

INTERNSHIP REPORT

WIRELESS SENSOR NETWORKS

01-06-2024 – 30-06-2024

Submitted By:- Vignesh.T.A

22BML0059

B.Tech in Electronics and Communications
Engineering with Specialization in Biomedical
Engineering

Under the Guidance of –

Smt. Jemimah Ebenezer

Head, Wireless Sensor Networking
Computer Division
Indira Gandhi Centre for Atomic Research Kalpakkam



VIT[®]
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

School of Electronics Engineering (SENSE)

Vellore Institute of Technology

Vellore, TamilNadu

TIMELINE

The Internship was
completed during the
time period of **30 days**
from **01-06-2024** to
30-06-2024 in IGCAR
– Indira Gandhi Center
for Atomic Research
Kalpakkam

Table of Contents

Chapter No.	Page No.
1. Introduction	
1.1 Indira Gandhi Centre for Atomic Research (IGCAR)	1
1.2 Electronics and Instrumentation Group (EIG)	1
1.3 Computer Division (CD)	1
1.4 Wireless Sensor Networking (WSN)	2
2. Wireless Communication	2
2.1 Methods	2
2.2 Pros and Cons	3
2.3 Wireless Sensor Network	4
2.4 IEEE Protocols	4
2.5 ZIGBEE	5
2.6 Frames	7
3. Microcontrollers	8
3.1 Types	8
3.2 Parts	9
3.3 Central Processing Unit	12
3.4 Universal Asynchronous Receiver Transmitter	13
3.5 Analog to Digital Converter	13
4. Cortex M0 / M0+	15
4.1 SAM D20	15
4.2 Variants	16
5. Experiment	17
5.1 XCTU Config	17
5.2 Code for Packet Loss Calculation	18
5.3 UART Packet transmission	18
5.4 Packet Received Log File	21
5.5 Packet Loss Analysis for 5, 10 Nodes	22
5.6 Conclusion	23
6.2 Acknowledgements	23

1. Introduction

1.1 Indira Gandhi Centre for Atomic Research (IGCAR):

Indira Gandhi Centre for Atomic Research [IGCAR], the second largest establishment of the Department of Atomic Energy next to Bhabha Atomic Research Centre, was set up at Kalpakkam in 1971 with the main objective of conducting broad based multidisciplinary programme of scientific research and advanced Engineering, directed towards the development of sodium cooled Fast Breeder Reactor [FBR] technology, in India. This is part of the second stage of Indian Atomic Energy Programme, which is aimed at preparing the country for utilization of the extensive Thorium reserves and providing means to meet the large demands of electrical energy in 21 st century. In meeting the objectives, a modest beginning was made by constructing a sodium cooled Fast Breeder Test Reactor [FBTR], with a nominal power of 40 MWt,. It is the first of its kind in the world to use Plutonium Uranium mixed carbide as a driver fuel. With the experience and expertise gained by the successful operation of FBTR, the Centre has embarked upon the design and construction of 500 MWe, Prototype Fast Breeder Reactor [PFBR]. The PFBR is under advanced stage of construction and commissioning by BHAVINI. IGCAR is headed by Shri C G Karhadkar, Distinguished Scientist

1.2 Electronics and Instrumentation Group (EIG)

Electronics and Instrumentation Group of IGCAR is responsible for design and development of indigenous technology in the areas of Electronic Instrumentation & Control systems for fast breeder reactors and reprocessing plants that include Development of Distributed Digital Control System, Safety Critical and safety related Systems, Safe & Secure PLC , Virtual Control Panel based Control Room, Full-scope Operator Training Simulator, 3D modelling, animation & visualization of FBR subsystems and VR walkthrough of structures, Cyber Security Management for IT & I&C systems. EIG is headed by Shri. N.Sridhar, Director, EIG, IGCAR

1.3 Computer Division (CD):

Computer Division is responsible for designing, building and maintaining state-of the-art high-performance supercomputing facility that continues to meet large scale computing and data-intensive requirements in multi-disciplinary domains and Implementation of IT-enabled Nuclear Knowledge Management system for Fast Reactors and associated domains, computational intelligence systems, cryptography, cyber security solutions, knowledge management and development. Computer Division is headed by Shri. R. Jehadeesan, Head, CD/EIG

1.4 Wireless Sensor Networking

Other major mandate of EIG/IGCAR includes Design and Development of advanced equipments and technology such as indigenous Wireless Sensor Networks for nuclear facilities. Wireless Sensor Network (WSN) is one of the recent technologies with bounteous potential to cater innumerable applications. This technology has been utilized at IGCAR to provide a continuous, fully automated data collection with centralized monitoring and storage facility for various applications in IGCAR. WSN is headed by Smt. Jemimah Ebenezer, SO/G, Head, WSN / EIG, IGCAR

2. Wireless Communication

2.1 Methods

Radio Waves: Utilized extensively in Wi-Fi networks, Bluetooth devices, and cellular communication (2G, 3G, 4G, 5G), radio waves operate in various frequencies and are capable of long-range transmission.

Infrared (IR): Often used for short-range communication, IR is employed in TV remote controls, some indoor wireless data transfer applications, and line-of-sight communication scenarios.

Microwaves: These higher-frequency waves are employed in satellite communication, radar systems, and point-to-point wireless links, offering both long-range and high-bandwidth capabilities.

Bluetooth: A short-range wireless technology ideal for connecting devices like smartphones, headphones, and peripherals within a limited area.

Near Field Communication (NFC): NFC enables close-range communication between devices, commonly used for contactless payments, access control, and data sharing.

Wi-Fi (Wireless Fidelity): This technology operates in the 2.4 GHz and 5 GHz frequency bands, providing high-speed internet access and local area network connectivity.

Cellular Networks: Mobile communication networks use a combination of radio waves and digital modulation techniques to provide voice and data services over large geographic areas.

Satellite Communication: Involves the use of satellites orbiting the Earth to relay signals between distant locations, facilitating global communication and broadcasting.

2.2 Pro and Cons:

Radio Communication:

Pros: Wide coverage area, Reliable for audio transmission

Cons: Limited data transmission capacity, Prone to interference

Microwave Communication:

Pros: High bandwidth, Supports long-distance communication

Cons: Requires line-of-sight, Susceptible to weather conditions

Infrared Communication:

Pros: Low interference, Secure due to line-of-sight requirement

Cons: Short range, Requires direct line-of-sight

Bluetooth:

Pros: Low power consumption, Easy pairing between devices

Cons: Limited range (typically up to 100 meters), Lower data transfer rates

Wi-Fi:

Pros: High data transfer rates, Wide availability and support for many devices

Cons: Can be prone to interference from other electronic devices, Security concerns

Cellular Networks:

Pros: Wide coverage area, High mobility support

Cons: Can be expensive, Signal strength varies with location and infrastructure

Zigbee:

Pros: Low power consumption, Good for low-data-rate applications like home automation

Cons: Limited range, Not suitable for high data rate applications

NFC (Near Field Communication):

Pros: Very low power consumption, High security for short-range communication

Cons: Very short range (typically up to 10 cm), Limited data transfer rates

Li-Fi (Light Fidelity):

Pros: Very high data rates, Secure as it is confined to illuminated area

Cons: Requires line-of-sight, Can be affected by ambient light conditions

Acoustic Communication:

Pros: Effective for underwater communication, Can penetrate through water where radio waves cannot

Cons: Low data transfer rates, Limited range and prone to distortion in noisy environments

2.3 Wireless Sensor Network

A Wireless Sensor Network (WSN) is a network comprised of small, autonomous devices equipped with sensors that collect and transmit data through wireless communication. These devices, known as nodes, are typically low-cost and energy-efficient, making them suitable for various applications such as environmental monitoring, healthcare systems, industrial automation, and smart infrastructure. WSNs are designed to operate in diverse and often challenging environments, where traditional wired networks are impractical or expensive to deploy. They play a crucial role in gathering real-time information, enabling remote monitoring and control, and facilitating data-driven decision-making processes across numerous fields. As technology advances, the potential of WSNs continues to expand, driving innovations that enhance efficiency, sustainability, and quality of life.

2.4 IEEE Protocols

IEEE 802.11 (Wi-Fi)

Description: A set of standards for wireless local area networks (WLANs).

IEEE 802.15 (Wireless Personal Area Networks)

Subsets:

802.15.1 (Bluetooth)

802.15.4 (Zigbee)

IEEE 802.16 (WiMAX)

Description: Standards for wireless metropolitan area networks (WMANs).

IEEE 802.20 (Mobile Broadband Wireless Access)

Description: Standards for mobile broadband services.

IEEE 802.22 (Wireless Regional Area Networks)

Description: Standards for wireless networks using TV white spaces.

IEEE 802.11ah (Wi-Fi HaLow)

Description: Low-power, long-range version of Wi-Fi, operating in sub-1 GHz bands.

IEEE 802.11ax (Wi-Fi 6)

Description: Next-generation Wi-Fi with improved efficiency and capacity.

IEEE 802.11ad/ay (WiGig)

Description: Standards for multi-gigabit wireless communications in the 60 GHz band.

2.5 ZIGBEE

Zigbee is a specification for a suite of high-level communication protocols using low-power digital radios. It is based on the IEEE 802.15.4 standard for low-rate wireless personal area networks.

Key Features

Low power Consumption:

Designed for devices with low power consumption, making it ideal for battery-operated devices.

Low Data Rate:

Typically supports data rates of up to 250 kbps, suitable for intermittent data transmission.

Range and Coverage:

Effective range typically extends from 10 to 3000 meters, depending on the environment, power output and obstructions.

Mesh Networking:

Supports mesh, star, and tree topologies, providing robust and reliable network structures that can cover large areas through multi-hop communication.

Scalability

Can support large networks with thousands of devices.

Security:

Implements 128-bit AES encryption for secure data transfer.

Applications

Home Automation:

Smart lighting, heating, and air conditioning systems.

Security systems and smart locks.

Industrial Automation:

Monitoring and control of machinery.

Energy management systems.

Healthcare:

Remote health monitoring and medical data transmission.

Smart Metering:

Utility meters for electricity, gas, and water.

Consumer Electronics:

Smart appliances and remote controls.

Environmental Monitoring:

Sensors for temperature, humidity, and air quality.

Advantages

Interoperability: Supports a wide range of devices and vendors.

Reliability: Mesh networking enhances reliability through redundant communication paths.

Cost-Effective: Low-cost modules and low power consumption reduce operational costs.

Limitations

Data Rate: Lower data rate compared to other protocols like Wi-Fi.

Range: Limited range compared to some other wireless technologies, although mesh networking helps mitigate this.

Zigbee is widely used in various industries due to its balance of low power consumption, reliability, and cost-effectiveness.

XBEE

XBee is a brand of radio modules made by Digi International that support various wireless communication protocols, including Zigbee, but also others such as 802.15.4, Wi-Fi, and LTE. These modules are known for their ease of use and flexibility in implementing wireless communication in various applications. Supports multiple protocols: Zigbee, 802.15.4, DigiMesh, Wi-Fi, and cellular (LTE).

2.6 Frames

In UART (Universal Asynchronous Receiver/Transmitter) communication, data is transmitted in units called frames. Each frame consists of several components that ensure the proper synchronization and integrity of the data. Here's a breakdown of the different parts of a UART frame:

Components of a UART Frame

Start Bit:

Purpose: Signals the beginning of a frame.

Details: A single bit with a value of 0 (low voltage level). The start bit alerts the receiver that a new data byte is coming.

Data Bits

Purpose: Contains the actual data being transmitted.

Details: Typically, 5 to 9 bits long (most commonly 8 bits). These bits represent the data byte.

Parity Bit (Optional):

Purpose: Provides a simple error-checking mechanism.

Details: Can be set to even, odd, or none. In even parity, the parity bit is set to 1 if the number of 1s in the data bits is odd, making the total number of 1s even. In odd parity, the parity bit is set to 1 if the number of 1s in the data bits is even, making the total number of 1s odd.

Stop Bits:

Purpose: Signals the end of the frame.

Details: Can be 1, 1.5, or 2 bits long. These bits have a value of 1 (high voltage level). They provide a brief period for the receiver to process the received byte and prepare for the next frame.

3. Microcontroller

3.1 Types of Microcontroller

8-bit Microcontrollers

Description: Process data in 8-bit chunks, meaning the CPU's internal data paths, registers, and most instructions are designed to operate on 8-bit wide data.

Examples: 8051: A classic microcontroller architecture widely used in embedded systems.

AVR (e.g., ATmega328): Popular in Arduino boards.

PIC16: A family of microcontrollers from Microchip Technology.

Applications: Simple control tasks, home automation, basic IoT devices, educational purposes.

2. 16-bit Microcontrollers

Description: Process data in 16-bit chunks, providing a balance between performance and resource usage.

Examples:

MSP430: From Texas Instruments, known for low power consumption.

PIC24: A family of microcontrollers from Microchip Technology.

Applications: Industrial control, automotive applications, advanced IoT devices, medical instruments.

3. 32-bit Microcontrollers

Description: Process data in 32-bit chunks, offering higher performance suitable for more complex applications.

Examples:

ARM Cortex-M: A widely used family of microcontrollers with various models (e.g., Cortex-M0, M3, M4, M7).

ESP32: From Espressif Systems, known for integrated Wi-Fi and Bluetooth.

PIC32: From Microchip Technology, providing high performance.

Applications: Advanced automotive systems, robotics, complex IoT devices, multimedia applications.

4. DSP Microcontrollers (Digital Signal Processing)

Description: Combine microcontroller features with digital signal processing capabilities, allowing for real-time signal processing.

Examples:

dsPIC: From Microchip Technology, combining DSP and microcontroller features.

C2000: From Texas Instruments, designed for real-time control applications.

Applications: Audio and speech processing, motor control, real-time data processing, telecommunications.

5. FPGA-based Microcontrollers

Description: Integrate microcontroller functionalities with field-programmable gate array (FPGA) capabilities, allowing for custom hardware configurations.

Examples:

Microsemi SmartFusion: Combines an ARM Cortex-M3 processor with FPGA fabric.

Xilinx Zynq: Combines ARM Cortex-A9 processors with FPGA fabric.

Applications: Custom hardware applications, advanced digital signal processing, high-performance computing.

3.2 Parts of Microcontroller

1. Central Processing Unit (CPU)

Description: The brain of the microcontroller, responsible for executing instructions from the program memory.

Function: Performs arithmetic and logic operations, controls data flow, and coordinates the activities of other components.

2. Memory

Types:

Flash Memory: Non-volatile memory used to store the firmware or program code. Retains data even when power is off.

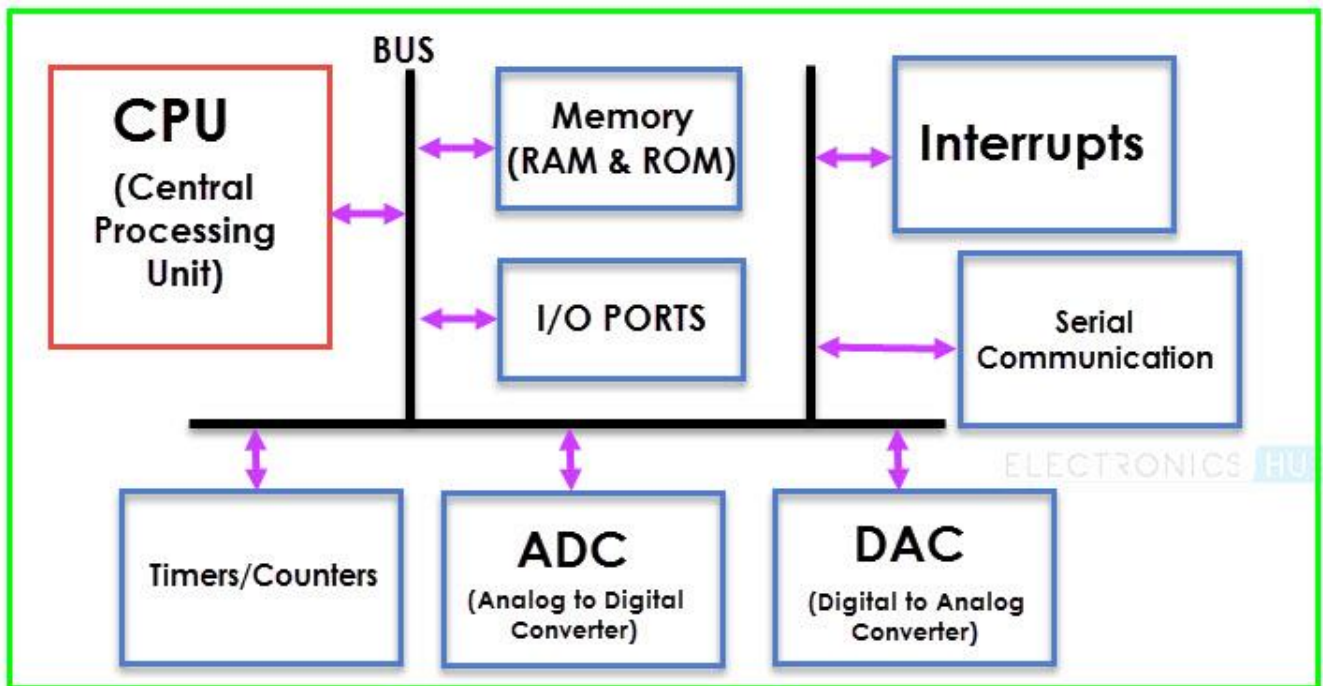
RAM (Random Access Memory): Volatile memory used for temporary data storage during program execution.

EEPROM (Electrically Erasable Programmable Read-Only Memory): Non-volatile memory used for storing small amounts of data that must be saved when power is removed.

3. Input/Output Ports (I/O Ports)

Description: Interfaces through which the microcontroller communicates with external devices.

Function: Allow the microcontroller to read data from sensors, switches, and other input devices, and to control actuators, LEDs, displays, and other output devices.



4. Timers/Counters

Description: Internal hardware modules that measure time intervals or count events.

Function: Used for generating precise time delays, measuring time intervals, and counting external events.

5. Analog-to-Digital Converter (ADC)

Description: Converts analog signals (e.g., from sensors) into digital values that the CPU can process.

Function: Enables the microcontroller to interface with analog devices.

6. Digital-to-Analog Converter (DAC)

Description: Converts digital values into analog signals.

Function: Allows the microcontroller to output analog signals to devices such as speakers and motors.

7. Serial Communication Interfaces

Types:

UART (Universal Asynchronous Receiver/Transmitter): Used for serial communication with devices like computers and other microcontrollers.

SPI (Serial Peripheral Interface): Used for high-speed communication between the microcontroller and peripherals like sensors, SD cards, and displays.

I2C (Inter-Integrated Circuit): Used for communication with multiple peripherals using fewer pins.

8. Interrupt Controller

Description: Manages the handling of interrupt signals.

Function: Allows the microcontroller to respond to external or internal events (interrupts) immediately, temporarily halting the main program to execute an interrupt service routine (ISR).

9. Oscillator/Clock

Description: Provides the clock signal that synchronizes the operation of the microcontroller.

Function: Determines the speed at which the microcontroller executes instructions.

10. Power Supply and Reset Circuit

Power Supply: Provides the necessary power for the microcontroller to operate.

Reset Circuit: Resets the microcontroller to a known state, either on power-up or in response to an external reset signal.

11. Watchdog Timer

Description: A timer that resets the microcontroller if the program execution fails or hangs.

Function: Enhances system reliability by automatically recovering from software errors.

12. Programmable I/O Ports

Description: General-purpose input/output pins that can be configured by software.

Function: Allows flexible interfacing with a variety of external devices.

These components work together to enable a microcontroller to perform it

3.3 Central Processing Unit

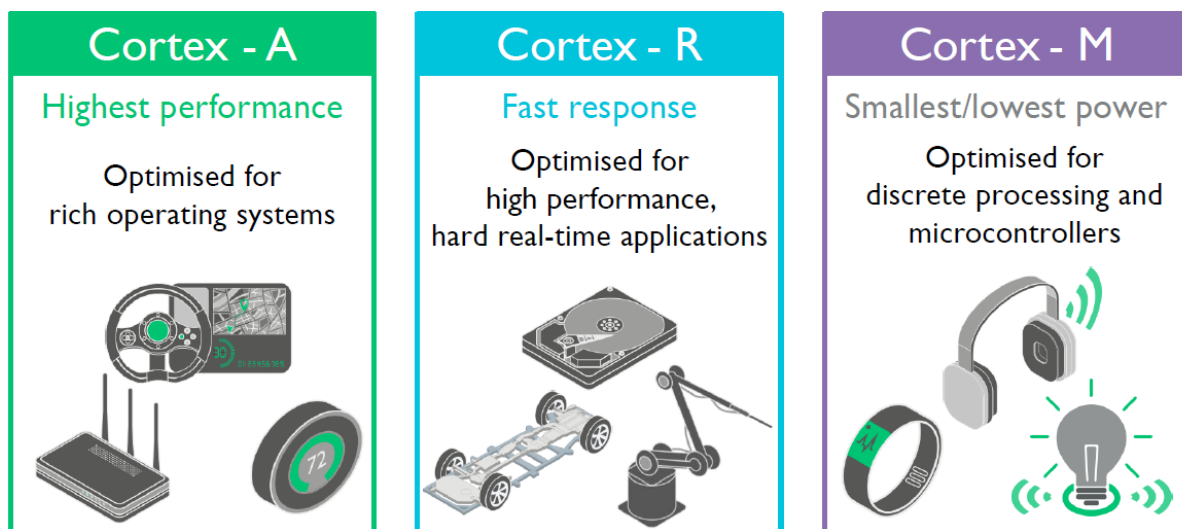
Architecture

CISC	RISC
Emphasis on hardware	Emphasis on software
Multiple instruction sizes and formats	Instructions of same set with few formats
Less registers	Uses more registers
More addressing modes	Fewer addressing modes
Extensive use of microprogramming	Complexity in compiler
Instructions take a varying amount of cycle time	Instructions take one cycle time
Pipelining is difficult	Pipelining is easy

Types

The ARM Cortex-M is a group of 32-bit RISC ARM processor cores licensed by ARM Limited. These cores are optimized for low-cost and energy-efficient integrated circuits, which have been embedded in tens of billions of consumer devices.

ARM Architecture: For Diverse Embedded Processing Needs



3.4 Universal Asynchronous Receiver/Transmitter

UART (Universal Asynchronous Receiver/Transmitter) is a hardware communication protocol used for asynchronous serial communication between devices. It is widely used in embedded systems and microcontroller applications due to its simplicity and effectiveness for short-distance communication.

Key Features

Asynchronous Communication:

No need for a shared clock signal between the transmitting and receiving devices.

Full-Duplex:

Supports simultaneous two-way communication.

Data Framing:

Data is transmitted in frames consisting of a start bit, data bits, an optional parity bit, and one or more stop bits.

Configurable Baud Rate:

The rate at which data is transmitted can be configured, typically ranging from 1,200 to 115,200 bps (bits per second) or higher.

Error Detection:

Optional parity bit for basic error detection.

3.5 Analog to Digital Converters

An Analog-to-Digital Converter (ADC) is a device that converts an analog signal, such as a sound picked up by a microphone or light entering a digital camera, into a digital signal. This process is essential for digital devices to process and store analog signals. There are several types of ADCs, each with its own advantages and applications. Here are the main types:

1. Successive Approximation Register (SAR) ADC

Description: Uses a binary search algorithm to convert the analog signal to digital. It approximates the input signal with a digital value in a step-by-step manner.

Applications: Widely used in microcontrollers, data acquisition systems, and digital oscilloscopes.

2. Flash ADC (Parallel ADC)

Description: Uses a bank of comparators to compare the input signal with reference voltages. It provides a digital output in a single step.

Applications: High-speed applications such as radar systems, digital oscilloscopes, and high-frequency data acquisition.

3. Delta-Sigma ($\Delta\Sigma$) ADC

Description: Uses oversampling and noise shaping techniques to convert the analog signal into a digital output. It integrates the signal over time, resulting in high resolution.

Applications: Audio and precision measurement applications, such as digital audio converters, and high-resolution sensors.

4. Dual Slope ADC

Description: Converts the input signal by integrating it over a fixed period, then de-integrating using a known reference. The time taken for de-integration is measured and converted to digital.

Applications: Digital voltmeters, instrumentation, and applications requiring high accuracy.

5. Pipeline ADC

Description: Combines multiple stages of low-resolution ADCs to achieve a high-resolution output. Each stage processes a portion of the signal and passes the residual to the next stage.

Applications: High-speed and high-resolution applications, such as video processing, communication systems, and medical imaging.

6. Sigma-Delta ($\Sigma\Delta$) ADC

Description: Similar to Delta-Sigma ADCs, it uses oversampling and filtering to achieve high resolution. The difference is mainly in implementation details.

Applications: High-precision applications, such as instrumentation and audio applications.

7. Integrating ADC

Description: Measures the average value of the input signal over a period of time by integrating the input signal and then converting the integration result to a digital value.

Applications: Digital voltmeters and other measurement instruments where noise rejection is crucial.

4. Cortex M0/M0+

4.1 SAM D20

The SAMD20 series is a family of microcontrollers developed by Microchip Technology, which leverages the ARM® Cortex®-M0+ core. This microcontroller series is designed to deliver a low-power, high-performance solution suitable for a broad range of applications, from consumer electronics to industrial control systems

Key Features

Core and Performance:

ARM Cortex-M0+ Core: The SAMD20 microcontrollers are built around the ARM Cortex-M0+ core, which is optimized for low power consumption and high efficiency.

Operating Frequency: It can operate at speeds up to 48 MHz, providing ample processing power for various tasks.

Low Power Consumption: With advanced power management features, the SAMD20 series is ideal for battery-powered and energy-sensitive applications.

Memory:

Flash Memory: The SAMD20 devices offer up to 256 KB of Flash memory, which is used for storing application code.

SRAM: They also include up to 32 KB of SRAM for dynamic data storage.

Peripheral Features:

UART, I2C, and SPI: The SAMD20 series includes multiple communication interfaces such as UART, I2C, and SPI, facilitating connectivity with various peripherals and other microcontrollers.

ADC and DAC: Integrated analog-to-digital converters (ADC) and digital-to-analog converters (DAC) support a wide range of analog signal processing needs.

Timers and PWM: Multiple timer and PWM (Pulse Width Modulation) channels are available for tasks requiring precise timing control.

USB Support: Some models in the SAMD20 family come with USB support, allowing for easy connection to USB devices.\

4.2 Variants:

The SAM D20 is a series of low-power microcontrollers from Microchip Technology (previously Atmel) based on the ARM Cortex-M0+ core. It is designed for a variety of applications requiring low power consumption and high performance. Let's delve into the SAM D20 variants, and XCTU software related to XBee modules.

SAM D20 Variants

The SAM D20 series includes various models differentiated by features such as memory size, package type, and the number of peripherals. Key variants include:

Memory Size:

Flash memory ranges from 16 KB to 256 KB.

SRAM ranges from 2 KB to 32 KB.

Package Types:

Available in various packages like QFN, TQFP, and BGA, catering to different space and design requirements.

Peripherals:

USART/UART: Up to 6.

I2C (SERCOM): Up to 6.

SPI (SERCOM): Up to 6.

Timers: Various configurations including 16-bit and 32-bit timers.

ADC: Up to 12-bit, 350 ksps.

DAC: Up to 10-bit.

PWM: Multiple channels.

Capacitive Touch: Integrated support for capacitive touch sensing.

Notable SAM D20 Models

SAM D20E: More general-purpose variants with a balanced set of peripherals.

SAM D20G: Higher pin count versions for more I/O options.

SAM D20J: Highest pin count and maximum peripheral integration.

Key Features

Low Power Consumption: Suitable for battery-operated devices.

Flexible Clock System: Multiple clock sources and clock gating for power saving.

Event System: Allows peripherals to communicate directly without CPU intervention.

Integrated Analog Modules: ADC, DAC, and analog comparators for signal processing.

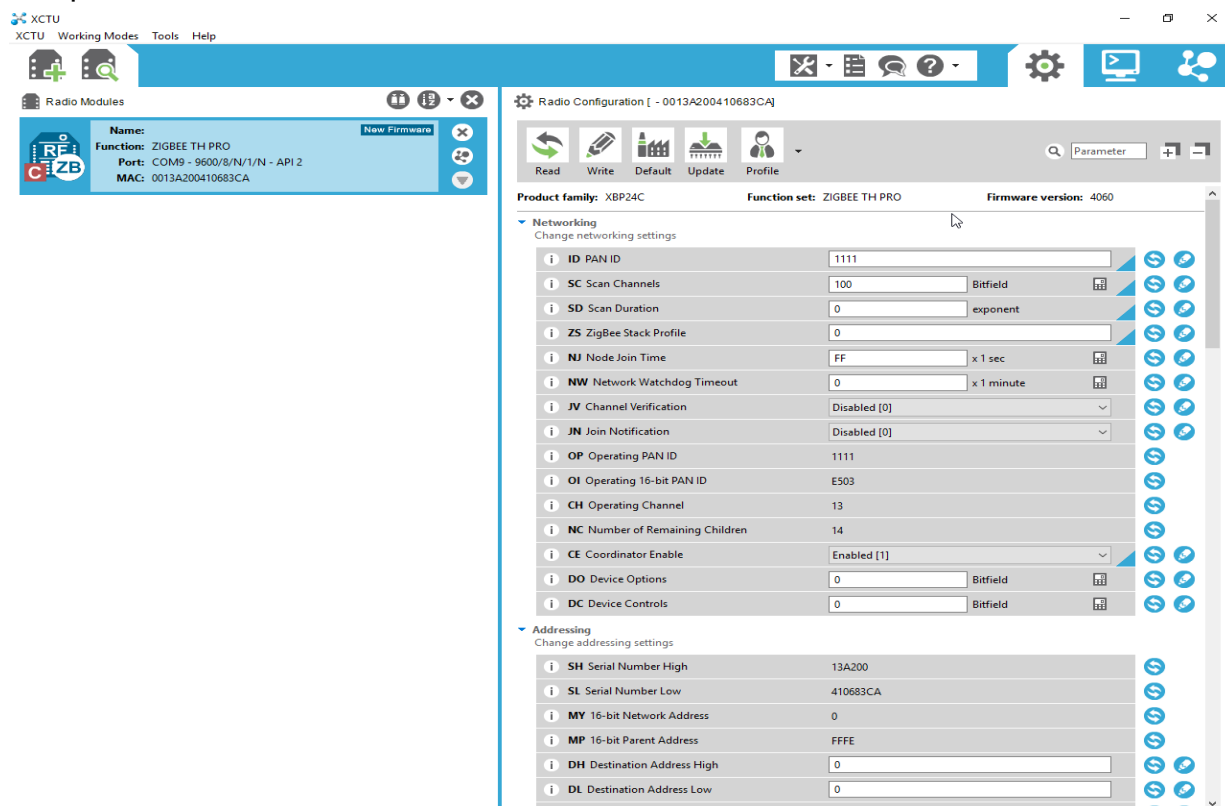
5. Experiment

5.1 XCTU Config

XCTU is a configuration and testing software tool developed by Digi International for use with XBee modules. It provides a graphical interface for configuring and testing the communication settings of XBee RF modules.

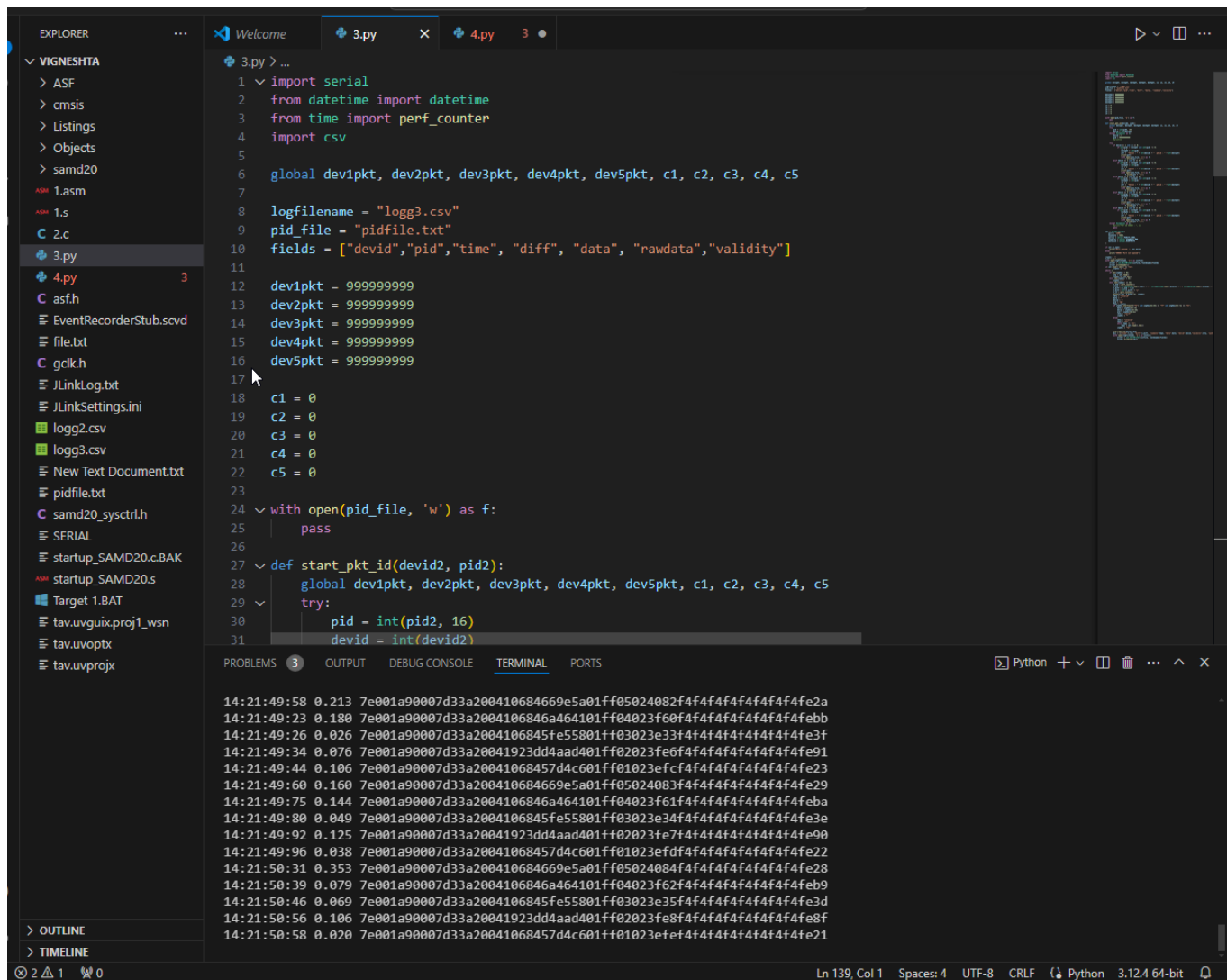
Here channel 8 was selected for transmission of packets through ZIGBEE protocol. PAN ID of the network was chosen to be 1111 along with an encryption with a passkey. Transparent mode of transmission was used for debugging purposes and later switched to API mode of transmission.

One XBEE module was configured to be a controller module and the rest of the modules were configured to be as router nodes. The controller node receives the data from all the router nodes and the data is logged in the serial port of the module using a python code discussed in the next topic.



5.2 Code for Packet Loss Calculation:

This python code is used to log the received data from the controller XBEE node to a log file in the .csv format. Also, this code performs calculations of total packets sent and total packet received and hence calculated the percentage of packet loss on each node. The packets are sent with embedded device ID, packet number and test data to perform this analysis. This code also checks for packet integrity with a checksum and classifies packets and assigns validity. Finally outputs the processed data to another csv file.



```
1 import serial
2 from datetime import datetime
3 from time import perf_counter
4 import csv
5
6 global dev1pkt, dev2pkt, dev3pkt, dev4pkt, dev5pkt, c1, c2, c3, c4, c5
7
8 logfilename = "logg3.csv"
9 pid_file = "pidfile.txt"
10 fields = ["devid", "pid", "time", "diff", "data", "rawdata", "validity"]
11
12 dev1pkt = 999999999
13 dev2pkt = 999999999
14 dev3pkt = 999999999
15 dev4pkt = 999999999
16 dev5pkt = 999999999
17
18 c1 = 0
19 c2 = 0
20 c3 = 0
21 c4 = 0
22 c5 = 0
23
24 with open(pid_file, 'w') as f:
25     pass
26
27 def start_pkt_id(devid2, pid2):
28     global dev1pkt, dev2pkt, dev3pkt, dev4pkt, dev5pkt, c1, c2, c3, c4, c5
29     try:
30         pid = int(pid2, 16)
31         devid = int(devid2)
```

```
14:21:49:58 0.213 7e001a90007d33a200410684669e5a01ff05024082f4f4f4f4f4f4fe2a
14:21:49:23 0.180 7e001a90007d33a2004106846a464101ff04023f60f4f4f4f4f4f4febb
14:21:49:26 0.026 7e001a90007d33a2004106845fe55801ff03023e33f4f4f4f4f4f4fe3f
14:21:49:34 0.076 7e001a90007d33a20041923dd4aad401ff02023fe6f4f4f4f4f4f4fe01
14:21:49:44 0.106 7e001a90007d33a20041068457d4c601ff01023efcf4f4f4f4f4f4fe23
14:21:49:00 0.160 7e001a90007d33a200410684669e5a01ff05024083f4f4f4f4f4f4fe29
14:21:49:75 0.144 7e001a90007d33a2004106846a464101ff04023f61f4f4f4f4f4f4feba
14:21:49:80 0.049 7e001a90007d33a2004106845fe55801ff03023e34f4f4f4f4f4fe3e
14:21:49:92 0.125 7e001a90007d33a20041923dd4aad401ff02023fe7f4f4f4f4f4f4fe90
14:21:49:96 0.038 7e001a90007d33a20041068457d4c601ff01023efdf4f4f4f4f4f4fe22
14:21:50:31 0.353 7e001a90007d33a200410684669e5a01ff05024084f4f4f4f4f4fe28
14:21:50:39 0.079 7e001a90007d33a2004106846a464101ff04023f62f4f4f4f4f4f4feb9
14:21:50:46 0.069 7e001a90007d33a2004106845fe55801ff03023e35f4f4f4f4f4f4fe3d
14:21:50:56 0.106 7e001a90007d33a20041923dd4aad401ff02023fe8f4f4f4f4f4f4fe8f
14:21:50:58 0.020 7e001a90007d33a20041068457d4c601ff01023efef4f4f4f4f4f4fe21
```

5.3 UART Packet Transmission

For transmitting the packets from the router node, each router node is programmed with the following code. This code is written in c using Keil software. First the data is prepared by appending headers and footers required by the API transmission of frames. Then the data is arranged accordingly to properly construct a frame. Then the packet is sent through the serial port

to the XBEE module attached to the microcontroller externally. The XBEE module then transmits the data to re controller node where the received data is processed by the above python code.

Code:

```
#include "samd20_sysctrl.h"
#include "gclk.h"
static void gclk_init(void);
static void gclk_config_main_clk(int clk_src);

int main(void) {

    uint64_t devid = 0x0a;
    uint64_t sbit = 0xff;
    uint64_t ebit = 0xfe;
    uint64_t dta = 0xf4f4f4f4f4f4f4f4;
    uint64_t pktid = 0x000000;

    uint64_t frame1 = 0x7E001C1001000000; //7E 00 1C 10 01 00 00 00
    uint64_t frame2 = 0x0000000000fffe00; //00 00 00 00 00 FF FE 00
    uint64_t frame3 = 0x0000000000000000; //00 FF 01 01 00 00 F4 F4
    uint64_t frame4 = 0x0000000000000000; //F4 F4 F4 F4 F4 F4 FE 52

    uint64_t devid64 = devid << 40;
    uint64_t sbit64 = sbit << 48;
    uint64_t ebit64 = ebit << 8;
    uint64_t pktid64 = pktid << 16;
    frame3 = frame3 + devid64 + sbit64 + pktid64 + ((dta & 0xffff000000000000) >> 48);
    frame4 = frame4 + ebit64 + ((dta & 0x0000ffffffffffff) << 16);

    //devid64 = 0x0000ff0000000000 f3
    //pktid64 = 0x000000ffffffff0000 f3
    //sbit64 = 0x00ff000000000000 f3
    //ebit64 = 0x00000000000000ff f4
    //dta64 = 0xffffffffffff0000 f4 000000000000ffff f3
    //csum64 = 0x00000000000000ff f4

    gclk_init();
    gclk_config_main_clk(GCLK_GENCTRL_SRC_OSC8M_Val);
    PORT->Group[1].DIRSET.reg = 0x1000;
    //PORT->Group[1].OUTSET.reg = 0x1000;

    PM->APBCMASK.reg |= (1U << 2); // PM
    while(GCLK->STATUS.bit.SYNCBUSY){};

    *((int *)0x42000800U) |= 0x1; // SERCOM0 SWRST
    while(((int *)0x42000800U) & 1){
        /* The module is busy resetting itself */
    }
    while(((int *)0x42000800U) & 2){
        /* enable */
    }
    while(GCLK->STATUS.reg & GCLK_STATUS_SYNCBUSY){};

    GCLK->CLKCTRL.reg = 0x400D;

    while(GCLK->STATUS.bit.SYNCBUSY & GCLK_STATUS_SYNCBUSY){};
    PORT->Group[0].PINCFG[4].bit.PMUXEN = 1;
    PORT->Group[0].PINCFG[5].reg = 0x3;
    PORT->Group[0].PMUX[2].reg = 0x33; // PORT

    while(SERCOM0->USART.STATUS.bit.SYNCBUSY | GCLK->STATUS.bit.SYNCBUSY){};
    SERCOM0->USART.CTRLA.reg = 0x40100004; // SERCOM0 CTRLA
    while(SERCOM0->USART.STATUS.bit.SYNCBUSY | GCLK->STATUS.bit.SYNCBUSY){};
    SERCOM0->USART.BAUD.reg = 65536 * (1 - (16 * 9600.0 / 1000000.0));
    SERCOM0->USART.CTRLB.reg = 0x30000; // CTRLB
    while(SERCOM0->USART.STATUS.bit.SYNCBUSY | GCLK->STATUS.bit.SYNCBUSY){};
    SERCOM0->USART.CTRLA.reg |= 0x2; // SERCOM0 CTRLA
    while(SERCOM0->USART.STATUS.bit.SYNCBUSY | GCLK->STATUS.bit.SYNCBUSY){};

    uint64_t pktid8 = 0x00;
    while (1) {
        //if(SERCOM0->USART.INTFLAG.bit.RXS){
        //    PORT->Group[1].OUTSET.reg = 0x1000;
        //    while(!SERCOM0->USART.INTFLAG.bit.RXC){};
        //    while(SERCOM0->USART.STATUS.bit.SYNCBUSY | GCLK->STATUS.bit.SYNCBUSY){}; // SYNCBUSY
        //    dd = SERCOM0->USART.DATA.reg;
        //    while(SERCOM0->USART.STATUS.bit.SYNCBUSY | GCLK->STATUS.bit.SYNCBUSY){}; // SYNCBUSY
        //    PORT->Group[1].OUTCLR.reg = 0x1000;
        //}
```

```

        if(pktid == 0x0 & pktid8 < 0xfffff){
            pktid8 = pktid8 + 0x1;
            frame3 = frame3 & 0xfffff000000ffff;
            frame3 = frame3 + (pktid8 << 16);
        }

        uint64_t csum = 0x0;
        uint64_t g = frame3 & 0x00ffffffffffff;
        frame4 = frame4 & 0xffffffffffff00;
        for(int i=0;i<8;i++){
            csum = (csum + (g & 0xff)) & 0xff;
            g = g >> 8;
        }
        g = frame4 & 0xffffffffffff00;
        for(int i=0;i<8;i++){
            csum = (csum + (g & 0xff)) & 0xff;
            g = g >> 8;
        }

        g = frame2;
        for(int i=0;i<8;i++){
            csum = (csum + (g & 0xff)) & 0xff;
            g = g >> 8;
        }

        g = frame1 & 0x000000ffffffff;
        for(int i=0;i<8;i++){
            csum = (csum + (g & 0xff)) & 0xff;
            g = g >> 8;
        }

        csum = (0xff - csum) & 0xff;
        frame4 = frame4 + csum;

        PORT->Group[1].OUTTGL.reg = 0x1000;

        //printf("%"PRIx64 "\n", csum);

        uint64_t h = 0x0;
        uint64_t j = frame1;
        for(int i = 0;i<8;i++){
            h = (j & 0xff00000000000000) >> 56;
            j = j << 8;
            SERCOM0->USART.DATA.reg = h;
            while(!SERCOM0->USART.INTFLAG.bit.TXC){};
        }

        j = frame2;
        for(int i = 0;i<8;i++){
            h = (j & 0xff00000000000000) >> 56;
            j = j << 8;
            SERCOM0->USART.DATA.reg = h;
            while(!SERCOM0->USART.INTFLAG.bit.TXC){};
        }

        j = frame3;
        for(int i = 0;i<8;i++){
            h = (j & 0xff00000000000000) >> 56;
            j = j << 8;
            SERCOM0->USART.DATA.reg = h;
            while(!SERCOM0->USART.INTFLAG.bit.TXC){};
        }

        j = frame4;
        for(int i = 0;i<8;i++){
            h = (j & 0xff00000000000000) >> 56;
            j = j << 8;
            SERCOM0->USART.DATA.reg = h;
            while(!SERCOM0->USART.INTFLAG.bit.TXC){};
        }

        for(int i = 99999;i!=0;i--){
            int wkejnweijvn = 100;
        }

        while(SERCOM0->USART.STATUS.bit.SYNCBUSY | GCLK->STATUS.bit.SYNCBUSY){}; // SYNCBUSY
    }
}

static void gclk_init(void) {
    /* Enable the APB/APBA clock - where the GCLK uses this clock */
    PM->APBAMASK.reg |= PM_APBAMASK_GCLK;

    /* Software reset the module to ensure it is re-initialized correctly */
    GCLK->CTRL.reg = GCLK_CTRL_SWRST;

    while(GCLK->CTRL.reg & GCLK_CTRL_SWRST){
        /* wait for reset to complete */
    }
}

static void gclk_config_main_clk(int clk_src) {
    while((GCLK->STATUS.reg & GCLK_STATUS_SYNCBUSY));
    /* Write the Divisor value to the register */
    GCLK->GENDIV.reg = GCLK_GENDIV_DIV(0);
    /* *((uint8_t*)&GCLK->GENDIV.reg) = GCLK_GENDIV_DIV(0); // |GCLK_GENDIV_ID(gclk_id);

    /* Wait for synchronization */
    while((GCLK->STATUS.reg & GCLK_STATUS_SYNCBUSY));
    GCLK->GENCTRL.reg = (GCLK->GENCTRL.reg & GCLK_GENCTRL_GENEN) | \
        GCLK_GENCTRL_ID(0) | GCLK_GENCTRL_RUNSTDBY | \
        GCLK_GENCTRL_SRC(clk_src);

    /* Wait for synchronization */
    while((GCLK->STATUS.reg & GCLK_STATUS_SYNCBUSY));
}

```

5.4 Packet Received Log file:

	A	B	C	D	E	F	G
1	devid	pid	time	diff	data	rawdata	validity
2							
3		0	17:49:4:11	0.30213999999978114	001a90007		invalid
4							
5	2	8994	17:49:4:18	0.06388459	f4f4f4f4f4f4	7e001a900	valid
6							
7	3	8938	17:49:4:22	0.04147690	f4f4f4f4f4f4	7e001a900	valid
8							
9	1	8934	17:49:4:26	0.03996909	f4f4f4f4f4f4	7e001a900	valid
10							
11	4	8986	17:49:4:53	0.26984649	f4f4f4f4f4f4	7e001a900	valid
12							
13	5	8974	17:49:4:67	0.14396400	f4f4f4f4f4f4	7e001a900	valid
14							
15	1	8935	17:49:4:71	0.03862669	f4f4f4f4f4f4	7e001a900	valid
16							
17	2	8995	17:49:4:74	0.02528220	f4f4f4f4f4f4	7e001a900	valid
18							
19	3	8939	17:49:4:77	0.03210950	f4f4f4f4f4f4	7e001a900	valid
20							
21	4	8987	17:49:5:61	0.28779240	f4f4f4f4f4f4	7e001a900	valid
22							
23	5	8975	17:49:5:22	0.15985689	f4f4f4f4f4f4	7e001a900	valid
24							
25	1	8936	17:49:5:25	0.03163869	f4f4f4f4f4f4	7e001a900	valid
26							
27	2	8996	17:49:5:29	0.03878090	f4f4f4f4f4f4	7e001a900	valid
28							
29	3	8940	17:49:5:44	0.15012830	f4f4f4f4f4f4	7e001a900	valid
30							
31	4	8988	17:49:5:60	0.16220289	f4f4f4f4f4f4	7e001a900	valid
32							
33	5	8976	17:49:5:74	0.14385039	f4f4f4f4f4f4	7e001a900	valid
34							
35	1	8937	17:49:5:78	0.03613000	f4f4f4f4f4f4	7e001a900	valid
36							
37	2	8997	17:49:5:81	0.03034430	f4f4f4f4f4f4	7e001a900	valid
38							
39	3	8941	17:49:5:88	0.06828840	f4f4f4f4f4f4	7e001a900	valid
40							
41	4	8989	17:49:6:13	0.24841050	f4f4f4f4f4f4	7e001a900	valid
42							
43		0	17:49:6:28	0.15434509999885915	7e001a900		invalid
44							
45	1	8938	17:49:6:31	0.03175430	f4f4f4f4f4f4	7e001a900	valid
46							
47	2	8998	17:49:6:36	0.04413829	f4f4f4f4f4f4	7e001a900	valid
48							
49	3	8942	17:49:6:41	0.05025610	f4f4f4f4f4f4	7e001a900	valid
50							
51	4	8990	17:49:6:67	0.26223079	f4f4f4f4f4f4	7e001a900	valid
52							

5.5 Packet Loss Analysis 5 Node

	A	B	C	D
1	Device ID	Total Pkts	Pkts Recive	Loss %
2				
3	1	1363	1303	0.985979
4				
5	2	1365	1314	0.992637
6				
7	3	1362	1303	0.996681
8				
9	4	1365	1312	0.991172
10				
11	5	1368	1310	0.997602

Packet Loss Analysis 10 Node

	A	B	C	D
1	Device ID	Total Pkts	Pkts Reciv	Loss %
2				
3	1	1363	1303	0.955979
4				
5	2	1365	1314	0.962637
6				
7	3	1362	1303	0.956681
8				
9	4	1365	1312	0.961172
10				
11	5	1368	1310	0.957602
12				
13	6	1166	1113	0.954545
14				
15	7	1115	1068	0.957848
16				
17	8	1365	1066	0.780952
18				
19	9	1362	1061	0.779001
20				
21	10	1366	1310	0.959004

5.6 Conclusion

The UART transmission using the SAMD20 Cortex M0+ microcontroller and XBee module was successfully tested with configurations of 10 and 5 nodes. Throughout the experiments, the packet loss percentage was recorded and analysed. The findings indicate that the performance of the UART transmission system varied based on the number of nodes in operation.

For the setup with 10 nodes, the packet loss percentage was observed to be 0.9258%, suggesting that the system can handle a moderate level of network traffic but may face challenges with higher node counts and higher volume of data transmission. On the other hand, the configuration with 5 nodes exhibited a significantly lower packet loss percentage of 0.9948%, proving that lower data rates and lower number of transmitting nodes leads to higher efficiency in data transfer and lower packet loss.

These results highlight the importance of network size in the performance of UART-based communication systems. The data suggests that while the SAMD20 Cortex M0+ and XBee module provide a robust platform for small to medium-sized networks, considerations must be made for potential packet loss in larger networks. Future work may involve optimizing the system to further reduce packet loss, improve overall communication reliability and packet integrity.

6. Acknowledgements

- (1) I wish to acknowledge Shri. C G Karhadkar, Distinguished Scientist, Director, IGCAR for granting me the permission to undertake this internship in the esteemed organisation IGCAR.
- (2) I wish to acknowledge Shri. N. Sridhar, Director, EIG for permitting me in the group to undertake my internship at this core group.
- (3) I wish to acknowledge Shri. R. Jehadeesan, Head, Computer Division, EIG for allowing me to undertake my internship in the Division.
- (4) I wish to acknowledge Smt. Jemimah Ebenezer, Head, WSN for her valuable technical guidance and efforts for my successful internship in Wireless Communication area in Wireless Sensor Networking Section.
- (5) I also wish to thank Shri. T.S. Shri Krishnan, Scientific Officer, WSN for his valuable technical inputs and guidance during my internship with Wireless Communication.