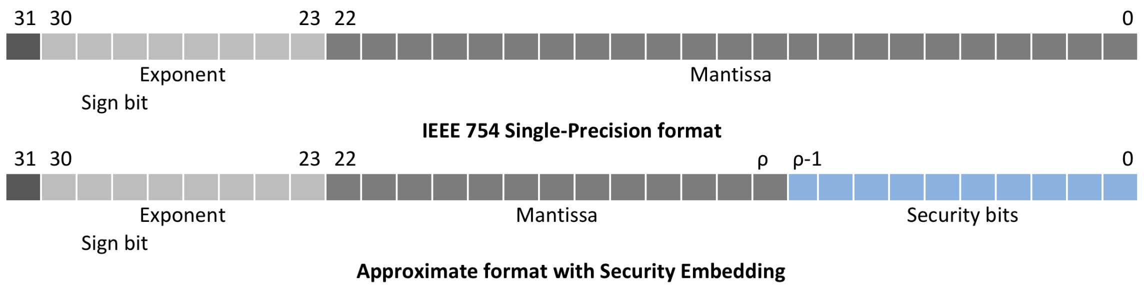


Figure 9. Security embedding in IEEE 754 Single-precision floating-point format. Figure based on [35]

In [36] several AC techniques are surveyed based on the strategies used to implement it. As AC is wide spectrum and not exclusive to the strategies presented in Table 4. One of the strategies presented in [36] is Precision Scaling, which is also used in this thesis. Precision scaling presented in [36] is the strategy to reduce the bit-width and decrease the computations and storage overhead.

The authors in [36] also surveyed a different paradigm of precision scaling called Dynamic Precision Scaling (DPS). DPS was experimented on physics based animation to improve its efficiency and find the minimum precision required at the design time. DPS detects the instability in simulations during runtime by measuring the change in energy between consecutive simulation steps and comparing it with a predefined threshold. The system restores maximum precision, if it detects instability during simulations. Then the system reduces the precision progressively and detects its minimum value until the system is stable. DPS gives different optimization opportunities to the system and leads to a hierarchical architecture at different levels for Floating Point Units (FPU) with different precisions. It was shown by the authors that the use of their technique increased the efficiency by up to 50% and performance by up to 55% as compared to the single level baseline FPUs [37].

The survey also referred to another application using precision scaling for accessing off-chip data to save energy. Precision scaling was applied to clustering problem based on mixed model with a requirement of access to off-chip data. The precision scaling is reduced to eliminate the possibility of functional error by keeping the correct order of the distance of cluster and samples. The authors of the proposed technique showed by implementing the technique that there was energy saving between 40 to 60 percent with unnoticeable loss in accuracy with a model deviation of 0.3 from the accurate data [38].

As AC has wide and open spectrum, interested readers can find additional explanations in the survey by S. Mittal in [36] and the taxonomy of AC by T. Moreau *et al.* in [3].

This chapter has presented some background information related to NVMs, EDS, TC, EP, and AC. With the understanding gained through the chapter, it is now possible to establish an experimental platform to evaluate those techniques and propose techniques for data approximation and radio control. This is described in the next chapter.

3 Experimental Setup

In this chapter, TC, DP and AP are used to implement an application for a battery-less wireless node. A test bed is set up to characterize the system and analyze the behavior of the application. Moreover, techniques for data approximation and radio control are proposed.

3.1 Hardware platform

Initial experiments were conducted using the following apparatus to replicate the power consumption results obtained in [7].

- Two FRAM based EXP-MSP430FR5739 boards.
- CC2500 Low Power 2.4 GHz RF Transceiver.
- FLUKE 123 industrial scopemeter to monitor the TC benchmark.
- KEYSIGHT E3630A DC power supply
- HEWLETT PACKARD 34401A Multimeter

On of the two nodes (Node 1) was powered using an USB power supply and used as a sender. To power the second node (Node 2) from EH source, a supply setup as shown in Figure 10 could be used, but this was not used in these experiments. For EH source, a voltage regulator with a low pass filter needs to be used at the input to stabilize the voltage. A voltage divider is used to divide the voltage in half and input at pin P1.5 for triggering the CTPL in case of voltage drop. A voltmeter is used to monitor the output of voltage divider and a scopemeter was connected with CTPL benchmark pin P4.0 to monitor if CTPL is triggered. An ammeter was connected in series with Vcc for monitoring the current used by the node.

The setup is as shown in the diagram in Figure 10.

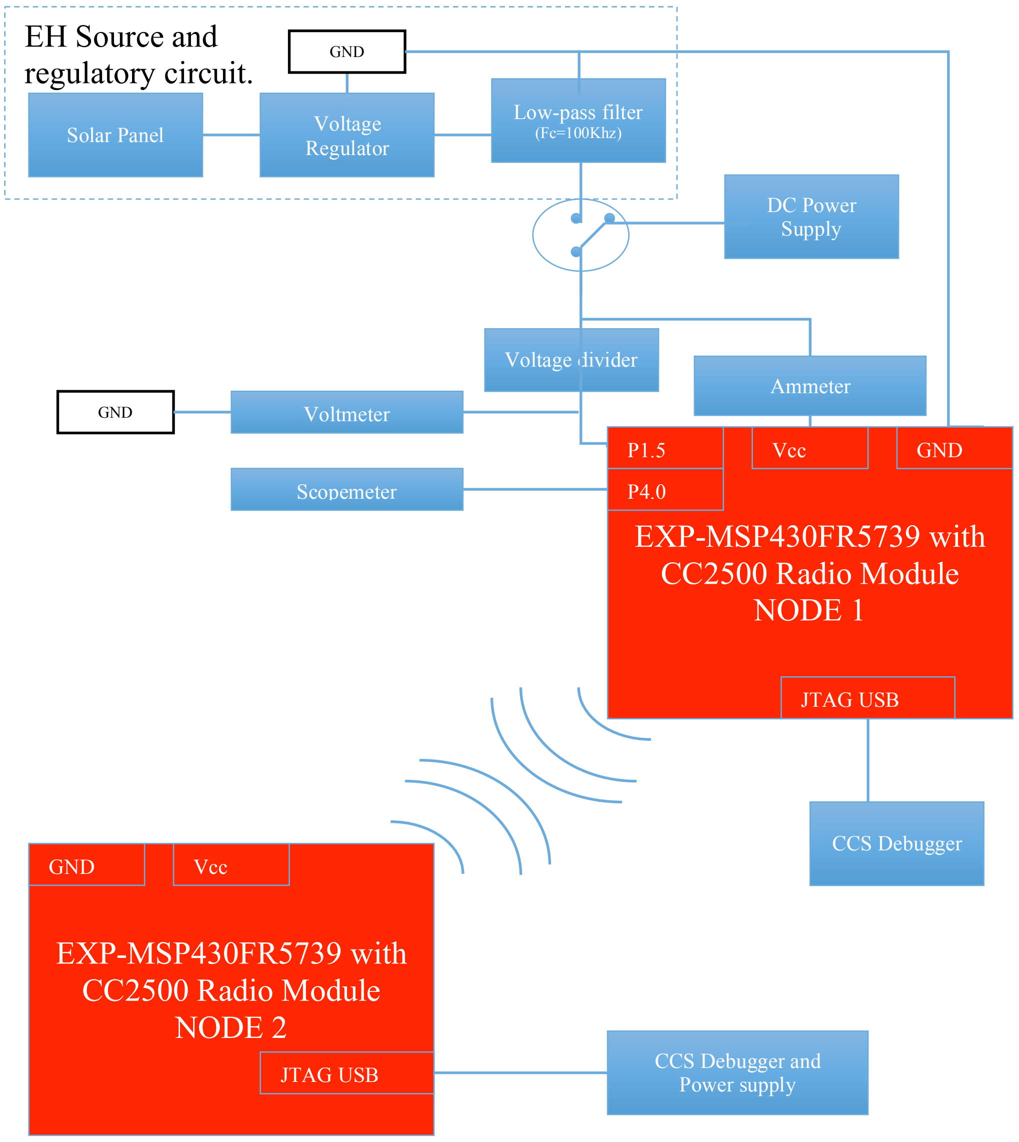

EH Source and regulatory circuit.
GND
Solar Panel
Voltage Regulator
Low-pass filter (Fc=100Khz)
DC Power Supply
Voltage divider
Ammeter
GND
Voltmeter
Scopemeter
P1.5
Vcc
GND
P4.0
EXP-MSP430FR5739 with CC2500 Radio Module NODE 1
JTAG USB
CCS Debugger
GND
Vcc
EXP-MSP430FR5739 with CC2500 Radio Module NODE 2
JTAG USB
CCS Debugger and Power supply

Figure 10. Schematic diagram for experimental setup of two nodes communicating using SimpliciTI. This setup is used to replicate that of [7].

3.2 Software modules

A simple peer-to-peer example provided by Texas Instruments (TI) was used to set up the communication. The example uses SimpliciTI, which is Texas Instruments proprietary low power radio frequency communication protocol [39]. Additionally FRAM utilities also provided by TI was used to enable TC in the system. A finite state machine was used in [7] to establish a network using TC and it was modified to the following, as shown in Figure 11.

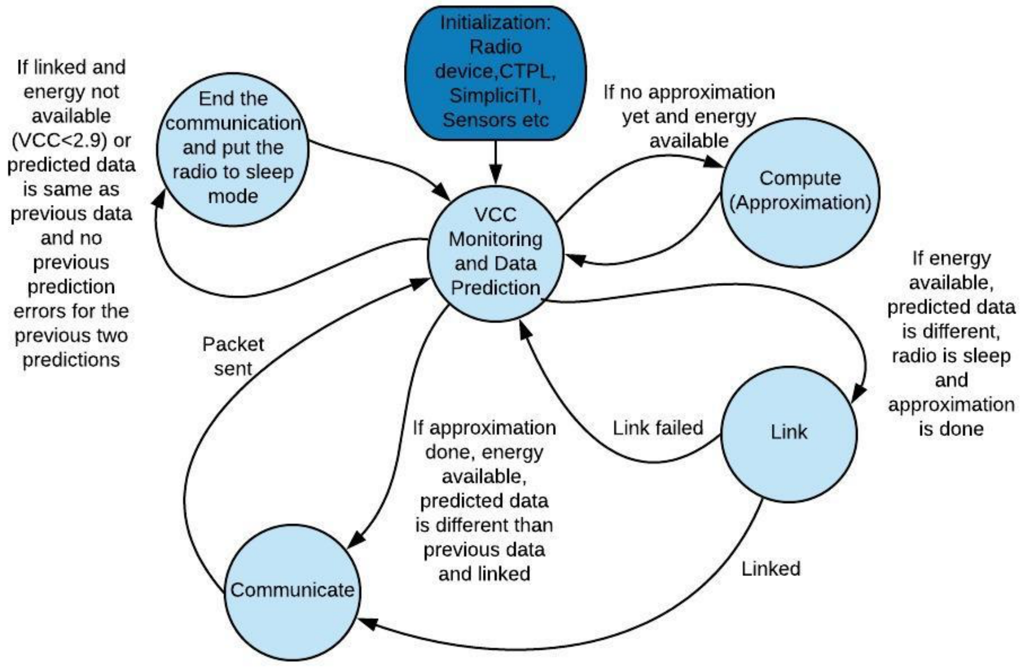

If linked and energy not available (VCC<2.9) or predicted data is same as previous data and no previous prediction errors for the previous two predictions
End the communication and put the radio to sleep mode
Initialization: Radio device,CTPL, SimpliciTI, Sensors etc
If no approximation yet and energy available
Compute (Approximation)
VCC Monitoring and Data Prediction
If energy available, predicted data is different, radio is sleep and approximation is done
Packet sent
If approximation done, energy available, predicted data is different than previous data and linked
Link failed
Link
Linked
Communicate

Figure 11. Modified finite state machine for TC and AC.. Figure inspired from [7]

As shown in Figure 11 the system initializes the radio device, CTPL library from FRAM utilities, internal temperature sensor and ADC. After initialization, the system monitors Vcc. If the energy is available, then the system collects the sensor data. After collecting the data, the system moves to compute state. The system performs all the computations in the compute state. Once the computations are done, the system monitors the voltage and if the energy is available, then the system tries to establish link with its peer. If the link fails, it jumps back to Vcc monitoring state and if the link is successful, it goes to communicate state. Once the state machine is in the communicate state, then it sends the packet and after sending the packet the state machine jumps to Vcc monitoring state. After monitoring the voltage and if the voltage drops below 2.9V, the radio ends the communication and goes to sleep mode and if the power is available, then the system collects the sensor data and goes to compute state and does the computations for approximate computing.

3.2.1 CTPL Mechanism

A high priority interrupt is used for Comparator D of MSP430FR5739 to trigger CTPL. If Vcc drops below 2.5 V, then the CTPL is triggered and CTPL_enterShutdown() function is called, as shown in Figure 12.

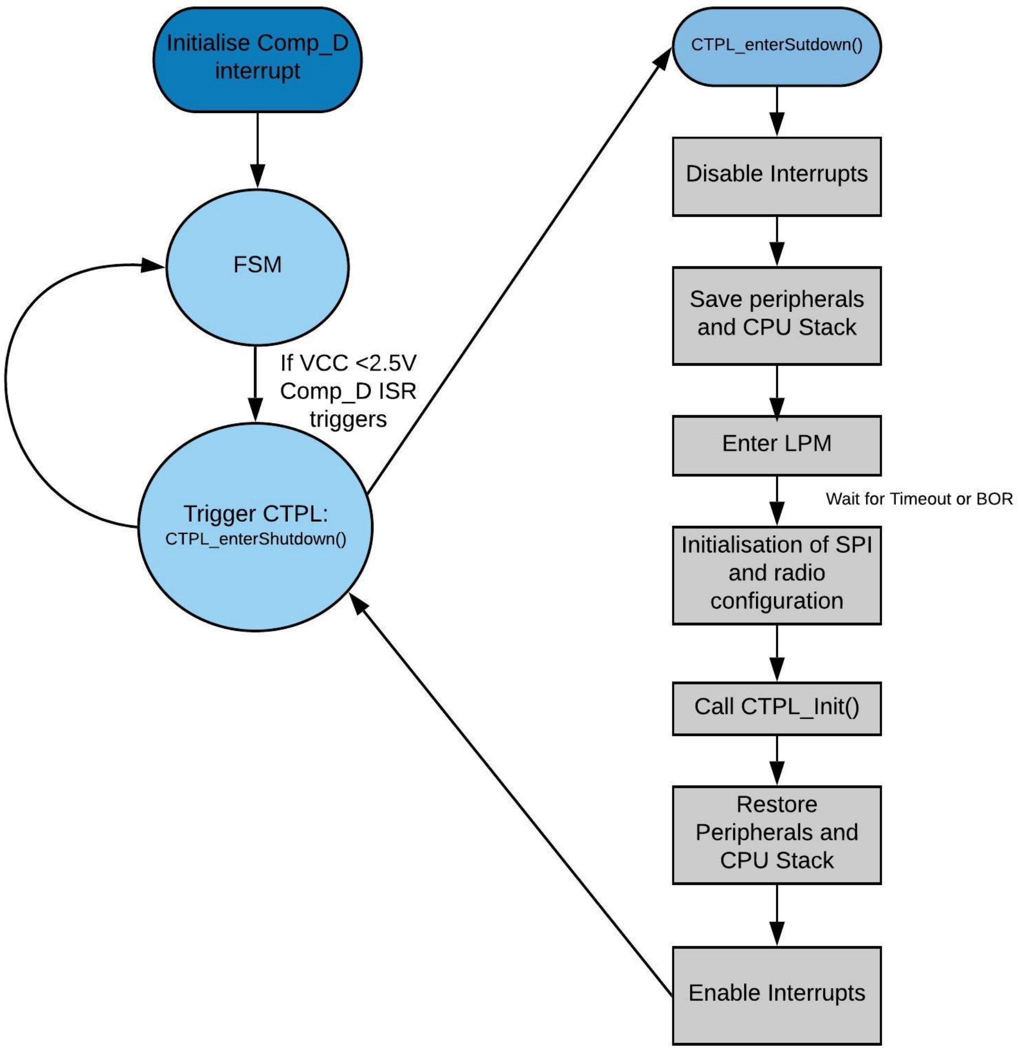

Initialise Comp_D interrupt
FSM
If VCC <2.5V Comp_D ISR triggers
Trigger CTPL: CTPL_enterShutdown()
CTPL_enterSutdown()
Disable Interrupts
Save peripherals and CPU Stack
Enter LPM
Wait for Timeout or BOR
Initialisation of SPI and radio configuration
Call CTPL_Init()
Restore Peripherals and CPU Stack
Enable Interrupts

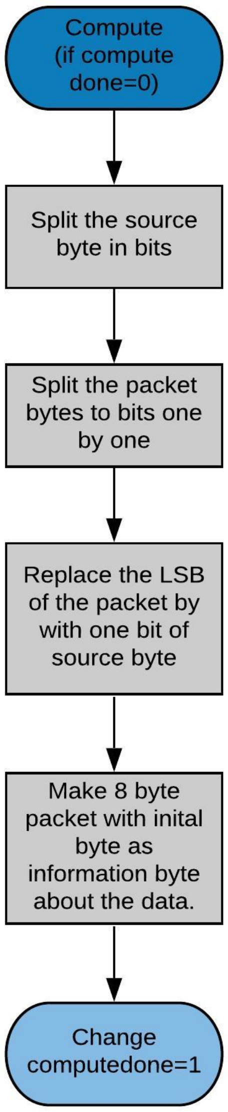
Figure 12. CTPLexecution flow and its integration with the developed program. Figure inspired by [27]

The interrupt service routine (ISR) disables all the other interrupts and takes a snapshot of the system by saving the peripherals and CPU stack to FRAM. Once the snapshot is taken, the system enters into low power mode (LPM) and waits for the system to enter the brown out reset (BOR) or time out after the availability of power.

At the availability of power, the system initializes the radio module and SPI interface to communicate with the radio module. After initialization, it calls the CTPL_init() function which restores the peripherals and CPU stack from the stored snapshot. After restoring the snapshot, the interrupts are enabled and enters to the state machine.

3.2.2 Proposed approximation of data

Once the state machine enters the compute state, it processes the data and embeds the equivalent of an additional byte of information to the existing eight bytes of information, keeping the size of the packet constant. To do so, the system splits the source byte to be embedded into the packet bytes into bits. Then the LSB of each the packet bytes are replaced with the information from the source byte and the packet is made ready to be transmitted. The flowchart for this technique is shown in Figure 13.



Compute (if compute done=0)
Split the source byte in bits
Split the packet bytes to bits one by one
Replace the LSB of the packet by with one bit of source byte
Make 8 byte packet with inital byte as information byte about the data.
Change computedone=1

Figure 13. AC execution flow for the developed program

3.2.3 Proposed Radio Control using DP

LINE-P data prediction is added to the application to predict the future data and controls the radio based on the predictions. For a temperature monitoring application, the algorithm then checks the previous prediction errors and whether the absolute error for previous two predictions is less than 2 °C. It also checks if the difference between the current data and the predicted data is less than 2 °C. If both conditions are true, then the system put the radio to sleep and monitors the new data and Vcc. The system continues to collect the data and do the predictions and once the absolute predictions error is higher than 2 °C, or if three consecutive transmissions are missed, then the status of radio is set to awake and resumes transmissions.

3.3 Challenges

During the implementation process, several challenges have been encountered that include issues with hardware as well as software.

3.3.1 Hardware related issues

Initially it has been tried to implement the system and replicate the results from [7]. This was not possible as there were current leaking issues with the available off the shelf MSP430FR5739 microcontrollers. The main challenge was to identify the issue; however, using the trial and error method it was identified. To resolve this problem new devices were ordered which was time consuming.

3.3.2 Software related issues

To identify the hardware related issues, it was recommended to update the software and implement the project on a different system. But the new compiler and software were incompatible with our application.

It was advised that maybe the issues are due to software. It was tried to start the project from scratch. However, different sorts of issues including incompatibility of the platforms, unavailability of the peer-to-peer example for MSP430FR5739 and lack of support from TI for SimliciTI protocol using MSP430FR5739 rendered this approach impossible. Due to these issues, it was recommended to use the available system instead.

This chapter has presented the experimental setup used in this thesis, including the CTPL implementation and modifications made the state machine of the application. Moreover, data approximation and radio control techniques have been proposed. The next chapter presents the results, which have been obtained with this setup.

4 Results

Despite the previously-mentioned challenges, it was finally possible to produce the desired results in line with the objective of this thesis. This chapter presents theresults obtained based on the experiments that have been carried out using the experimental setup explained in the previous chapter.

4.1 Current consumption

Experiments were conducted to characterize the power consumption of one node in different states of the application. To find out the power consumption, the input current and voltage to the node were recorded at different states. It was noted that when the radio is in sleep mode, the maximum current drew by the node was 14 mA, while the node drew 20 mA when communicating with its peer. With a 3 V supply voltage, this translate to 42 mW and 60 mW, respectively. The characterization of the application with respect to power is shown in Table 5.

Table 5. Power Consumption of MSP430FR5739 plus CC2500 radio module in different states of the application

State	**Current (average)**	**Voltage**	**Power**
Idle	2 mA	3 V	6 mW
Computations	14 mA	3 V	42 mW
Linking	20 mA	3 V	60 mW
Communicating	20 mA	3 V	60 mW

4.2 TC Results

TC results were replicated based on the work in [5]. Figures 14 and 15 show that the CTPL utility is triggered when the source voltage level goes below 2.5 V where the CPU state (PC, SP, GPR, Stack, Registers, etc.) is saved to the FRAM memory and retained for as long as the power is not available. Once the power is available again, the CPU state is resumed and processor starts working from the point where it stopped due to power loss. It can be seen from Figure 14 that CTPL was triggered when the power loss happened (tested by manually decreasing the voltage using the power supply knob) and after the once the power was resumed, the CTPL Benchmark pin P4.0 toggled as shown.

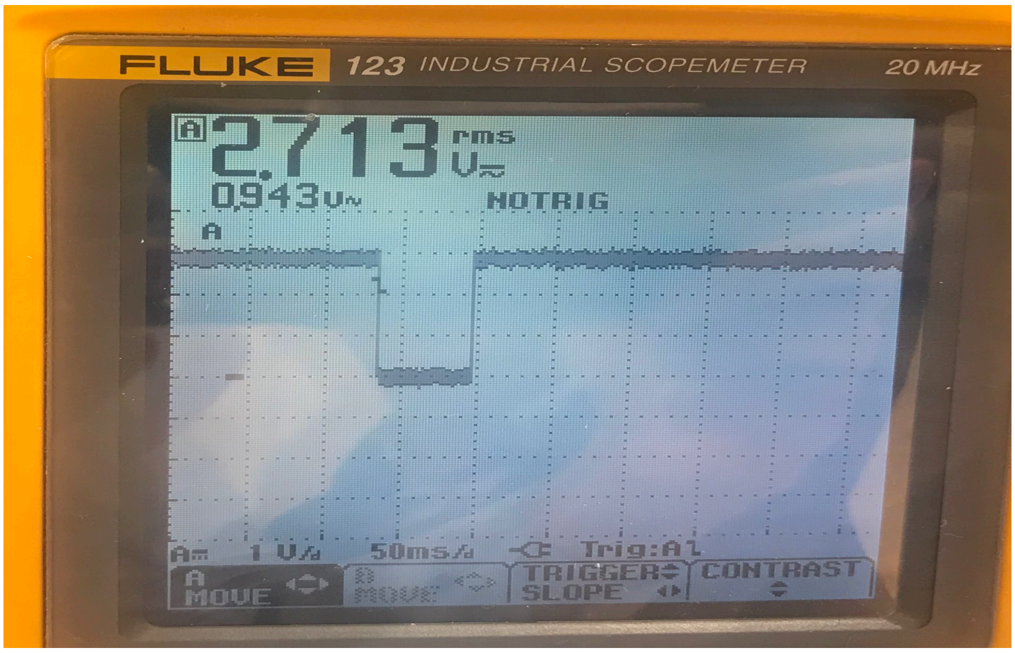


Figure 14. Triggering of CTPL utility in case of voltage drop

When the reference input voltage at P1.5 was removed, the system triggered CTPL and then checked that the input voltage was available. As a result, the benchmark pin (P4.0) was toggling as shown in Figure 15.

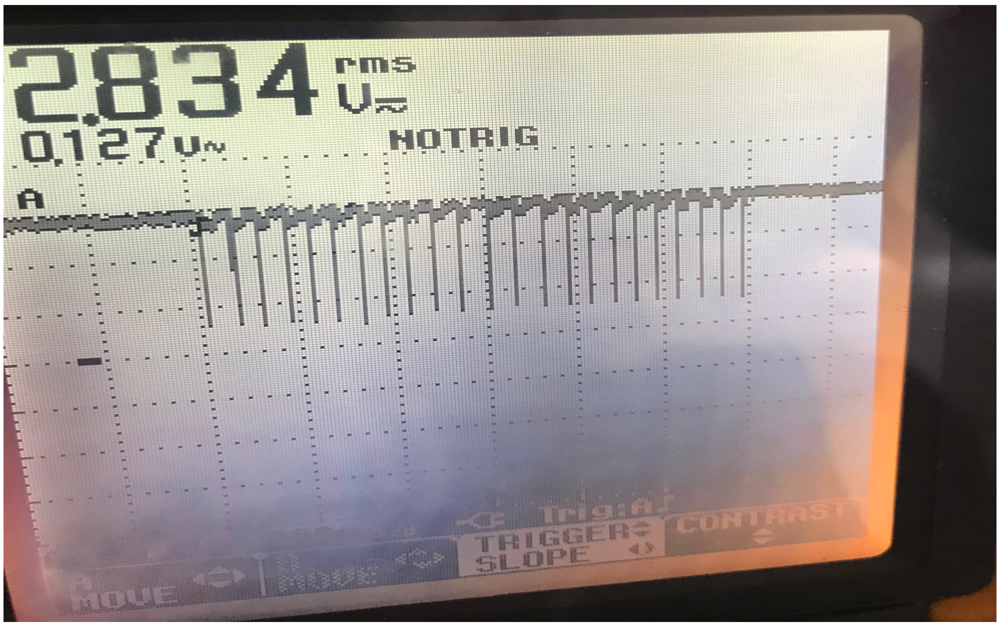


Figure 15. Triggering of CTPL utility (benchmark pin P4.0) in case of removing the reference input voltage at pin P1.5

4.3 DP results

The dataset of the New York city wind-speed [40] was fed into the proposed data prediction algorithm (a slightly adapted version of LINE-P EP model). The selected data was recorded on 29/10/2012 starting from 8:00 am and collected for each hour. The six hours data was fed into our model to predict the future 1-hour data, as shown in Figure 16.

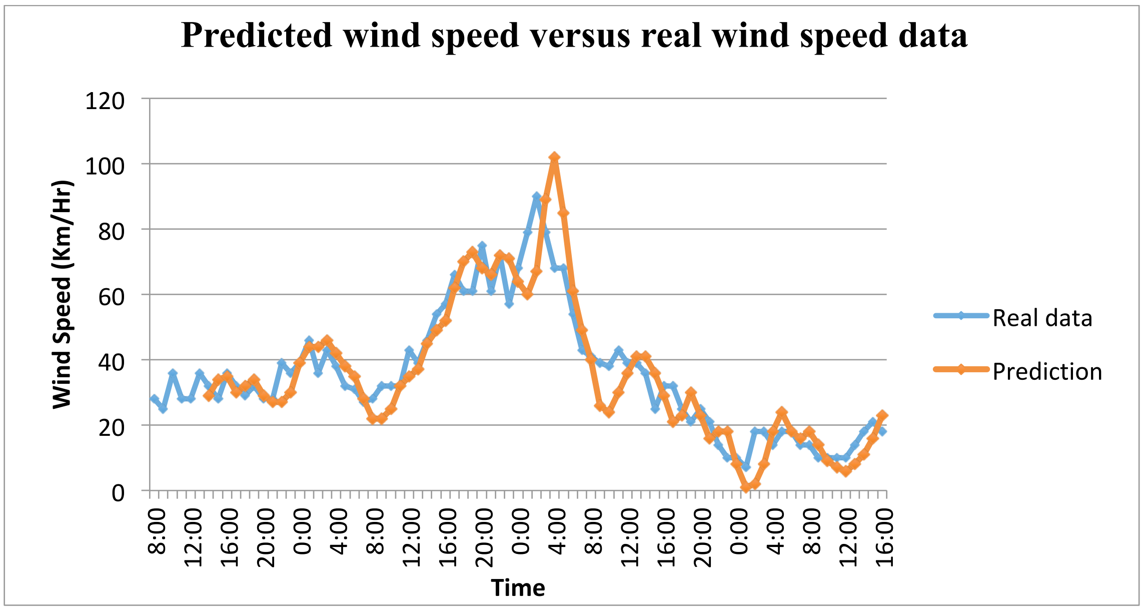


Figure 16. Wind-speed data versus predicted data using the slightly adapted LINE-P energy prediction model

Figure 16 shows the predicted results (in red) from the proposed technique along with the real data (in blue) for the wind speed in NY City. The results show that the predicted values are very close to the real values. However, minor errors do exist in the predicted values. Table 5 presents the MSE, MAE and MPE measured for both the real values of data and predicted ones. As the wind speed ranges from 0-100 km/h and in some instants the prediction errors are very high, so the MSE value of the whole dataset is 68.74, as shown in Table 6. Considering the variations in the dataset, the obtained MAE is 5.76% and the obtained MPE is 19.5%. The error value is higher as it is due to squaring the individual errors and is amplified because of a few outliers.

Table 6. MSE, MAE and MPE for DP of wind-speed

Mean Squared Error (MSE)	68.74
Mean Absolute Error (MAE)	5.76
Mean Percentage Error (MPE)	19.5%

In the second case, a dataset of the daily temperature data of the city of Rome [41] was taken into account to be used in our DP model. Figure 17 presents the values of the real temperature values (in blue) and the predicted temperature values (in red). In this case the difference between the real and predicted values are visibly higher as the data range is small (0 to 16) and the variations are also higher.

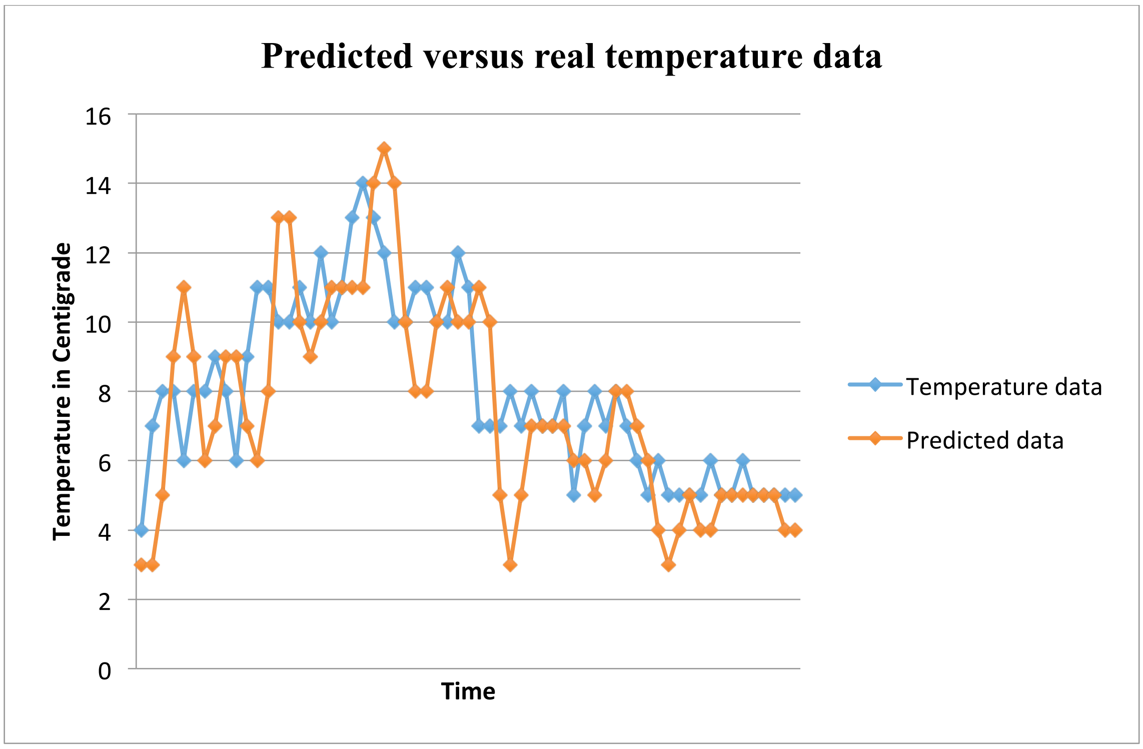


Figure 17. Temperature data versus predicted data using LINE-P energy prediction

Table 7 lists the MSE, MAE and MPE between the real and predicted values. As the temperature ranges from 0 to 16 in one day, the MSE value of the whole dataset is lower. Considering the variations in the dataset, the obtained MAE is 1.66 and the obtained MPE is 21.0%.

Table 7. MSE, MAE and MPE of DP for temperature in Rome

Mean Squared Error (MSE)	4.58
Mean Absolute Error (MAE)	1.66
Mean Percentage Error (MPE)	21.0%

4.4 AC results

In what follows, the results from incorporating approximate computing into the existing platform are presented. Our results illustrate that the benefits of exploiting approximate computing techniques into the existing platform are two folds. Firstly, more data can be incorporated into the existing data packet at the cost of losing a certain amount of accuracy. Secondly, the numbers of transmissions that are needed to transfer a certain amount of data from the sender to the receiver are reduced, at the cost of transmitting less accurate data. Both of these concepts are explained in the following.

Incorporating AC in a single packet for adding extra information to be sent as part of a single transmission

Initially, a single transmission was taken into account where a packet that is composed of 8 bytes of data is transmitted from the sender to the receiver. In the first case, the LSB of each of the 8 bytes of the data packet, i.e. 8 bits in total, are combined to enable the equivalent of an additional byte to be sent as part of the existing data of the packet. This can be seen in Figure 18. Reducing the data width of each byte by 1 bit frees 12.5% of the total packet payload for the extra information to be carried at the cost of reduced accuracy. The maximum accuracy loss is in the range of ±1 for any integer data.

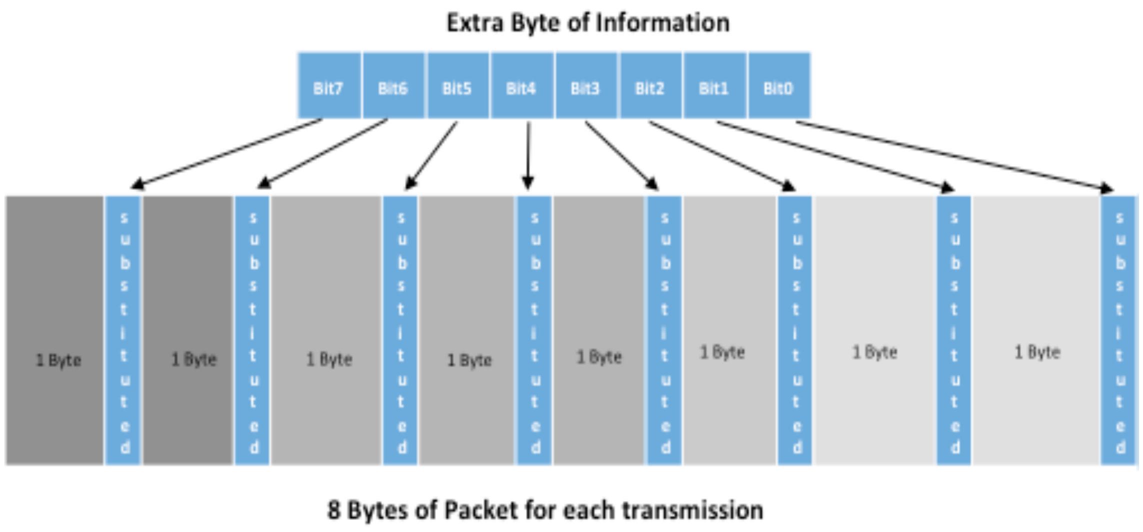
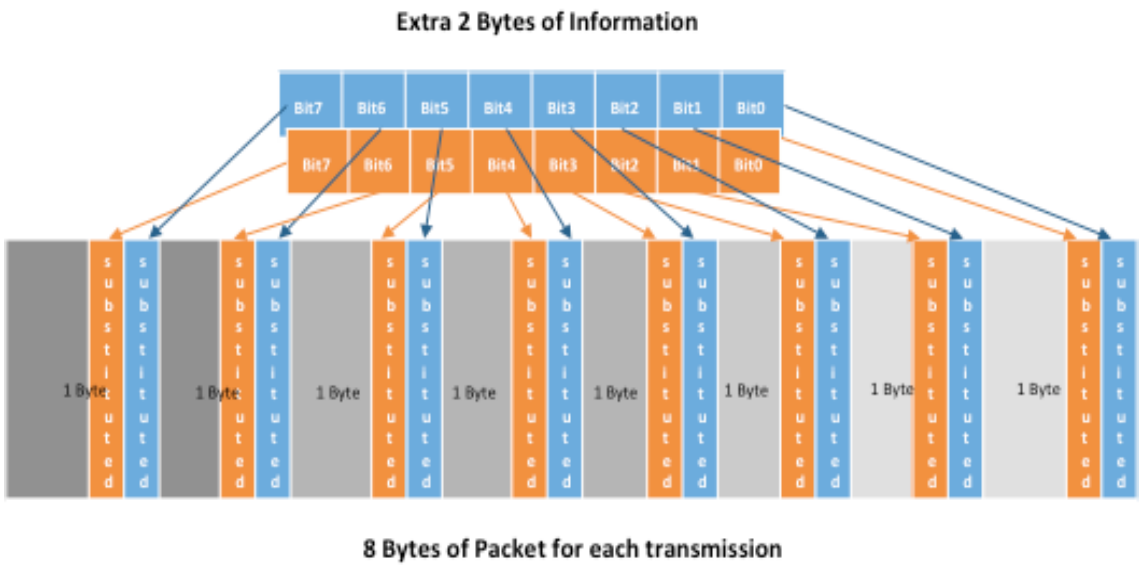

Extra Byte of Information
Bit7
Bit6
Bit5
Bit4
Bit3
Bit2
Bit1
Bit0
1 Byte
substituted
8 Bytes of Packet for each transmission

Figure 18. Enabling the equivalent of 1 additional byte of information by approximating the single LSB of the 8 bytes of a packet, whereby the additional information bits are substituted to the approximated bits.

In the second case, the first two LSBs of each of the 8 bytes of the packet (i.e. 16 bits in total) were combined to enable the equivalent of an additional 2 bytes of information to be sent as part of the existing data of the packet. This is shown in Figure 19. The price for targeting 2 bits in each byte of the original 8-byte packet is that the data reduces 2 bits of its accuracy but an extra payload of 25% is freed for the extra information to be transmitted as part of the existing data. The maximum accuracy loss for a 2 bit reservations is in the range of ±3 for any integer data.



Extra 2 Bytes of Information
Bit7
Bit6
Bit5
Bit4
Bit3
Bit2
Bit1
Bit0
1 Byte
substituted
8 Bytes of Packet for each transmission

Figure 19. Enabling the equivalent of 2 additional bytes of information by approximating 2 LSB bits in each of the original 8 bytes of a packet, whereby the additional information bits are substituted to the approximated bits.

In the third case, the first three LSBs of each of the 8 bytes of the packet (i.e. 24 bits in total) were combined to enable the equivalent of an additional 3 bytes of information that can be sent as part of the existing data of the packet. In this case the accuracy is further reduced but payload to transfer extra data is increased. Assuming that the application cannot tolerate more loss of accuracy, no further approximations are performed so as not to decrease the accuracy any further.

Table 8 summarizes this whole concept of exploiting the existing payload of a single packet transmission by substituting the bits in the existing packet for sending additional information as part of the existing data. It is clear that as the number of substituted bits in each packet increases, the payload for extra information also increases but the accuracy drops accordingly.

Table 8. Added information through approximation and loss of accuracy in a single transmission

Added information through approximation (in a single transmission scenario)						
Packet Size	**Approximated bits per Byte**	**Added information per Packet**	**Max Error occurrence (in integer)**	**# of transmissions**	**% of information per packet**	**Power consumed per transmission**
8 Bytes	0	0 Byte	0	1	100.0%	20 mA * 3.3 V = 66 mW
	1	1 Byte (approximated)	±1		112.5%	20 mA * 3.3 V = 66 mW
	2	2 Bytes (approximated)	±3		125.0%	20 mA * 3.3 = 66 mW V
	3	3 Bytes (approximated)	±7		137.0%	20 mA * 3.3 V = 66 mW

Incorporating AC in a multi node scenario where the data size and the number of transmissions required to send this data is taken into account

In the second scenario, the number of transmissions from each device were taken into account by targeting the amount of data that is transmitted from the sender to the listener. No significant gains could be achieved by considering only a single transmission of 8 bytes of a packet. However, the number of transmissions could be reduced significantly if large chunks of data is taken into account. Let us consider a case where 80 bytes of data is to be sent from the sender to the listener such that 10 transmissions are needed for all the data to be sent where each packet can carry 8 bytes of data.

If the LSB of each byte of 80 bytes of this data is targeted, a total of 80 bits, i.e. the equivalent of an additional 10 bytes of data can be incorporated within the existing data size. So, out of 80 bytes of this data, the equivalent of 10 bytes are incorporated within the existing space, reducing the whole data size to 70 bytes at the cost of reducing the accuracy of the data bits by 12.5%. And to send 70 bytes of less accurate data with the same packet size of 8 bytes, only 9 transmissions are needed in total. Similarly, if 800 bytes of data is considered, the transmissions could be reduced from 100 to 90 and so on. Table 9 summarizes these details with the actual number of transmissions needed for transmitting 100% of accurate data along with reduced number of transmissions for reduce-accuracy data and the corresponding percentage of accuracy loss.

The last column of Table 9 shows that these reduced number of transmissions are a multiple of the transmitting nodes when a multi-node scenario is taken into account. So, if a single device is to transmit 8K bytes of data in a single day, and if we can reduce 100 transmissions through approximating the data, so in 1000 such devices that transmit 8K bytes of data each, the total number of transmissions are reduced by 10K, which is expected to have a significant impact on the overall network efficiency in terms of energy and bandwidth.

Table 9. Reducing the number of transmissions through approximation

Reduced number of transmissions through approximation					
Transmission Data	**Approximated bits per byte**	**Added information per packet**	**Number transmissions per device**	**Percentage of reduced transmissions per device**	**Reduced number of transmissions in a multi node scenario**
8 Bytes	0	0	1	No reductions possible in a single packet	NA
	1	1 byte			
	2	2 bytes			
	3	3 bytes			
80 Bytes	0	0	10	0%	No. of Devices * transmissions/Device
	1	10 bytes	9	10%	
	2	20 bytes	8	20%	
	3	30 bytes	7	30%	
800 Bytes	0	0	100	0%	No. of Devices * transmissions/Device
	1	100 bytes	90	10%	
	2	200 bytes	80	20%	
	3	300 Bytes	70	30%	
8000 Bytes	0	0	1000	0%	No. of Devices * transmissions/Device
	1	1000 Bytes	900	10%	
	2	2000 Bytes	800	20%	
	3	3000 Bytes	700	30%	

The dataset of the New York city wind-speed of section 3.2.2 [40] was incorporated with a single bit (LSB) approximation for transmission from the sender to the listener. The results obtained at the listener were then analyzed. Figure 18 summarizes the results of original data vs. the approximated data. The graph shows that the approximated data is almost identical to the original data as the two curves overall each other, i.e. the approximated graph is hidden behind the graph of original data.

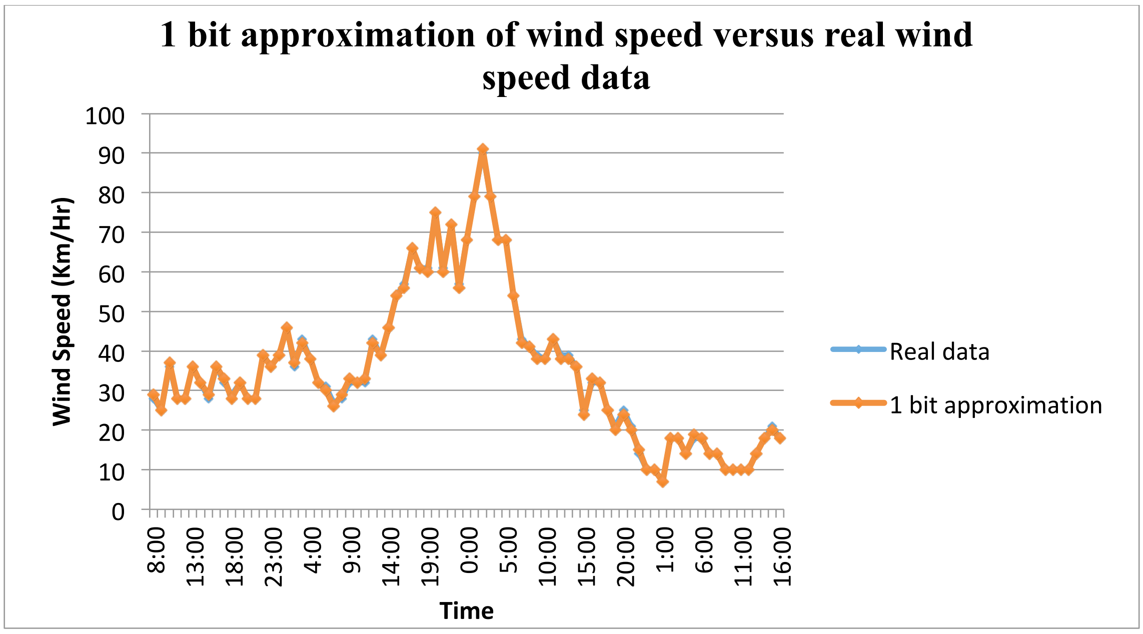


Figure 20. 1 bit approximation of the wind speed dataset of NY city (The real data and the 1 bit approximation curves overlap each other).

Table 10 also depicts high accuracy since the values of the MSE, MAE and MPE are quite low.

Table 10. MSE, MAE and MPE of approximated wind speed data of NY City

Mean Squared Error (MSE)	0.33
Mean Absolute Error (MAE)	0.33
Mean Percentage Error (MPE)	1.06%

Similarly, the temperature dataset of the Rome city of section 3.2.2 [41] was incorporated with a single bit (LSB) approximation for transmission. The results obtained at the listener are summarized in Figure 21. The graphs shows that with 1 bit approximation the accuracy losses are higher for this data set as compared to the previous one. This is also depicted from the values of MSE, MAE and MPE as summarized in Table 11.

Table 11. MSE, MAE and MPE of approximated temperature data of Rome City

Mean Squared Error (MSE)	0.74
Mean Absolute Error (MAE)	0.66
Mean Percentage Error (MPE)	11.11%

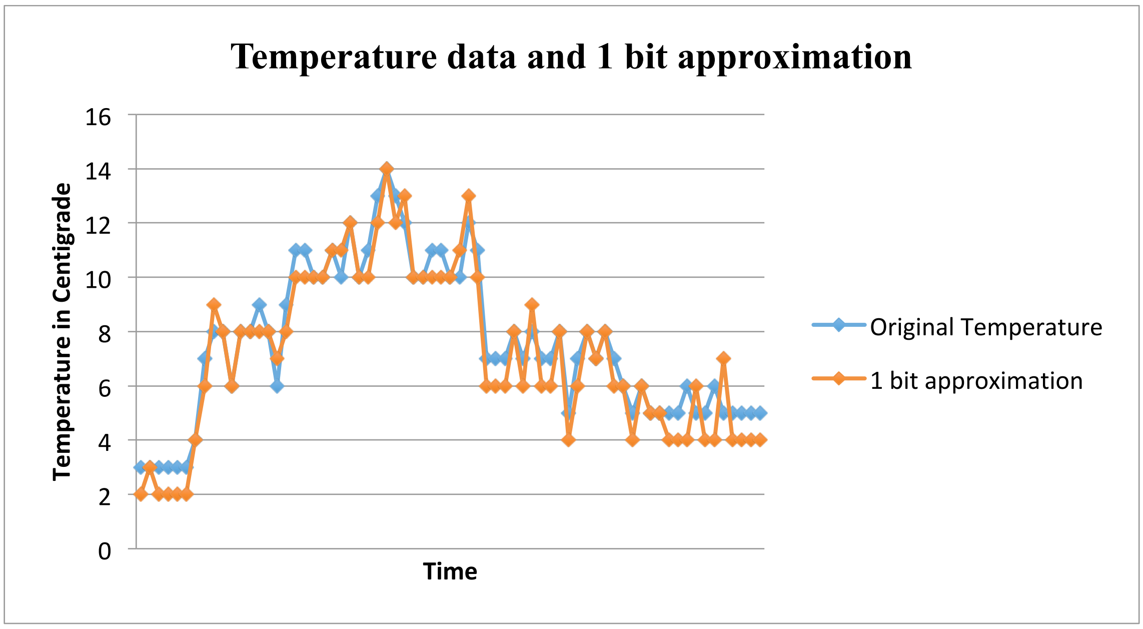


Figure 21. 1bit approximation of the Temp dataset of Rome city

Next, the incorporation of 2 bits approximation in the New York city wind-speed of section 3.2.2 [40] and the temperature dataset of the Rome city of section 3.2.2 [41] is summarized in Figure 22 and Figure 23 where the values of MSE, MAE and MPE are summarized in Table 12 and Table 13. The accuracy slightly decreases for both datasets, as expected.

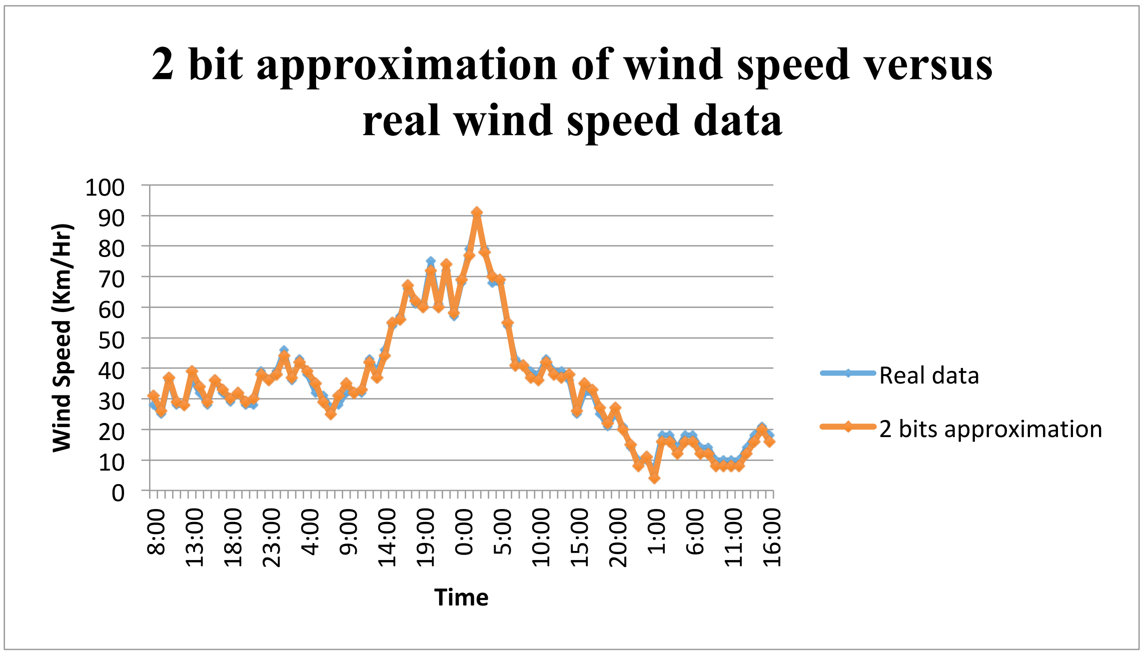


Figure 22. 2 bit approximation of the Wind dataset of the NY city. A few more differences between the real and approximated curves can be seen as compared to the previously shown 1 bit approximation.

Table 12. MSE, MAE and MPE of 2 bit Approximated Wind speed data of NY City

Mean Squared Error (MSE)	2.61
Mean Absolute Error (MAE)	1.36
Mean Percentage Error (MPE)	6.11%

Similarly, the comparison of 2 bit approximation of the temperature data from Rome and the original temperature data is shown in Figure 23, while the MSE, MAE and MPE are shown in Table 13.

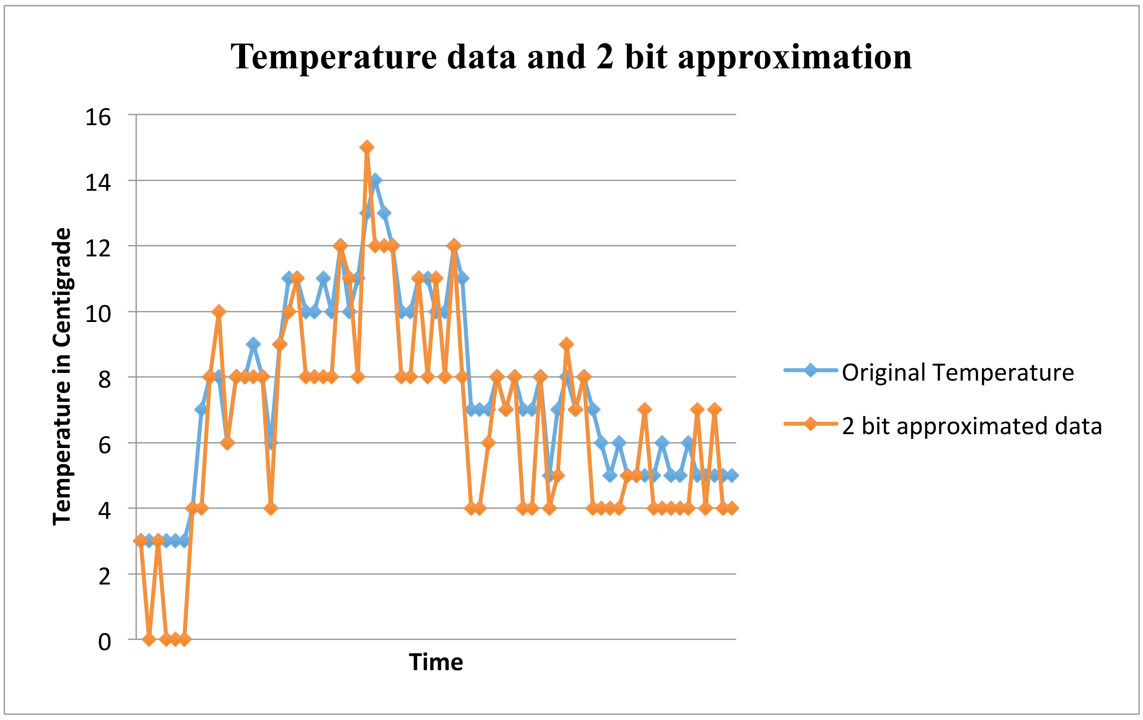


Figure 23. 2 bit approximation of the Temperature dataset of Rome. A few more differences between the real and approximated curves can be seen as compared to the previously shown 1 bit approximation.

Table 13. MSE, MAE and MPE of 2 bit Approximated temperature data of Rome

Mean Squared Error (MSE)	2.79
Mean Absolute Error (MAE)	1.25
Mean Percentage Error (MPE)	20.32%

4.5 Results with radio control

In the next set of experiments, DP was used to control the radio transmissions. In this experiment, one packet of information, including information type, temperature data, current time (3 bytes), current voltage level, talker ID and next predicted data, was sent.

In the proposed technique, radio transmissions are controlled by the value of the current prediction and absolute error of the two previous predictions. If the absolute error of the two previous predictions and the difference between the prediction and the current value is less than 2, then the system will send the current data and put the radio into sleep mode. The peer uses the same algorithm and will go to sleep. If consecutively three packets are not transmitted, then the fourth packet will be transmitted unconditionally.

The temperature dataset was used for the experiments to see the effects of the radio control strategy with DP. It was noted that out of 63 packets, only 48 transmissions were done, saving a total of 15 transmissions. This means that the total numbers of transmissions were reduced by 23.8%.

This strategy works well with the data in which variations are smooth or the data is consistent meaning that the predictions accuracy is high. If there are numerous variations then this strategy is not useful due to possibility of losing new data.

4.6 Observations

The following points were observed during the experiments and analysing the results.

1. The approximation error depends on the data scale. If the scale is large, e.g. in case of wind speed for NY city (0-100 km/h), the MPE with 1 bit approximation was 1.06%, and with 2-bit approximation, the MPE was 6.11%. If the scale is small, e.g. in case of temperature in Rome for the month of January (0-15 °C), the MPE with 1 bit approximation was 11.11% while the MPE for 2 bit approximation was 20.32%. It was observed that with smaller scale the MPE is irrelevant as it is scale dependent.
2. The MAE for 1 bit approximation and 2 bit approximation of the temperature data from Rome was 0.66 and 1.25 °C, respectively. Similarly, the MAE for 1 bit and 2 bit approximations of wind speed of NY City were 0.33 and 1.36 km/h. It was observed that the MAE for 2 bit approximation was 1.25 °C. The maximum error for 1 bit approximation could be ±1 while for 2-bit approximation it could be ±3.
3. The DP was used on the temperature data of Rome and wind speed of NY and it was observed that there were drastic changes in some occasions and the

predictions were not as accurate as expected. Due to these variations, the MSE for wind speed was 68.74 due to squaring the individual error and it was observed that in this scenario, the MSE becomes irrelevant, while the MAE was only 5.76 km/h meaning that the absolute error was not high, and more often it was in the range of acceptable numbers as can be seen in Figure 16. The MSE for the temperature data was 4.58 and it was due to the smaller scale of the data but the data prediction model also showed that the MAE was 1.66 °C for predictions. It was observed that the model is more suitable for the data with smoother variations.

4. It was also observed that in both datasets, a reduction of 12.5% in number of transmission for 1 bit and 25% reduction in number of transmissions for 2-bit approximation was achieved.
5. Radio control results showed that there were additional 23.8% reductions in the number of transmissions for the temperature data but this reductions depends on the accuracy of DP, for example in case of wind speed, a 5 km/h difference was kept in mind but even with 5 km/h error tolerance, there were no reductions in the number of transmissions due to high variations in the data predictions.
6. With these results it was noted that the approximation techniques combined with data DP makes that the systems error is visible, deterministic and coarse grained with the loss of packets due to errors in data predictions.

5 Summary and Perspectives

With an aim to efficiently utilize the energy obtained from various energy harvesting sources, the techniques of transient computing and approximate computing were combined with the data and energy prediction models so as to exploit the harvested power in a much efficient and effective way so as to prolong the life-time of a battery-less energy-driven wireless node. These mentioned techniques were applied to a test-bed consisting of two nodes that combine Texas Instruments' FRAM based MSP-EXP430FR5739 micro-controllers with CC2500 radio modules that communicate with each other through the Texas Instruments' SimpliciTI protocol.

To exploit the transient computing capability, the Compute Through Power Loss utility of the TI's FRAM based microcontrollers were utilized. On top of that, two approximate computing techniques were incorporated into the SimpliciTI protocol at different levels so as to increase the utilization of its existing bandwidth. To further exploit the number of transmissions, the data and energy prediction models were used in the best possible way to reduce the energy consumptions during transmissions.

Based on the obtained experimental results, it was concluded that transient computing can play an important role in enabling battery-less energy harvesting nodes where the power is not stable and continuous. To further reduce the power consumption of energy harvesting nodes, approximate computing offers much potential at the cost of accuracy loss. This thesis results show that the accuracy lost due to incorporation of approximations depend on the nature and type of data that these nodes have to deal with. The accuracy of the data is also dependent on the number and position of the approximated bits. For example, the data that can be represented only in lower order bits (in binary), a 1-bit approximation leads to 11.11% of MPE, whereas for the data that can be represented in higher order bits (in binary), a 1 bit approximation gives an MPE of 1.06% only. The results also illustrate that the number of transmissions could be reduced by 10%, 20%, or even 30% through incorporation of approximations but again at the loss of accuracy.

Overall, the work presented in this thesis illustrates that it is feasible to combine transient computing, approximate computing, and data prediction for reducing the overall energy consumption of the wireless nodes.

The work and results invite further research and development in line with the used approaches.

Firstly, the implemented techniques could be evaluated on more use-cases with various constraints and error tolerance levels. In addition, the CC2500 RF module could be replaced by a e.g. Quectel BG96 radio module to enable transient computing, approximate computing and data prediction on larger scale network such as NB-IoT.

Secondly, energy prediction itself (in addition to data prediction) could be used to improve the decision-making process regarding both transmission and computation.

Thirdly, more advanced (and thus more adaptive) prediction models based on e.g. neural networks and their implementation requirements could be investigated.

References

[1] G. V. Merrett and B. M. Al-Hashimi, “Energy-driven computing: Rethinking the design of energy harvesting systems,” *Proc. 2017 Des. Autom. Test Eur. DATE 2017*, pp. 960–965, 2017.

[2] A. Rodriguez Arreola *et al.*, “Approaches to Transient Computing for Energy Harvesting Systems: A Quantitative Evaluation,” *Proc. 3rd Int. Work. Energy Harvest. Energy Neutral Sens. Syst. - ENSsys ’15*, pp. 3–8, 2015.

[3] T. Moreau *et al.*, “A Taxonomy of General Purpose Approximate Computing Techniques,” *IEEE Embed. Syst. Lett.*, vol. 10, no. 1, pp. 2–5, Mar. 2018.

[4] F. Ahmed, G. Tamberg, Y. Le Moullec, and P. Annus, “Dual-Source linear energy prediction (LINE-P) model in the context of WSNs,” *Sensors (Switzerland)*, vol. 17, no. 7, 2017.

[5] B. Things and Q. Ju, “Predictive Power Management for Internet of Battery-Less Things,” *Ieee Trans. Power Electron.*, vol. 33, no. 1, pp. 299–312, 2018.

[6] H. Jayakumar, A. Raha, and V. Raghunathan, “QUICKRECALL: A low overhead HW/SW approach for enabling computations across power cycles in transiently powered computers,” *Proc. IEEE Int. Conf. VLSI Des.*, pp. 330–335, 2014.

[7] F. Ahmed, C. Kervadec, Y. Le Moullec, G. Tamberg, and P. Annus, “Autonomous Wireless Sensor Networks: Implementation of Transient Computing and Energy Prediction for Improved Node Performance and Link Quality,” *Comput. J.*, no. October, 2018.

[8] G. V Merrett, “Invited - Energy harvesting and transient computing,” *Proc. 53rd Annu. Des. Autom. Conf. - DAC ’16*, pp. 1–2, 2016.

[9] D. Kumar, “A Study about Non-Volatile Memories,” no. 10, 2016.

[10] Y. Xie, Y. Li, B. Song, and C. Jiang, “Study on a Conceptual Mobile Memory of Interface Designation,” no. Iceeecs, pp. 260–264, 2018.

[11] B. Ransford, J. Sorber, and K. Fu, “Mementos: System support for long-running computation on RFID-scale devices,” *Asplos*, no. May 2013, pp. 159–170, 2011.

[12] D. Balsamo, A. S. Weddell, G. V Merrett, B. M. Al-Hashimi, D. Brunelli, and L. Benini, “Hibernus: Sustaining Computation During Intermittent Supply for Energy-Harvesting Systems,” *IEEE Embed. Syst. Lett.*, vol. 7, no. 1, pp. 15–18, 2015.

[13] T. Instruments, “Intelligent System State Restoration After Power Failure With Compute Through Power Loss Utility,” no. April. pp. 1–16, 2015.

[14] H. Jayakumar, A. Raha, and V. Raghunathan, “Hypnos: An ultra-low power sleep mode with SRAM data retention for embedded microcontrollers!,” in *2014 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*,

2014, p. 10.

[15] D. Balsamo *et al.*, “Graceful Performance Modulation for Power-Neutral Transient Computing Systems,” *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, vol. 35, no. 5, pp. 738–749, 2016.

[16] C. Pan *et al.*, “A lightweight progress maximization scheduler for non-volatile processor under unstable energy harvesting,” *ACM SIGPLAN Not.*, vol. 52, no. 4, pp. 101–110, 2017.

[17] D. Balsamo *et al.*, “Hibernus++ : A Self-Calibrating and Adaptive System for Intermittently-Powered Embedded Devices,” *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, vol. 11, no. 4, pp. 1–8, 2016.

[18] G. Lukosevicius, A. R. Arreola, and A. S. Weddell, “Using Sleep States to Maximize the Active Time of Transient Computing Systems,” pp. 31–36, 2017.

[19] C. Pan, M. Xie, and J. Hu, “ENZYME: An energy-efficient transient computing paradigm for ultralow self-powered IoT edge devices,” *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, vol. 37, no. 11, pp. 2440–2450, 2018.

[20] A. Gomez, L. Benini, and L. Thiele, “Designing the Batteryless IoT.” Design, Automation and Test in Europe (DATE 2017) PhD Forum, Lausanne, Switzerland, pp. 1–2, 2017.

[21] A. Gomez, L. Sigrist, M. Magno, L. Benini, and L. Thiele, “Dynamic Energy Burst Scaling for Transiently Powered Systems,” pp. 349–354, 2016.

[22] A. Gomez, L. Sigrist, T. Schalch, L. Benini, and L. Thiele, “Wearable, energy-opportunistic vision sensing for walking speed estimation,” *SAS 2017 - 2017 IEEE Sensors Appl. Symp. Proc.*, 2017.

[23] D. Balsamo, A. S. Weddell, G. V. Merrett, B. M. Al-Hashimi, D. Brunelli, and L. Benini, “Hibernus: Sustaining Computation During Intermittent Supply for Energy-Harvesting Systems,” *IEEE Embed. Syst. Lett.*, vol. 7, no. 1, pp. 15–18, 2015.

[24] A. Rodriguez Arreola *et al.*, “Approaches to Transient Computing for Energy Harvesting Systems: A Quantitative Evaluation,” *Proc. 3rd Int. Work. Energy Harvest. Energy Neutral Sens. Syst. - ENSsys ’15*, pp. 3–8, 2015.

[25] H. Jayakumar, A. Raha, and V. Raghunathan, “QUICKRECALL: A low overhead HW/SW approach for enabling computations across power cycles in transiently powered computers,” *Proc. IEEE Int. Conf. VLSI Des.*, pp. 330–335, 2014.

[26] D. Balsamo *et al.*, “Hibernus++ : A Self-Calibrating and Adaptive System for Intermittently-Powered Embedded Devices,” *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, vol. 11, no. 4, pp. 1–8, 2012.

[27] Texas Instruments, “MSP MCU FRAM Utilities version 03.10.00.10 User ’ s Guide,” pp. 1–70, 2017.

[28] J. R. Piorno, C. Bergonzini, D. Atienza, and T. S. Rosing, “Prediction and management in energy harvested wireless sensor nodes,” *Proc. 2009 1st Int. Conf. Wirel. Commun. Veh. Technol. Inf. Theory Aerosp. Electron. Syst. Technol. Wirel. VITAE 2009*, pp. 6–10, 2009.

[29] D. K. Noh and K. Kang, “Balanced energy allocation scheme for a solar-powered sensor

system and its effects on network-wide performance," *J. Comput. Syst. Sci.*, vol. 77, no. 5, pp. 917–932, 2011.

[30] S. Kosunalp, "A New Energy Prediction Algorithm for Energy-Harvesting Wireless Sensor Networks With Q-Learning," *IEEE Access*, vol. 4, pp. 5755–5763, 2016.

[31] A. Cammarano, C. Petrioli, and D. Spenza, "Pro-Energy: A novel energy prediction model for solar and wind energy-harvesting wireless sensor networks," *MASS 2012 - 9th IEEE Int. Conf. Mob. Ad-Hoc Sens. Syst.*, pp. 75–83, 2012.

[32] F. Ahmed, G. Tamberg, Y. Le Moullec, and P. Annus, "Adaptive LINE-P: An adaptive linear energy prediction model for wireless sensor network nodes," *Sensors (Switzerland)*, vol. 18, no. 4, pp. 1–26, 2018.

[33] Muhammad, H. K. Qureshi, U. Saleem, M. Saleem, A. Pitsillides, and M. Lestas, "Harvested Energy Prediction Schemes for Wireless Sensor Networks: Performance Evaluation and Enhancements," *Wirel. Commun. Mob. Comput.*, vol. 2017, pp. 1–14, 2017.

[34] M. J. Ismail, R. Ibrahim, and I. Ismail, "Adaptive neural network prediction model for energy consumption," *ICCRD2011 - 2011 3rd Int. Conf. Comput. Res. Dev.*, vol. 4, pp. 109–113, 2011.

[35] M. Gao, Q. Wang, M. T. Arafin, Y. Lyu, and G. Qu, "Approximate computing for low power and security in the internet of things," *Computer (Long. Beach. Calif).*, vol. 50, no. 6, pp. 27–34, 2017.

[36] S. Mittal, "A Survey of Techniques for Approximate Computing," *ACM Comput. Surv.*, vol. 48, no. 4, pp. 1–33, 2016.

[37] T. Y. Yeh, P. Faloutsos, M. Ercegovac, S. J. Patel, and G. Reinman, "The art of deception: Adaptive precision reduction for area efficient physics acceleration," *Proc. Annu. Int. Symp. Microarchitecture, MICRO*, pp. 394–406, 2007.

[38] Y. Tian, Q. Zhang, T. Wang, F. Yuan, and Q. Xu, "{ApproxMA:} Approximate Memory Access for Dynamic Precision Scaling," *Proc. 25th Ed. Gt. Lakes Symp. {VLSI} - {GLSVLSI} '15*, pp. 337–342, 2015.

[39] Texas Instruments, "Introduction to SimpliciTI," pp. 1–14, 2010.

[40] D. Beniaguev, "Historical Hourly Weather Data 2012-2017," 2017. [Online]. Available: https://www.kaggle.com/selfishgene/historical-hourly-weather-data/metadata. [Accessed: 10-Mar-2019].

[41] M. A. Alswailim, H. S. Hassanein, and M. Zulkernine, "CRAWDAD dataset queensu/crowd_temperature (v. 2015-11-20): derived from roma/taxi (v. 2014-07-17)."

Appendix 1 – Source-Code

The complete source-code contains many files and hundreds of line of code, thus this appendix presents only extracts of the source-code, specifically some parts that have been modified or added for implementing the proposed techniques and application.

```
case COMPUTE :    //  Energy  is  available  for  the  least-energy
consuming task i-e Computations.
            // while(1)   //for debugging

            // {
                  computedone=1;
                  temp=readTemperature();
                  //updateDataMessage_approx()
                  //updateDataMessage();

#ifdef Data_Prediction

            percent_error=historical_data_length;
// percent_error here is just used as a flag and it is not actually an
error

                  updateDataMessagewithdp();

                  if(percent_error!=(historical_data_length-1))
                  {
                        computedone=0;
                        State = ENERGY_MONITORING;

                        break;

                  }

#endif

            updateDataMessage_approx();
```

```
                  //uart_send();

                        //for(j=0;j<100000;j++)  //added by SIK

                        //    temp=rand()*8.55;

            //   }

                  State = ENERGY_MONITORING;

                break;


void updateDataMessagewithdp()

{

      updateTime(&time);

      //vcc_level       =                                    (uint8_t)
computeEnergyLevel(getVoltageADC10());

      temp        = readTemperature();

      data_msg_send[0]  =     0xFF;

      data_msg_send[1]  =     temp;

      data_msg_send[2]  =     time.sec;

      data_msg_send[3]  =     time.min;

      data_msg_send[4]  =     time.hour;

      data_msg_send[5]  =     vcc_level;

      data_msg_send[6]  =     ++talker_tid;



#ifdef Data_Prediction

      data_predictioncall();

      data_msg_send[7]  =     data_prediction1;

//
```

```
      if(((historicaldata_predictions[64-1]-
historical_data[historical_data_length-1])<<2)                      &&
((historicaldata_predictions[64-2]-
historical_data[historical_data_length-2])<<2))

            {

            if ((temp-data_prediction1)<<2)

                  isCommunicating = 0;

                  SMPL_Unlink(sLinkID1);

                  SMPL_Ioctl(IOCTL_OBJ_RADIO,   IOCTL_ACT_RADIO_SLEEP,
0);   // Turn off the radio

                  radioAwake = 0;

            }

      historical_data_length++;

#endif

}

void updateDataMessage_approx()

{

            svalues.sbytedata = sbyte;

            sbit[0]=svalues.sbits.bit0;

            sbit[1]=svalues.sbits.bit1;

            sbit[2]=svalues.sbits.bit2;

            sbit[3]=svalues.sbits.bit3;

            sbit[4]=svalues.sbits.bit4;

            sbit[5]=svalues.sbits.bit5;

            sbit[6]=svalues.sbits.bit6;
            sbit[7]=svalues.sbits.bit7;

            k=0;

            for (j=1;j<8;j++) {

            if(msg_test[j]!=254)

            {

            svalues.sbytedata= msg_test[j];    // needs to be changed
based on the data you want to send

            svalues.sbits.bit0=sbit[j];

            msg_test[j]=svalues.sbytedata;

            }

            }

            sbyte=temparrayhc_added[kk+11];

            svalues.sbytedata = sbyte;

            sbit[0]=svalues.sbits.bit0;

            sbit[1]=svalues.sbits.bit1;

            sbit[2]=svalues.sbits.bit2;

            sbit[3]=svalues.sbits.bit3;

            sbit[4]=svalues.sbits.bit4;

            sbit[5]=svalues.sbits.bit5;

            sbit[6]=svalues.sbits.bit6;

            sbit[7]=svalues.sbits.bit7;

            for (j=1;j<8;j++) {

            if(msg_test[j]!=254)

            {
```

```
            svalues.sbytedata= msg_test[j];    // needs to be changed
based on the data you want to send

            svalues.sbits.bit1=sbit[j];

            msg_test[j]=svalues.sbytedata;

            }

            }

}

void data_predictioncall()

{

#ifdef Data_Prediction


      data_prediction1=(38)*(historical_data[historical_data_length-
1])+(94)*(historical_data[historical_data_length-
2])+(50)*(historical_data[historical_data_length-3])-
(38)*(historical_data[historical_data_length-4])-
(38)*(historical_data[historical_data_length-5])-
(6)*(historical_data[historical_data_length-6]);

      if(data_prediction1<=0) // To avoid negatives values

            data_prediction1=0;

      else

            data_prediction1=data_prediction1/100;    //  /!\  Loss  of
accuracy

      //test=64;        //testing

      push_data(data_prediction1,          historicaldata_predictions,
historical_data_length);

      //push_data(data_prediction1, historicaldata_predictions, test);

      nb_data_prediction ++;

      historical_data_length++;

      //test++;         //testing
      //percent_error_previous=percent_error_avg;

      //percent_error=(abs((data_prediction1-
historical_data[historical_data_length-
1]))*100)/historical_data[historical_data_length-1];

      if (percent_error_flag==0)

      {
            percent_error_avg=percent_error;

            percent_error_flag++;

      }
      else
      {
      percent_error_avg=(percent_error_previous+percent_error);

            percent_error_avg=(percent_error_avg/2);

      }
#endif

}


void savedata()

{

      //Received_data2[kk]=svalues.sbytedata;

      for (j=1;j<8;j++)

      {

            Received_data1[ii]=data_msg_receive[j];

            ii++;

            if(data_msg_receive[j]==254)

                  {

                  for (j=1;j<50;j++)
                  SPIN_ABOUT_A_SECOND;

                  //SMPL_Ioctl(                        IOCTL_OBJ_RADIO,
IOCTL_ACT_RADIO_SLEEP, 0);    // Turn off the radio

                  }

      }

      kk++;

      if(kk==11 || ii==70)

      {

       kk=0;

       ii=0;

      }

}

void decode_Packet()

{

            for (j=1;j<8;j++)

            {

            svalues.sbytedata= data_msg_receive[j];

            sbit[j]=svalues.sbits.bit0;

            //msg_test[j]=svalues.sbytedata;

            }

             for (j=1;j<8;j++)

             {

             svalues.sbytedata= data_msg_receive[j];


             sbit[j]=svalues.sbits.bit1;
```

```
        //msg_test[j]=svalues.sbytedata;

        }

        k=0;

        svalues.sbits.bit0=sbit[k];

        k++;

        svalues.sbits.bit1=sbit[k];

        k++;

        svalues.sbits.bit2=sbit[k];

        k++;

        svalues.sbits.bit3=sbit[k];

        k++;

        svalues.sbits.bit4=sbit[k];

        k++;

        svalues.sbits.bit5=sbit[k];

        k++;

        svalues.sbits.bit6=sbit[k];

        k++;

        svalues.sbits.bit7=sbit[k];

        k++;

        Received_data2[kk+11]=svalues.sbytedata;

}
```