



**UNIVERSITY OF NAIROBI**

**DEVELOPMENT OF A LOW-COST AUTOMATED MICROCONTROLLER  
BASED WIRELESS GAMMA COLUMN SCANNER PROTOTYPE**

BY

HENRY KIOKO MUTINDA

B.Sc. (Hons), Elec. Eng.

S56/12719/2018

A Thesis Submitted in Partial Fulfillment of the Requirements for the Award of the  
Degree of Master of Science in Nuclear Science of the University of Nairobi.

© August, 2022

## DECLARATION

I declare that this thesis is my original work and has not been submitted elsewhere for examination, award of a degree or publication. Where other people's work or my own work has been used, this has properly been acknowledged and referenced in accordance with the University of Nairobi's requirements.


**Henry Kioko Mutinda**

**Reg. No.: S56/12719/2018)**

Department of Electrical and Information Engineering

Faculty of Engineering

University of Nairobi

Signature .....  ..... Date ..... 05-08-2022 .....

This thesis is submitted for examination with our approval as research supervisors:

	Signature	Date
Dr. M. I. Kaniu Department of Physics University of Nairobi P.O Box 30197-00100, Nairobi, Kenya. <a href="mailto:ikaniu@uonbi.ac.ke">ikaniu@uonbi.ac.ke</a>		10-08-2022

Mr. M. J. Mangala Department of Electrical & Information Engineering University of Nairobi P.O Box 30197-00100, Nairobi, Kenya. <a href="mailto:michael.mangala@uonbi.ac.ke">michael.mangala@uonbi.ac.ke</a>		12.08.2022
--	---	------------

## **DEDICATION**

This thesis is dedicated to my family members who have been my source of encouragement and continually supports me financially and spiritually. Special tribute also goes to my great friend Henry K. Wambua, for his support, insight and inspiration throughout this study.

## **ACKNOWLEDGEMENT**

I wish to express my sincere gratitude to my research supervisors and friends for their support and contribution towards the preparation of this thesis and above all, the almighty God.

Specifically, I thank my supervisors; Dr. Ian Kaniu and Mr. Michael Mangala for their invaluable guidance throughout the course of this research studies and for steering my interest in the research topic until completion.

Special thanks to my brothers; Felix and George for their untiring continuous support and assistance throughout the study period.

## ABSTRACT

The aim of this project was to develop and test a microcontroller-based column scanner prototype, where data is transmitted wirelessly, stored and displayed on to a handheld controller module to show a scan profile of the column in real-time. A laboratory-scale scanner prototype was constructed using a wood frame; 1 m length and 0.5 m width and a circular acrylic material (diameter 25 cm and radial width 10 cm) that houses a low activity source, 114 mCi  $^{241}\text{Am}$  radiation source (NER-492, S/N: A-377) and Velleman K2645 GM detector. The scanner is operated by two 12 VDC geared motors and two-timing belts to drive the source-detector assembly, in both vertical and horizontal plane, along the column frame for preset increment heights of 5-10 cm movements for auto counting at the preselected preset time between 1-10s. Current scanning systems are limited to only vertical scan movement and cabled. Ultrasonic proximity sensor and 3-axis accelerometer were used to determine and limit both vertical and horizontal plane orientations of source-detector, respectively. The GM detector was connected to a microcontroller-based circuit which also controls the movement motors. Four 2.4 GHz nRFfl2401 radio modules facilitated wireless communication between the scanner and the handheld module within a radius of 100 m. A handheld controller module consisting an LCD display screen (3.5" x 2.5") was used to display the scan configurations and results. The software used to run the microcontrollers for both scanners and the handheld module was developed using the Arduino IDE, which is an open-source software using C++. The scanner offers the option to operate in both gamma scanning mode and for use in radio tracer measurements. In gamma scanning, two modes of operation are possible; automatic and manual counting. In radiotracer measurement mode, the detector is maintained in a fixed position for counting to assume a typical field radiotracer measurement. Reliability and safety features used included: CRC, Limit switch, 3-axis accelerometer and Internal watchdog. The model column is an 8" diameter PVC pipe fitted with two 5 cm wide, 6 mm mild steel rings at 22 cm intervals to represent distillation column and separation trays in a typical industrial process setup was tested for faults using the prototype. From the scan profile generated in gamma column scanning test, it was possible to locate the two separation trays and the simulated malfunction. The obtained results were validated through a comparison with the actual physical measurements. For radiotracer experiment, the instantaneous introduction and withdrawal of the radioactive source at different time

intervals was detected from the profile drawn in real-time display on the LCD screen with peaks corresponding to the various time intervals of radioactive source introduction and withdrawal. This project has demonstrated the possibility for the adoption of wireless control of gamma-based column scanners and wireless data transmissions to increase efficiency, minimize personnel exposure through increased operation ranges by eliminating cabling, reduced costs of operation, and has the potential to use in the harsh industrial environment. However, this prototype is suited only for laboratory demonstrations on practical applications of gamma column use in the diagnosis of industrial processes.

## TABLE OF CONTENTS

DECLARATION .....	ii
DEDICATION .....	iii
ACKNOWLEDGEMENT .....	iv
ABSTRACT .....	v
LIST OF FIGURES.....	11
LIST OF TABLES .....	12
LIST OF ABBREVIATIONS AND ACRONYMS.....	13
CHAPTER ONE .....	14
INTRODUCTION.....	14
1.1 Background to the study.....	14
1.2 Statement of the Problem.....	16
1.3 Research Objectives.....	16
1.3.1 Main objective .....	16
1.3.2 Specific objectives .....	17
1.4 Justification and Significance of the Study.....	17
1.5 Scope and Limitations of the Study.....	17
1.6 Organization and Structure of Thesis.....	18
CHAPTER TWO.....	19
LITERATURE REVIEW.....	19
2.1 Chapter Overview.....	19
2.2 Microprocessor applications and Microsystems.....	19
2.2.1 Optimization of Microprocessors with Respect to Computational paradigms .....	21
2.3 Internet of Things (IOT).....	21
2.3.1 Sensors in IoT .....	22
2.3.2 Communication in IoT .....	23
2.3.3 Data Signal processing in IoT .....	23
2.3.4 Data compression.....	24
2.3.5 Data security .....	24
2.4 Embedded systems.....	25
2.4.1 DC Motor Speed Control .....	26
2.4.2 Wireless communication network systems .....	26
2.5 Gamma Column Scanning and Radiotracer Measurements in Industries.....	27

CHAPTER THREE.....	30
THEORETICAL FRAMEWORK .....	30
3.1 Chapter Overview.....	30
3.2 Methods Used in Gamma Column Scanning.....	30
3.2.1 Classical Gamma Column Scanning Method .....	30
3.2.2 Modern Gamma Scanning .....	30
3.2.3 Gamma Radiation Detectors .....	32
3.2.4 Principles of Radiotracer Measurements .....	32
3.3 Data Acquisition and processing in Measuring Systems.....	36
3.3.1 Data Signal Sampling .....	36
3.3.2 Noise in Data Signals.....	37
3.3.3 Data Transmission .....	38
3.3.4 Digital Signal Processing.....	40
3.4 Nibras Data Acquisition Systems.....	40
CHAPTER FOUR.....	43
METHODOLOGY .....	43
4.1 Chapter Overview.....	43
4.2 System Design and Construction.....	43
4.3 Components used in the designed system development.....	47
4.3.1 Arduino Mega .....	48
4.3.2 DC Geared Motors .....	48
4.3.3 A 3.5” Thin Film Transistor LCD Screen.....	49
4.3.4 nRFL2401 Radio Module .....	49
4.3.5 433 MHz Transceiver and Receiver Pair Module.....	49
4.3.6 MPU6050 3- Axis Accelerometer .....	50
4.3.7 Ultrasonic Proximity Sensor .....	50
4.3.8 Power supply.....	50
4.3.9 Velleman K2645 Geiger Muller Detector.....	50
4.3.10 DC Motor driver .....	51
4.3.11 A Limit switch .....	51
4.3.12 Circular acrylic detector-source assembly .....	51
CHAPTER FIVE.....	54
RESULTS AND DISCUSSION .....	54



5.1 Chapter Overview.....	54
5.2 Overview of the developed prototype system.....	54
5.2.1 Scanner head system .....	54
5.2.2 Control Module system.....	56
5.2.3 Software Development.....	59
5.2.4 Scanner Data Acquisition .....	63
5.2.5 Scanner Data Transmission.....	63
5.2.6 Data Treatment in Control module: Reception, Buffing, Arraying and Plotting....	64
5.2.7 System Validation for Reliability and Safety .....	65
5.3 Test Results.....	67
5.3.1 Gamma column scanning setup .....	67
5.3.2 Radiotracer Configuration for Laboratory Measurements.....	73
CHAPTER SIX .....	77
CONCLUSIONS AND RECOMMENDATIONS.....	77
6.1 Conclusions.....	77
6.2 Recommendations.....	77
REFERENCES.....	79
APPENDICES.....	87
Appendix I: Components Specifications.....	87
Appendix II: System Booting and Distance Reading.....	90
Appendix III: Obtaining Scanner Orientation.....	76
Appendix IV: Scanner State Transitions.....	79
Appendix V: Scanner Movement.....	82
Appendix VI: Data Receiver and Utility Checks.....	88
Appendix VII: Gamma Chart Plotting Code.....	92
Appendix VIII: Plotting Radio Tracer Chart Points.....	99
Appendix IX: Detector Code.....	116
Appendix X: nRFL2401 and 433MHz RF Code.....	101
Appendix XI: Command Checking and Execution.....	104
Appendix XII: EEPROM code.....	127

## LIST OF FIGURES

Figure 1.1: Detector-source arrangement and scan profile .....	15
Figure 3.1: Radiotracing operation .....	33
Figure 3.2(a): Radiotracer application in flow measurement .....	34
Figure 3.2 (b): Scan profile for pulse velocity flow measurement .....	34
Figure 3.3: Leak detection using radiotracer .....	35
Figure 3.4: Short-circuit determination using radiotracer.....	35
Figure 3.5: Nibras System circuit. ....	41
Figure 3.6: Nibras System display. ....	41
Figure 3.7: PC connection for Nibras system. ....	42
Figure 4.1: Outlook of the conceptualised automated gamma scanning system prototype. ....	43
Figure 4.2: Block diagram of the designed gamma scanning system prototype. ....	44
Figure 4.3: The conceptualised graphical model of the physical system. ....	45
Figure 4.4: Scanner frame assembly unit.....	46
Figure 4.5(a): Schematic diagram of scanner head module components. ....	47
Figure 4.5(b): Schematic diagram of handheld control module components.....	48
Figure 4.6: (a) Voltage divider schematic, (b)Velleman K2645 GM .....	51
Figure 4.7: Design sketch of the Circular acrylic assembly. ....	53
Figure 5.1: Constructed Scanner Head Features assembly – top view.....	55
Figure 5.2: Scanner Head circuitry component layout. ....	56
Figure 5.3: Wireless controller circuitry component layout. ....	57
Figure 5.4: Keypad buttons component layout. ....	57
Figure 5.5: Keypad buttons electrical design circuit. ....	58
Figure 5.6: Constructed Wireless Handheld Control module.....	58
Figure 5.7: State machine coding flowchart for the developed system. ....	60
Figure 5.8: Software execution for scanner head/control module/user flowchart.....	61
Figure 5.9: Software execution for operator/control module flowchart.....	62
Figure 5.10: Sample of Scanner data serialization code. ....	64
Figure 5.11: HC-SRO4 -Ultrasonic sensor calibration.....	66
Figure 5.12: Scanner head system outlook with the model column. ....	68
Figure 5.13: The developed wireless handheld controller module.....	69
Figure 5.14: Menu options display for mode on the wireless controller.....	70
Figure 5.15: Menu options display for parameter selection on the wireless controller.....	71
Figure 5.16: MATLAB vs LCD column profile plots. ....	72
Figure 5.17: (a) Radiotracer scan simulation results on display .....	74
Figure 5.17: (b) Scan profile of radiotracer data using MATLAB.....	74

## LIST OF TABLES

Table A1.1: Arduino Mega Specifications .....	87
Table A1.2: 12V DC motor Specifications.....	87
Table A1.3: nRFL2401 Specifications .....	88
Table A1.4: 433 MHz RF Transmitter Module Specifications.....	88
Table A1.5: 433MHz RF receiver module specifications.....	88
Table A1.6: Power supply rating calculation.....	89

## LIST OF ABBREVIATIONS AND ACRONYMS

ABS	-	Acrylonitrile Butadiene Styrene
ADC	-	Analog-Digital Converter
AI	-	Artificial Intelligence
CNC	-	Computer Numerical control
CRC	-	Cyclic Redundancy Check
DC	-	Direct Current
EEPROM	-	Electrically Erasable Programmable Read Only Memory.
GDP	-	Gross Domestic Product
GM	-	Geiger Muller
IDE	-	Integrated Development Environment.
LCD	-	Liquid Crystal Display
LSB	-	Least Significant Bit
NER	-	Named Entity Recognition
PC	-	Personal Computer
RX	-	Receiver
SDRAM	-	Synchronous Dynamic Random-Access Memory
SRAM	-	Static Random-Access Memory
TFT	-	Thin Film Transistor
TX	-	Transmitter
USB	-	Universal Serial Bus
VDC	-	Volt of Direct Current

## CHAPTER ONE

### INTRODUCTION

#### 1.1 Background to the study

Industrialization is one of Kenya's long-term development goals towards the achievement of Vision 2030 goals. Currently, the government has identified four areas to invest in, branded as '*The Big Four Agendas*'; the manufacturing sector, health, housing and food security. The government aims at increasing the GDP share in the manufacturing sector from the current 9.2% to 20% by the year 2022. Nevertheless, the potential of application of Non-Destructive Nuclear Techniques is not significantly utilized to date (Ngui *et al.*, 2016).

The Kenya Government recognizes the role of nuclear science applications as a prerequisite for sustainable social and economic development, in general. Nuclear applications are widely used in; medicine, manufacturing and industry, agriculture and food security, development of water resources, development of energy resources, and in research institutions, for various multi-disciplinary studies. Nuclear technology finds a wide range of application in industrial and medical fields; specifically, these include the following techniques; Computed Tomography Scan, X-ray radiography, gamma column scanning, radiotracer measurements, among others (Kim *et al.*, 2011).

Gamma scanning technology is usually applied in industries to diagnoses the various processes. The method utilizes a radioactive source that emits radiations and a radiation detector, which records the density profile. Such industrial processes include; flow measurements, process equipment/systems, leakage detection of buried pipelines, residence time distributions, isotope hydrology and water resource management, sediment transport study, among other uses (Kim *et al.*, 2011). In principle, Gamma column scanning is widely employed in the inspection of anomalies involving columns in oil refineries and petroleum industries and reactors. These columns may be a tray or packed distillation and fractionation towers. In this method, the detector and the sealed source of radiation move simultaneously opposite in the same horizontal plane along the column

being diagnosed, as shown in Fig.1.1 in which the intensity readouts show the internal profile of the column (Froystein *et al.*, 2005).

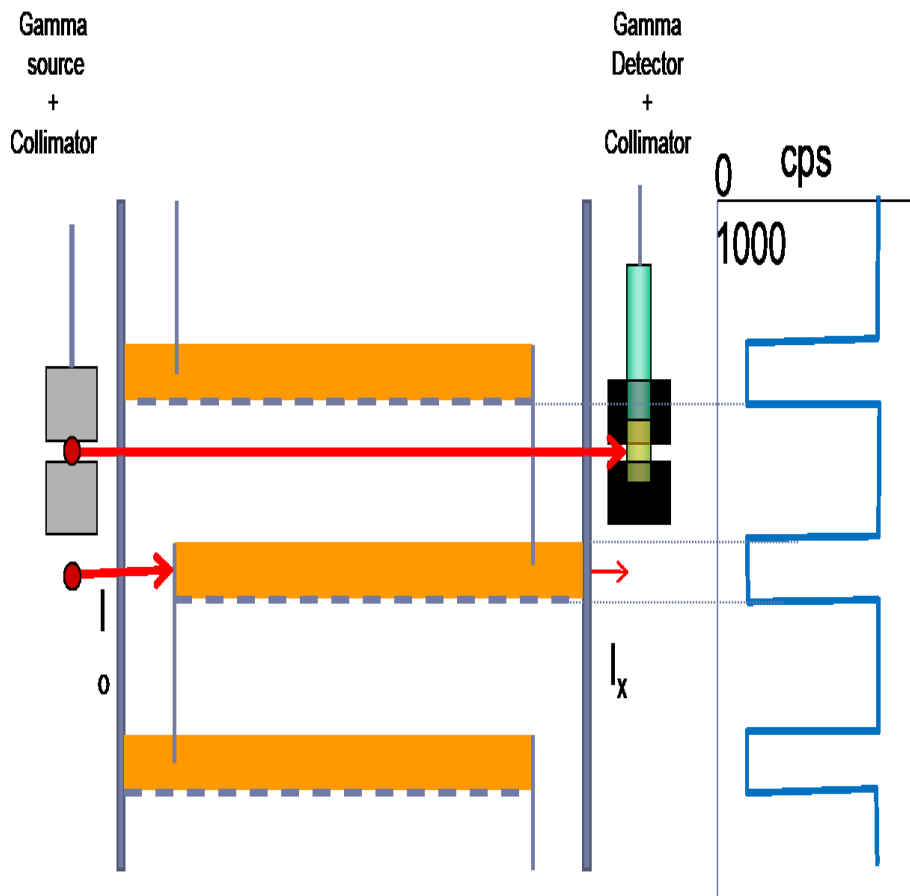


Figure 1.1: Detector-source arrangement and scan profile (Source: Sanches *et al.*, 2007).

Analysis of the generated profile through analysis software and comparing it with perfect mechanical design can enable one to deduce valuable conclusions about the equipment's functionality (Johansen, 2005). Examples include; deformed or displaced separation trays and even finer details relating to whether the column is entrained or flooded (Benahmed and Alami., 2012). These findings are helpful for process engineers and plant operators identify process anomalies, optimize performance, and initiate maintenance operations where necessary. Optimized columns are profitable to the company and also yield up to standards products.

The existing gamma scanning systems are made up of a winched system to lower or raise the source and detector and a wired communication system to a computer or laptop running data logging and analysis software for the recorded radiations (Stoddart, 1979). Such systems are manually operated by lowering the source and detector simultaneously

with a data logger connected through cabling, which restricts the study to limited coverage. In some cases, the environment may be harsh and constrained, thereby requiring several assistants to do measurements. This system also requires the person to operate within the immediate vicinity, which exposes him/her to the radiations.

An automated scanning system with wireless data acquisition and control can increase efficiency, safety, accuracy, and reliability of data and is easily accessible to the harsh environment compared to manual operated systems (Walinjkar *et al.*, 2009). This research project aimed at developing a fully automated gamma column scanning system with wireless data acquisition and control. This research was motivated by the need for a low cost, portable, safe, easy to operate, modular scanner for separation columns with reduced cabling and a more flexible and versatile system for radiotracer studies.

## **1.2 Statement of the Problem**

The need for online diagnosis of industrial processes is vital for the optimized operation of the equipment. The complexity in the design, layout and operation of this process equipment necessitates non-destructive testing in real-time validation and investigation of the process models. The system currently in use, is cabled in nature. This feature limits it to studies in only small coverage. Furthermore, most industrial processes are operated under harsh conditions which are hazardous to human health. The current system that restricts the researcher close to the vicinity under investigation during study becomes unsuitable for long time studies in such circumstances. Radiation safety is highly recommended in the handling of radioactive sources. However small it might be, free doses should be avoided. The existing system subjects the researchers to long exposure time since it is not possible to conduct the investigative studies remotely. Its cumbersomeness and manually intensive nature added to the high cost of acquiring the equipment proves the existing system to be uneconomical.

## **1.3 Research Objectives**

### **1.3.1 Main objective**

This work aimed to design and develop a low-cost microcontroller-based automated gamma column scanner head with a wireless data acquisition system.

### **1.3.2 Specific objectives**

- (i) To develop a fully automated gamma scanning head system with dual-axis movement capabilities;
- (ii) To develop a portable control module with capabilities for data treatment, control of the scanning head and detector - source movement.
- (iii) To develop a wireless communication system between the control module and the scanner head.

### **1.4 Justification and Significance of the Study**

Gamma distillation columns are vital in manufacturing industries, especially ones dealing with products that require distillation. Regular gamma scanning is required to maintain these columns in a good working condition, which helps maintain quality products, reduce operational costs, and meet regulatory requirements. Such regulatory requirements are put into place to safeguard consumers against bad quality, limit energy consumption and safeguard the environment against pollution. Current gamma scanning systems are human-intensive, its limited to only one scan line at a time since it can only move vertically, simultaneous movement of the source and detector involve several stages of scanning which can run into several hours if not days due to need for further analysis using computer software. Cabled communication limits the study area, especially in radiotracer measurements, while confining the personnel within the vicinity of high radiation exposure. There is therefore a need for a microcontroller-based system with motorized source and detector movement in dual axis, wireless control and data transmission, and real-time scan results. This will significantly increase reliability and accuracy, reduce radiation exposure risks, and lower gamma scanning systems and operations costs.

### **1.5 Scope and Limitations of the Study**

In this project, a microcontroller-based lab-scale gamma scanner prototype has been designed and constructed. The prototype developed provides for the following salient features; motorized vertical and horizontal movement, scan data acquisition, and wireless



transmission within a radius of not more than 100 m to a hand-held battery-powered control module, data treatment and graphical display on miniature LCD. The prototype was designed for only one detector and tested using a low activity radioactive source Am-241, 114 mCi after configuration. The results were acquired compared with the physical trays locations and characteristics.

## **1.6 Organization and Structure of Thesis**

This thesis contains six chapters and an appendix. The design and development of the scanner methodology, data acquisition, processing and transmission, system hardware design, system control and wireless communication system, software and data treatment is discussed in detail in the main chapters. The appendix contains supporting information on the code developed to run the systems. Chapter 1 discusses the background of the research area, objectives and the scope of the study. Chapter 2 reviews literature with studies done on Microsystems and Microprocessor applications, Internet of things, Embedded systems, DC motor control and wireless communication networks. Chapter 3 reviews the theoretical principles on the existing systems for classical and modern ways of industrial gamma scanning, methods and applications, types of gamma radiation detectors, data acquisition and signal processing, and various modes of wireless communication Chapter 4 focuses on the methodologies used in the design, materials/components and development of both the hardware and software to produce a reliable system that meets the objectives of this project. Chapter 5 primarily provides the results and discussions of the prototype developed and validation of the prototype. Chapter 6 concludes the thesis with discussion and recommendation drawn from the study. Appendix contain the codes of various operations within the system, mainly in; data acquisition and processing, control, transmission, display and treatment.

## CHAPTER TWO

### LITERATURE REVIEW

#### 2.1 Chapter Overview

This chapter discusses the literature review of this study which has been divided into various sections. Section 2.2 reviews the recent developments of nanotechnology in the field of electronics and also focuses on some study cases in the application of microprocessors, microelectronics and microsystems while Section 2.3 reviews the relevance of the Internet of things and its application in the engineering world. Section 2.4 describes the technology driving embedded systems and their application. Section 2.5 winds the chapter by highlighting some case studies and challenges faced using the existing system.

#### 2.2 Microprocessor applications and Microsystems

Recent developments in wireless data communications, Micro-Electro-Mechanical Systems, and digital electronics technology have led to Wireless Sensor Networks' adoption in Industrial, Medical, learning and telecommunication fields (Stokic *et al.*, 2006). Wireless Networks have become one of the most vital technologies in the current world with the potential to provide people with a more comfortable life in their work. This technology has offered an effective solution to many problems. The rising popularity of these wireless networks has motivated many innovations in the engineering field, especially in embedded systems. These wireless networks consist of; battery-powered sensor module, a processing system for data computations and processing, and a subsystem with communication capabilities (Bhagwat *et al.*, 2017).

In research carried out by the U.S. military in the 1970s, it was found out that wireless networks were advantageous over conventional systems used since it offered wide coverage, high accuracy and optimal reliability at a relatively lower cost. However, wireless networks experienced some challenges: unreliable wireless communication systems, limited power source, and large-scale data deployment. de to control embedded systems in machines, the motor vehicle industry, robots, medical devices and other gadgets. Microprocessors play a vital role in the fabrication of microsystems. Many

microsystems have been integrated in medical instruments, industrial process equipment, and security and home appliances. According to D. Wise Kens (2008), microsystems will play a pivotal role in improving life in the near future.

Further, Zhirnov and Cavinlll (2011) elaborate that these systems present a potent tool in the arsenal to tackle most problems, especially in health, security, manufacturing, energy, environment, and production of food. Microsystems are made of three sections; electronics, communication system and sensors. With the rate of advancement in Nanotechnology, microsystems will be able to collect data remotely from the physical environment, perform computational analysis and transmit information through information networks. As microelectronics changed data processing and communications, so will microsystems solve challenges of non-electronic systems.

Wireless Integrated microsystems should be incorporated with nonelectrical components. In the study to develop a gas chromatography model, the use of electromechanical components and semiconductor in a unit hybrid system improves system functionality and makes it adaptable in many areas of application.

In a study carried out by Piotter (2012), nanotechnology proves to be a promising technology in the near future, specifically in automotive engineering. It has found applications in IT, Power generation, brown and white goods industries, machine production and chemical engineering. Emphasis should be made on enhancing the technology and large-scale production of complex microelectronic components.

Recent developments in the manufacture of meta devices have been done on how to configure nonlinear metamaterials to produce meta devices by combining actuation systems and quantum materials with meta-atoms. Microelectromechanical systems provide crucial platforms to manipulate the effective features of metamaterials and the incorporation of great functionalities with metamaterials (Zhao, 2019).

In bio instrumentation, microcontrollers are used to control leg prostheses, where microprocessors are used for feature extraction, projection and classification. The microelectronics used in data acquisition and signal processing help in the improvement of precision of the leg prostheses (Alberto *et al.*, 2000).

In a study carried out by Susumu Noda (2008), a band structure of photon can be produced in the spectrum of transmission of optics using microsystems. This makes it

possible to produce 2D and 3D crystals (Ho *et al.*, 2000). Optical microsystems thus contribute a lot to optical communications by the use of integrated optics. As this technology develops from a laboratory scale into the industrial world, it will be possible to develop optical systems dependent on the waveguide. This will be the driving force in the production of integrated micro-optical systems with high-performance features (Lee *et al.*, 2006).

In conclusion, microcontroller-based systems incorporated with actuators are reliably functional, consume low power, and are small in size, making them suitable in optical telecommunication systems (Dan, 2008). The industry is adopting nanotechnology, and many companies and other microsystem scientific groups are rising due to innovations in nanotechnology (Pradeep, 2015).

### **2.2.1 Optimization of Microprocessors with Respect to Computational paradigms**

For microprocessors to effectively fit in the future development and application of artificial intelligence and microsystems, they must be redesigned to ensure Efficiency- they must be optimized for low energy use and reduced costs with highly reliable performance and comprehensive area coverage. Security proof- this owes to the fact that the rising growth in nanotechnology is more susceptible to malicious attacks. Extensibility- they should have provisions for extending functionalities, especially for on-chip peripherals. They should also be configurable into different applications. Therefore, in the architecture of microprocessors, a lot of foresight will be required to make them future-proof in terms of future trends in their applications while maintaining conformity with current applications (Tosiron *et al.*, 2017).

### **2.3 Internet of Things (IOT)**

The Internet of things describes the interconnection of unique physical devices to collect data and run operations to improve productivity and eliminate data acquisition through human interventions. The proliferation of microelectronic technology will lead to a data explosion that will raise the cost of transmitting data regarding latency and energy consumed by these low power consuming devices. These costs can be reduced through edge computations before data transmissions at the nodes to read and utilize the data transmitted (Patel *et al.*, 2017).

While research has been done on the challenges facing communication in the IoT connections, more attention needs to be directed to the computational aspect of IoT embedded systems, especially microprocessors in the devices (Adegbija, 2017). To produce various microprocessors that are extensible, easy to scale and configure for future IoT generation, emphasis should be put on the optimization of microarchitecture and the paradigms of computations while in the design of the microprocessors. This will facilitate the effectiveness and reliability of edge computations.

Gaura *et al.*, (2006) proposed the following characteristics to be considered in the design of microprocessors to make them fit for wide applications: Intelligence. It should be dynamic to changing operation scenarios, Heterogeneity in which the system should offer seamless communication with others for efficient data use. Complexity in which it should be complex enough to run a wide range of applications. Scalability- since the future IoT will be made up of billions of devices, the microprocessor must be area efficient and portable. Real-time deadline where it should be stringent like in medical diagnostics. Spatial constraints in which they should be fault tolerant and adapt to changing conditions of operation. Finally, they should feature Inter-node support in which some execution resources are shared to execute duties efficiently.

### **2.3.1 Sensors in IoT**

IoT finds wide application in the field of sensors (Sundmaeker *et al.*, 2006). Sensors are used in data acquisition, especially in a physical environment like temperature, motion, pressure etc. The data of interest acquired using these sensors is subjected to further processing and analysis before being converted into a form conceivable by a human. Like in sensor fusion, IoT is used to fuse readings from several sensors and create more accurate, robust and qualitatively higher data than the raw data.

Algorithms used in the fusion of sensor is classified into several levels of memory intensities depending on the level of operation computations involved. For example, aggregation of data using simple computations like addition and mean. It can also involve complex applications like handling vector data such as streams of recorded videos from multiple sensing systems (Khiter and Khechiba, 2016). Like in medical diagnostics, several sensors can automatically monitor physiological parameters like heart beat rate and blood pressure using sensors that are not invasive. If IoT is incorporated into biomedical equipment like electrocardiography and electromyography machines, data can

be analyzed without the need to transmit it for analysis in another machine or platform (Gubbi *et al.*, 2008).

### **2.3.2 Communication in IoT**

According to Chandrakant *et al.* (2017), communication is common in IoT technology. This is due to the fact that its structure is intrinsically connected, and data transfer is through interconnected nodes. Some data transfer methods in IoT include Bluetooth and Wi-Fi, which follow specific protocols in transmission or communication. The communication technology driving data transmission in these systems is the software-defined radio (SDR). This technology is rapidly taking over the communication area due to its compatibility with IoT. This is due to its ability to perform the layer functions, which are physical in nature, in software instead of hardware.

SDR is highly flexible, making it easy to incorporate with other radio functions and bands without updating the hardware. It consists of an antenna, ADC and DAC interconnected for transmission and receiving data (Barr and Massa, 2006). The input data is then subjected to digital signal processing to output information in the required format. Typically, algorithms of SDR can be executed in the normal microprocessors; therefore, it is easily compatible with many types of the microprocessor, which makes it applicable in many fields of communication systems.

### **2.3.3 Data Signal processing in IoT**

Image signal processing is vital in many cases involving IoT use. This calls for microarchitectures in design of devices that offer efficient processing operations of images and other communication data signals (Chippa *et al.*, 2013). The recent developing IoT applications, such as automatic Number plate and face recognition, involves image processing in several forms like detection, recognition, extraction and classification. In medical applications, image processing increases the reproducibility of diagnostics of diseases, giving the medics quantitative information from archived images. This can supplement the data under use by medical specialists (Mort *et al.*, 2016).

There is a need to equip the portable medical instruments with image processing capabilities for quick diagnostics and analysis of scan results for patient condition assessment remotely (Bash *et al.*, 2006). For efficient and easy sharing of data by researchers, there is a need to embrace some data optimization technologies with emphasis made in IoT approach with the context in edge computing since image

processing rich in data and memory intensive. Besides, some data processing requires a large amount of data storage.

#### **2.3.4 Data compression**

To guarantee easy retrieval, transmission and analysis of data, several IoT techniques have emerged which facilitate the compression of large volumes of process data and reduce latency in the transmission and costs as well (Mohan *et al.*, 2012). This is motivated by the rapid increase in data transmission while the available communication systems have limited bandwidth. In the edge node, since the storage is limited during computations, compression of data helps minimize the storage memory required, especially in the IoT, which is constrained in terms of storage resources.

This data compression can be done using source encoding or channel encoding. However, in the case of edge node computing, source encoding is relevant and more applicable than channel encoding (Anderson *et al.*, 2012). Compression can be divided into two general categories; lossy and lossless. In scenarios where high-level fidelity is required, such as in medical imaging, lossless is more suitable.

#### **2.3.5 Data security**

According to Creinne *et al.* (2013), the safety and fidelity of data is highly recommended and a key consideration in every design of communication system. IoT systems are often deployed in potentially harmful areas susceptible to attacks and potential privacy issues in many cases. This necessitates security requirements as a fundamental factor to consider in order to ensure integrity of data and safety of the devices (Bortolotti *et al.*, 2014). Other sensitive applications like in medical equipment may require additional safety features from hardware and software to safeguard sensitive data from access to unauthorized persons (Koeberl *et al.*, 2014).

In most designs of these devices, the development of security features is at infant stages. There is a need to bridge the existing knowledge gap in the construction features of microprocessors. This will ensure that they can incorporate features that support algorithms for the execution of operational security requirements without compromising the functional integrity of the devices (Stitt *et al.*, 2003). Data encryption is commonly used to maintain confidentiality, where an algorithm is used to perform encryption and data generation that is only readable once decrypted (Saha, 2006). This encryption

algorithm is memory intensive hence the speed of encryption will be determined by the available memory access.

In his conclusion, Tosiron (2014), the Internet of things will take over in many applications ranging from medical, industrial, homestead, etc. This will significantly bring a constructive transformation in life and the global economy. Its proliferation will lead to the generation of huge amounts of data that will require transmission, bringing bottleneck competition in the limited bandwidth. To overcome this challenge, IoT electronic devices will require to be incorporated with edge computing capabilities with microprocessors and algorithms that can compute the data on the edge nodes and interpret.

## **2.4 Embedded systems**

In the contemporary world, technology is shifting from developing electrical and mechanical rigid solutions to modern challenges. Embedded systems are systems incorporated in other systems to increase their efficiency and proves to be at the core of every future development in the engineering world (Sukriti, 2009). In almost every technological equipment lies an aspect of an embedded system. In a peer-reviewed paper published by Oluwole (2015), embedded systems find wide application in the design of intelligent cars, buildings, industrial processes, aeronautic gears, and medical, agriculture and grid systems.

In these systems, micro-controllers are embedded in various mechanical and electronic components to perform specific tasks through actuators (Steve, 2003). These embedded systems control robust electrical and mechanical systems using a microprocessor plugged into the circuit. These systems are housed within the systems they control, and being a small unique microcomputer, they are programmed to run a predefined function. Since they are custom-made for a specific task, their design can be optimized to a small size with minimal microelectronic components, reducing the cost (Michael and Anthony., 2006).



Sukritti (2009) classifies embedded systems in two sections; Software and hardware. In his argument, the software has more heavy demands than the hardware development since it provides the human interface, machine control and data processing. Studies done show that in the continent of Africa, the engineering of embedded systems has not been fully tapped since no Institution offers it as a main course (Oyetoke., 2015). Nevertheless, this field can positively impact oil and gas, medical, finance institutions, aviation etc., if fully embraced.

#### **2.4.1 DC Motor Speed Control**

DC motors find many applications in rolling mills, electric cars and trains, cranes, robots, among other uses, due to ease in its speed control to give the performance characteristics required for specific tasks (Walced *et al.*, 2012). DC motors are classified into two classes; brushed and brushless. It is wise to pick the type which serves you best depending on your project requirements. Controllers produce a signal to represent the desired speed and maintain the DC motor in that specific speed (Pal *et al.*, 2012). Controllers may monitor the motor speed in order to provide feedback for reduction of error; hence the name closed-loop system. If not, it is called an open-loop system.

To control the speed of the DC motor at a fixed supply voltage, speed controllers are used. Proportional Integral –Derivative (PID) as a speed control mechanism is commonly used due to its robustness to modelling. However, it is limited due to the poor tuning of features required to meet the desired design (Hagglund., 1995). According to Emiliano *et al.* (2017), Pulse Width Modulation controllers are more suitable for small DC motors, especially for prototypes driven by microcontrollers, since they are easy to program and precise. In this mode of control, the speed controller works by varying the input voltage supply depending on the duty cycle of the output signal of the PWD. Else, the motor speed can also be controlled by instantaneously switching on and off the supply (50-200 mSecs) depending on motor inertia (Khechiba, 2016).

#### **2.4.2 Wireless communication network systems**

Wireless communication networks offer a wide range of applications: surveillance systems, tracking systems, health, agriculture, and military systems (Bhaskar *et al.*, 2014). Due to their outstanding features and limitations, they face various challenges, limiting their application and complicating their development. This system requires a middleware to link up between sensor hardware and the communication hardware. This middleware

will highly revolve around the software layer for the system to meet the design of the system and implementation goals.

In the contemporary world, wireless sensor networks are rapidly taking over the field. These networks comprise various micro – sensor nodes in large numbers, and the nodes communicate wirelessly like in radio media (Hiren *et al.*, 2014). The sensor nodes can function very well even in harsh conditions depending on the environment they are deployed. Generally, a wireless sensor network consists of; Field sensors, a data processing and aggregation system and wireless communication. Sensor nodes are connected with sink nodes in remote location wirelessly. These sensor nodes are limited in resource, while the sink nodes are more resourceful. When the two nodes are arranged together, they can be used to collect and process data through distributed systems. An advantage of these networks is that they are highly organized and don't require starting from an external mechanism (Nityananda, 2014). Besides, for the wireless sensor network to give a reliable performance, the programmers should develop a matching layer of software. The software helps in the management of the resources of the system and the performance of the system as a whole.

## **2.5 Gamma Column Scanning and Radiotracer Measurements in Industries**

The diagnosis of the process column using the gamma-ray technique plays a vital role in troubleshooting and identifying process problems. In his study using a newly generated gamma-ray scanning gauge, Khalid *et al.* (2006) argue that the gamma ray scanning technique gives the most precise picture and in real-time of all the other non-destructive techniques. According to Alami *et al.* (2006), more than 40 years ago, this technique has offered the following advantages over other techniques; its cost-effectiveness improves process efficiency and output in industrial processes and time-saving.

In a technical report from IAEA (2001), the majority of developed countries that run big petrochemical industries have embraced this technique as a means of problem identification in distillation columns without disrupting the production processes. El Badri *et al.* (2005), points out the need for African countries to embrace this technology to optimize the process installations, especially in petrochemical Industries.

The success of this technique is attributed to its outstanding capability to give information on the column internals non-intrusively, which is not possible with other techniques.

Using the gamma ray technique, the measurement system used provides a scan profile and is compared visually as per the mechanical design of the column when normally operating. In their findings Nourel *et al.* (2011), regions of abnormalities show differences or distortions in the profile obtained are interpreted to indicate the presence of problems in the internal structure of the column. This technique can be used to detect various abnormalities in distillation columns which include; flooding, weeping, mechanical damage, entrainment etc. In such process diagnosis using this technique, interpretation is done visually and relies heavily on human judgement; hence this requires experts with experience in the field. Sole reliance on human interpretation can be tedious, prone to wrong results, time-consuming and expensive expert service costs.

In conclusion, Ashraf *et al.* (2011) suggest the potential of automating the technique and incorporating pattern recognition software to eliminate human interpretation. He further argues that if improvements are made to the existing technique and an integrated gamma scanning system actualized, it will enhance productivity and lower operating costs.

Radiotracer measurements technology has a wide application in wastewater treatment plants, mining industry, petroleum and petrochemical industries (IAEA, 2008). In this technology, a radiotracer is injected at the inlet of a process, and appropriate detectors positioned at critical points and outlet monitor the flow of the radiotracer. The data obtained using these detectors, the mixing rates, residence time distribution and flow rate measurement, can be determined (Dawi, 2010). This information helps in decision making that will lead to process optimization, pollution reduction, energy-saving and improved product quality (Farooq *et al.*, 2003).

In practice, radiotracers can be used in large scale industrial processes, unlike conventional tracers which are constrained by their interference with industrial processes. In wastewater treatment, radiotracers offer an efficient way of assessing the effectiveness of the treatment units before discharging to the environment. In a study carried out by Wendy (2019) on the Dandora wastewater treatment plant in Kenya, a Radiotracer measurement confirmed that there were flow abnormalities and the pond was not effective for anaerobic wastewater treatment. However, the challenge faced was that the data acquisition system provided for a limited number of detectors, and the cabled nature of the system confined them to a small study area.

In another study carried out in Kenya by Gitau (2019) in assessing flow dynamics of clinker in the cement industry, through radiotracer technology, the functionality of the clinker was found to be at optimum since no flow abnormality like bypassing was traced. This is evidence that radiotracer technology is effective and has a wide application. However, the environment was harsh, and the system's nature constrained them within the immediate vicinity, which was unfriendly.

This project sought to close the gaps of the existing system in the following ways; Automation of detector-source movement in Vertical and lateral directions whereby an electromechanical movement drive unit was identified to be an effective tool in realization of the movement. This unit composed of a timing belt, Ultrasonic sensor, two 12VDC motors and 3-axis accelerometer, all controlled by a microcontroller.

Introduction of a wireless data transmission mechanism- To achieve this, radiofrequency mode of communication was identified due to the availability of the RF modules. These RF modules include a pair of 2.4 GHz nRF12401 radio modules and a pair at 433MHz frequency Radio module. The communication was powered by the microcontrollers.

Introduction of a highly portable handheld device for data processing and treatment to replace data logger and laptop. In the development of this gadget, the following components were identified for use due to their compatibility; TFT LCD screen, keypad and Microcontroller.

## CHAPTER THREE

### THEORETICAL FRAMEWORK

#### 3.1 Chapter Overview

This chapter discusses the basis of the study in which Section 3.2 discusses classical and modern Gamma column scanning method. Section 3.3 discusses data handling process while Section 3.4 focus on digital data processing using Nibras system.

#### 3.2 Methods Used in Gamma Column Scanning

##### 3.2.1 Classical Gamma Column Scanning Method

This is the basic gamma column scanning method in industry, in which, the source and detector are winched on top of the column and connected to a pulley system with a handle. By turning the handle, the source and detector are moved downwards or upwards simultaneously. The turns of the pulley system control the movement distance. The combination, source and detector, are also lowered to the elevations of interest in the column, for example, where a tray is located as per the technical specifications. The detector is connected to a logging system that continuously displays the number of counts per second throughout the scan progress. Resultant counts per second at each elevation are recorded and later plotted against elevation values to represent the column internal (Bao *et al.*, 2002).

The operation of the classical scanning involves one or two people lowering the source and detector, one person recording readings and another person determining the best elevation as per drawings (Jaafar, 2005). At best, the scanning can be carried out by three persons. This introduces errors especially positioning and reading out, and exposes operators to the harsh conditions in the column location. Extra work is also involved in manually plotting the counts or manually entering the data to a plotting software.

##### 3.2.2 Modern Gamma Scanning

Modern gamma scanning techniques attempt to solve shortcomings experienced using classical scanning techniques and makes the process as less human-involving as possible. The bare minimum attempt is to eliminate manual data readouts and plotting by dividing the system into two modules; one that controls the source and detector movement while

acquiring scan data. Another one receives the data and conducts scan interpretation. This is done by connecting the detector setup to a computer using appropriate software. During source and detector movement, data is transmitted via a cable to a computer loaded with plotting software, which automatically draws the recorded count per second versus elevation (Laraki, 2006).

To reduce errors due to human operated movement of the source and detector, the winched pulley mechanism is replaced by a motor-driven winch system. This motor-driven system is controlled from the connected computer, which is loaded with data acquisition and control software. As the source and detector move up or down the column side, the real-time plot of the counts per second at the specific elevations is generated. The motor-driven system consists of a gear mechanism that is machined to fineness to allow known movement length per revolution from which elevation is calculated. In some instances, rotary encoders are added to the drive unit to increase reliability. Motors used in the drive units include servo motors, dc geared motors or stepper motors chosen as per the system design (Gurashan *et al.*, 2006).

Programmable Logic Controllers (PLCs) have been used to control the movement of drive motors through electric motor drivers (Bora *et al.*, 2012). This has dramatically increased the reliability and efficiency of gamma scanners as the system can be operated for longer durations and scans carried out multiple times to provide comparisons. The addition of sensors in the scan systems has increased security as malfunctions and extremes can be detected and acted upon as the scan takes place, preventing damage or malfunctions. The communication between the movement PLC system and the control computer or laptop is usually cabled.

There have been attempts to have a visual 2-Dimensional analysis of columns based on the received counts per second. This involves a data synthesis software that reconstructs received data into graphical representation that is easy to analyse compared to graphical plots for the same parameters. These graphical representations are derived using higher-order mathematical functions and complex reconstruction software, which can also detect other types of tray properties like liquid holdup levels and the nature of packings (Haraguchi *et al.*, 2018).

### 3.2.3 Gamma Radiation Detectors

Gamma-ray photons are electromagnetic radiations and can interact with them through various processes such as photoelectric effect, Compton scattering, pair production and Raleigh scattering. Gamma-ray detection involves ionization where the photons release its energy to the electrons it is interacting with. These ionized electrons create a chain of collisions with other electrons in the matter. This collision energy is collected and used to represent a ray. The collection of this energy is either using a proportional counter like a solid-state semiconductor or indirectly using a scintillator detector (Diehl, 2001). The output from such detectors is usually an electrical pulse which is proportional to the amount of energy collected by the detector. The common types of gamma-ray detectors are Gas-filled detectors, Scintillation detectors and Solid-state detectors.

### 3.2.4 Principles of Radiotracer Measurements

A tracer is a substance whose properties can be used to identify another product. Radiotracer involves the application of radioactive isotopes or gamma-ray emitters in tracing operations. This application is divided into sealed applications where the isotope does not make contact with the material or system being analyzed, while radiotracers applications make contact with the material or medium (Alami and Bensitel, 2012). Radiotracer application has been successful in industrial setups. This is attributed to the active radioactive materials unique properties when used in tracing applications which cannot be achieved by standard industrial investigation techniques (Margrita, 1983). The unique properties of radioactive tracers like the ability to be detected through walls, ease of analysis and high sensitivity stand out among other tracing choices.

In radiotracer application, a tracer is injected at the inlet of a flow system as shown in Fig 3.1. Using a suitable detector outside the system outlet walls, emitted gamma-rays from the tracer are detected and recorded as flow occurs inside the system. Detected concentration versus time curve at the outlet is drawn. Based on Resident Time Distribution (RTD) concept, which is a density probability distribution for a particle entering and leaving the system, experimental residence time distribution  $E(t)$  can be calculated using the equation shown below;

$$E(t) = \frac{c_T(t)}{\int_0^{\infty} c_T(t)dt} \quad (3.1)$$

From this equation,  $t_a$ , which is the time it takes from injection at the inlet to detection at the outlet, can be calculated.

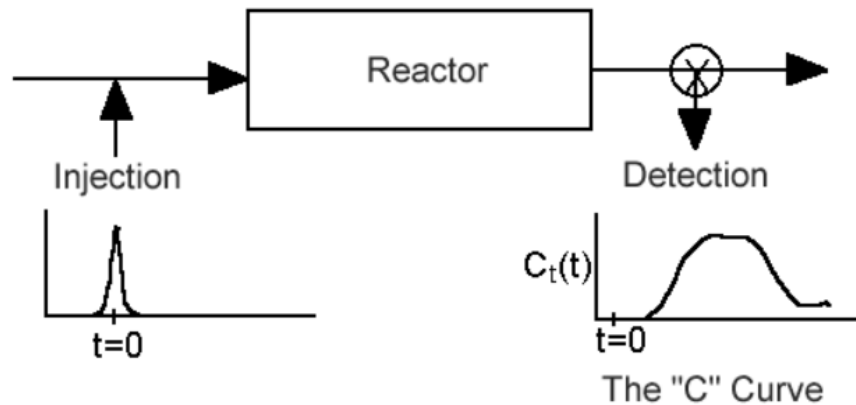


Figure 3.1: Radiotracing operation (Source: Alami and Bensitel, 2012).

#### 3.2.4.1 Flowrate measurement

Based on Allen's method, also known as the two peaks methods, flow rates in pipes can be determined. This method involves an injection of small gamma emitter source quantity in the process channel and external detection at the outlet to detect its arrival. The transit time between the first and the second impulse is used to calculate average flow speed, which can be converted to volumetric flow (Shen *et al.*, 2012). Fig.3.2 (a) shows the tracer injection points while Fig.3.2 (b) the corresponding profile generated, where the regions of peak indicate the outlet points. The time taken between the time of tracer injection can also be calculated in determination of residence time of process.



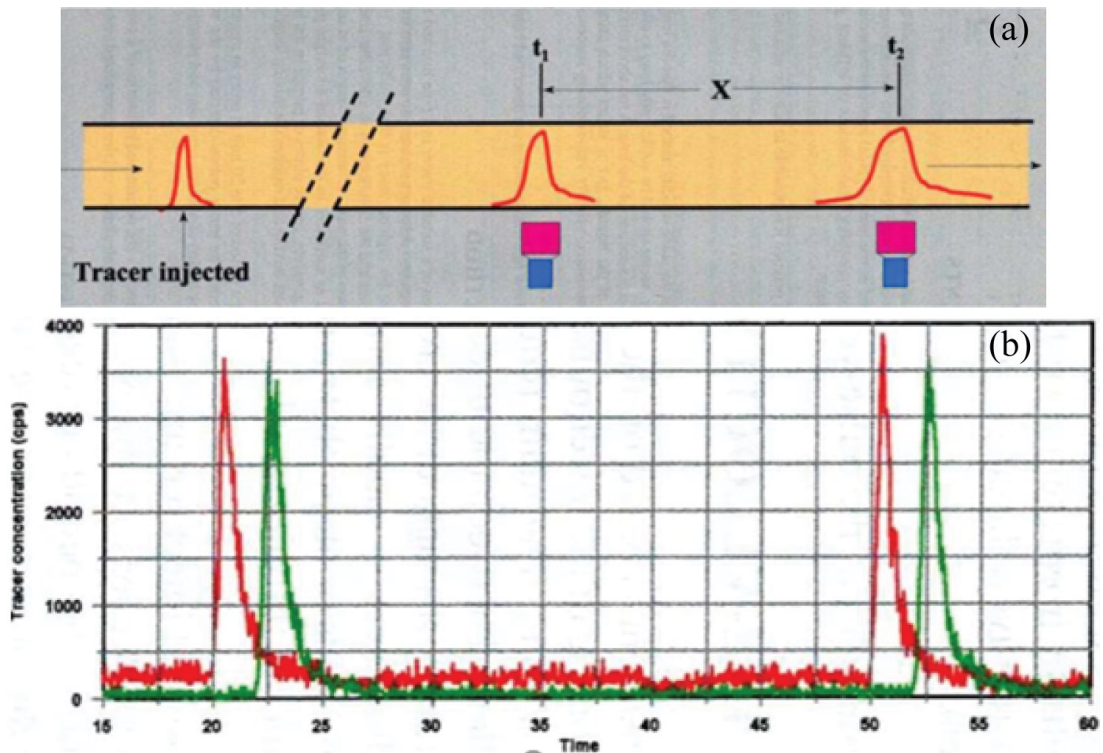


Figure 3.2(a): Radiotracer application in flow measurement and (b): Scan profile for pulse velocity flow measurement (Source: Alami and Bensitel, 2012).

### 3.2.4.2 Leak detection

Leaks are undesirable as they cause product losses and also contamination. Their presence, magnitude and location can be determined using radiotracer application. To determine the presence of a leak in a system, like a pipeline, or any suitable flow setup, the amount of detected radiation outside the system confines represents a leak. A detector at the end of the system can be used to quantify the radioactivity losses in the line. A common technique used in buried pipes is the “pig technique”. In this technique, a radiotracer is pumped into the pipeline, followed by a detector and a data logger assembled into a ‘pig’. As the pig moves inside the pipe, since the leak location has the tracer on the outside walls of the pipe, this radiation is picked up by the pig hence locating the leak. This method can detect leaks of even 0.1litre per minute (IAEA,2009). Fig 3.3 shows the buried pipe and tracer injection points and the corresponding scan profile which indicates the peaks corresponding to the points of leakage in the pipe.

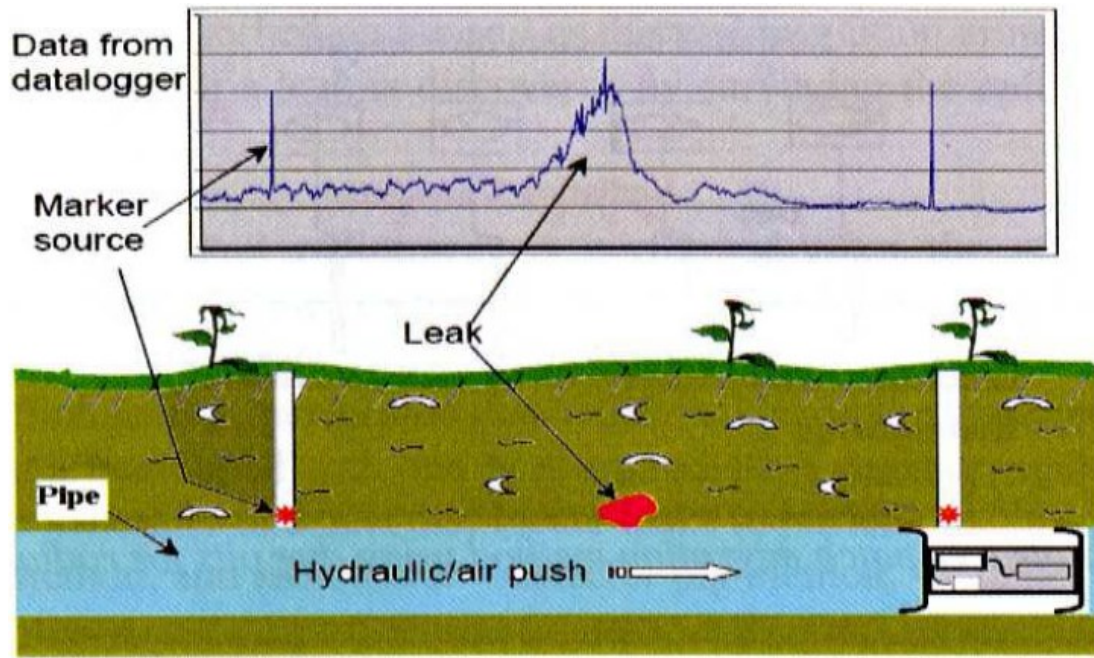


Figure 3.3: Leak detection using radiotracer (Source: Alami and Bensitel, 2012).

### 3.2.4.3 Determination of Short-Circuits in a Reactor

Short-circuits are quick fluid circulations in a system rather than flows. Injected tracer followed by detection shows a residence time distribution (RTD) with abnormalities indicated by a narrow peak over the primary initial response as shown in Fig 3.4 (Barati *et al.*, 2019). Short-circuit ratio ( $\alpha$ ) can be calculated using the formula below;

$$\alpha = \frac{A}{A+B} \quad (3.2)$$

where A and B are the respective total curve areas.

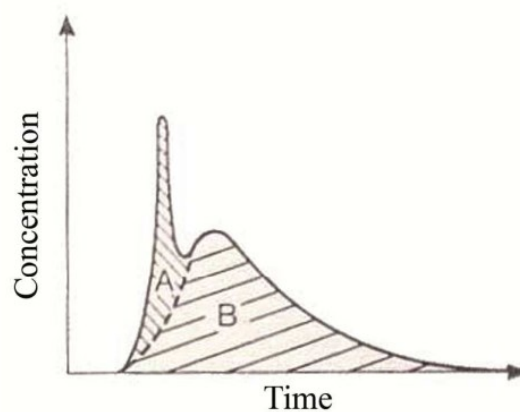


Figure 3.4: Short-circuit determination using radiotracer (Source: Alami and Bensitel, 2012).

### **3.3 Data Acquisition and processing in Measuring Systems**

Data acquisition involves reading electrical signals from a particular type of sensor into a processing unit, including a computer or a microprocessor. In any measuring system, a sensor forms the primary element. A sensor interfaces the system with the environment and provides an output that is relative to the measurand. Data acquisition starts with the measurement using a sensor coupled with a transducer. The transducer converts a measurable physical quantity into an electric signal which can be passed to a signal conditioner (Gyorki, 2004).

Signal conditioning refers to converting a signal produced by a transducer into a form that an analogue-to-digital converter can measure. Signal conditioning includes amplification, filtration, property isolation, current-to-voltage conversion as well as voltage-to-current conversion. An Analogue-to-digital converter (ADC) receives the conditioned signal and converts it from the raw form (analogue) to a form that a microprocessor or a computer can process (digital). An analogue signal is usually presented in the form of a continuous time-varying nature, while a digital signal is in the form of digital numbers. A digital number uses finite resolution steps to represent an input voltage (Bhagwat *et al.*, 2017).

ADC resolution is based on the number of bits representing the digital number, which is an n-bit resolution of 1 part in  $2^n$ . Types of ADCs include; Parallel converter, which uses a reference voltage at the full scale in the input and a resistor combination in series. These types of converters are very fast, with speeds of up to 500MHz (Madhvi *et al.*, 2015). Successive approximation ADC uses a digital-to-analogue converter and a single comparator. Voltage-to-frequency ADC converts a voltage at the input to a pulse output of a particular frequency. Integrating ADC utilizes the integration technique using a capacitor to determine the input voltage and output accurately. The difference between these types of ADCs is their resolution, operation speed and accuracy.

#### **3.3.1 Data Signal Sampling**

For an ADC to accurately convert the signal, sampling has to be done. Signal sampling is the process of acquiring several signal readings at regular intervals during which conversion occurs. Nyquist sampling theorem states that if the frequency of a signal is less than the cut-off frequency  $f_c$ , the signal's information can be obtained by sampling at  $2f_c$  (Gyorki, 2004).

According to Bentley (2008), sampling takes an approximate shape to that of the analogue signal to provide aliasing where a high-frequency signal is converted into a low-frequency signal. Modern ADCs operate at sampling ranges of between 5 to 10 times the signal's highest frequency to fulfil the Nyquist sampling theorem and guarantee accurate conversion.

During signal sampling, there are high chances that the digital value of that signal will be discrete just above the particular value of the analogue signal at that point or a discrete value just below the analogue signal value. This phenomenon is called quantization and comes into play due to the conversion of the continuous analogue signal into several discrete values. To avoid this, the ADC can be set up to shift the signal's least significant bit (LSB) to be within plus or minus 0.5 LSB as compared to being between 0 and 1 of the LSB (Kim *et al.*, 2011).

### **3.3.2 Noise in Data Signals**

Measuring systems are prone to noise. The typical noise sources can come from electromagnetic interference, electronic noise, noise from the mains power supply, to noise caused by high-frequency digital circuits. Noise can also be caused by unwanted variables at the physical measurement environment, such as vibrations (Marin-Palomo *et al.*, 2017).

Eliminating sources of noise at the measurement source is most effective compared to elimination before processing. Though certain noise levels are inevitable in an electrical circuit, the levels and vulnerability can be reduced. Some of the noise reduction techniques in data acquisition include proper noise-proof design, use of digital filters, earthing, protection of the signal line through shielding, separation of power and signal lines and use of software-based noise elimination techniques. Software techniques have routines that can detect noise and act upon the signal to guarantee integrity. Such techniques are more flexible, cost less, and suited for low frequency (James, 1997).

After digitizing a data signal, noise can be eliminated using mathematical approaches such as Fourier Transforms to provide a filtered and smooth signal. This takes place in three steps; the signal is subject to a Fourier transform; the signal's amplitude is then multiplied by the desired frequency response in the frequency domain. The signal is then subjected to inverse Fourier transform to restore it to the time domain. A digital filter can

adjust itself to any frequency response eliminating phase errors (Nureni and Yekini, 2015).

### **3.3.3 Data Transmission**

#### **3.3.3.1 Guided Data Transmission**

Guided data transmission uses a cabled medium to guide data signals to a destination. Cabled medium involves binding data in the cabling system throughout the transmission period. Common types of guided transmission included; twisted pair cables consisting of multiple wires twisted together, optical fibre that uses the principle of internal refraction of light and coaxial cable, consisting of a copper core cable and braided conductor noise (Ketheeswaren, 2009). Guided mediums of data transmission are affected by noise, length of cables and environmental conditions which wear off the cables.

#### **3.3.3.2 Wireless Data Transmission**

Data transmission is necessary in modern measuring systems due to decentralization and the need for remote monitoring. This has led to the development of telemetry systems capable of transmitting information in two directions, enabling cross-communication between systems over a long distance. Data transmission follows a set of rules commonly referred to as a protocol that differs from one transmission mode to another. Protocols ensure data is received at a rate that does not overwhelm the transmitter and the required integrity.

Wireless transmission systems use serial data transfer where all the data bits are transferred each bit at a time in a single path chain structure (PM). In serial transmission, bit rate  $R$  is used to refer to the number of bits transferred per second, the maximum being 1200bits per second (James, 1997).

To initiate a wireless communication protocol between two systems, the process starts with handshaking. This is the process where a device indicates its readiness to receive or transmit data or any other state so as to initiate the next action. Handshaking aims to facilitate a synchronized and orderly data transfer. Modes of wireless data transmission in measuring systems include Bluetooth, Wi-Fi, Radio Frequency Broadcasting, Satellite Communication and Cellular Communication (Madhvi *et al.*, 2015).

### **3.3.3.2.1 Radio Frequency communication**

Radiofrequency communication occupies the 3 kHz to 1GHz bandwidth. Transmission occurs through antennas, and propagation is in all directions meaning that any receiving antenna in the range can receive the signal from any other transmitting antenna. Through Frequency Modulation (FM) or Amplitude Modulation (AM), which involves encoding information, radio waves can be used to transmit voice, video and data. Once received, demodulation is carried out to recover the original data in the wave. Radio frequencies are divided into bands that comprise different transmission frequencies. This project uses the very high or ultrahigh band with frequencies of between 30 MHz and 30 GHz (Wireless technologies and the national information infrastructure, 1995).

Radiofrequency is affected by attenuation, noise interference and refraction if travelling for long distances. Despite these challenges, radiofrequency is widely used in radio broadcasting and industrial wireless communication due to its reliability over short transmission distances and cheap infrastructure (Nureni, 2015).

### **3.3.3.2.2 Microwave Communication**

Microwave communication uses the line-of-sight principle where the transmitter and receiver must be visible to each other. Microwaves operate in the 1GHz to 300GHz bandwidth. During operation, the microwave can be focused into a narrow beam. A pair of unidirectional antennas cannot be affected by another pair that is unidirectional. Microwave transmission is used in bulk data transfers for large systems. Microwave communication has several advantages: the antennas are usually short due to the high transmission frequency, has a high data capacity, and the antennas occupy small areas. This communication model has several disadvantages: attenuation by solid objects suffers from reflectance by surfaces, suffers from diffraction around solid objects, and is refracted by the atmosphere (Kirchhoff *et al.*, 2006).

### **3.3.3.2.3 Bluetooth communication**

Bluetooth communication, first developed in early January 2000, revolutionized wireless communication. This communication technology allows two or more synchronized devices to communicate with each other without the use of cables. Operating at a 2.4GHz frequency spectrum, Bluetooth can transmit data at the rate of 172kilobites per second. In a Bluetooth communication setup, the hosting device is normally referred to as the master while the connecting device is referred to as the slave.

A network of Bluetooth devices or systems is referred to as a piconet; two piconets form a scatternet, essentially a cluster of Bluetooth networks. Communication in Bluetooth starts with an inquiry to establish identity, paging to form a connection and then a full connection where the data transfer takes place (Madhvi *et al.*, 2015).

### **3.3.4 Digital Signal Processing**

Upon reception, a digital signal is subjected to the processing according to user needs. Digital signal processing starts with noise elimination; after this a signal is subjected to non-linear processing. In non-linear processing, the data obtained from ADC is subjected to nonlinear operations to convert the data to the respective domain. Domains in digital signal processing include time domain, space domain and frequency domain. Time-domain represents the signal in a one-dimension format while space and frequency domains present signal in multi-dimensions (Bora and Sarma, 2012).

To present a signal in time or space domain, digital filters are used, which are usually designed using mathematical models. Frequency domain signals are obtained by subjecting the signal to a Fourier transform which converts time and space into magnitude and direction each respective frequency processed.

### **3.4 Nibras Data Acquisition Systems**

The Nibras data acquisition system is an instrument used to acquire data and treatment during gamma scanning. It has been helpful since 2013 and is associated with software for processing data obtained during scanning exercise. This system can run in count mode or soft mode. The count mode uses a rotary switch where you can set the count time between 6 seconds to 10 minutes. In the backside, it has two ports for BNC connectors for detector and signal checking. The front side has a pushed bottom and an LCD display to show the radiation intensity for each step during scanning process (Benahmed and Alami, 2012). It consists of four batteries of 1.5V, after which the system cannot run more until recharged. Two batteries of 1.5V are used to supply a 3V converter for direct current (DC), as shown in Fig.3.5.

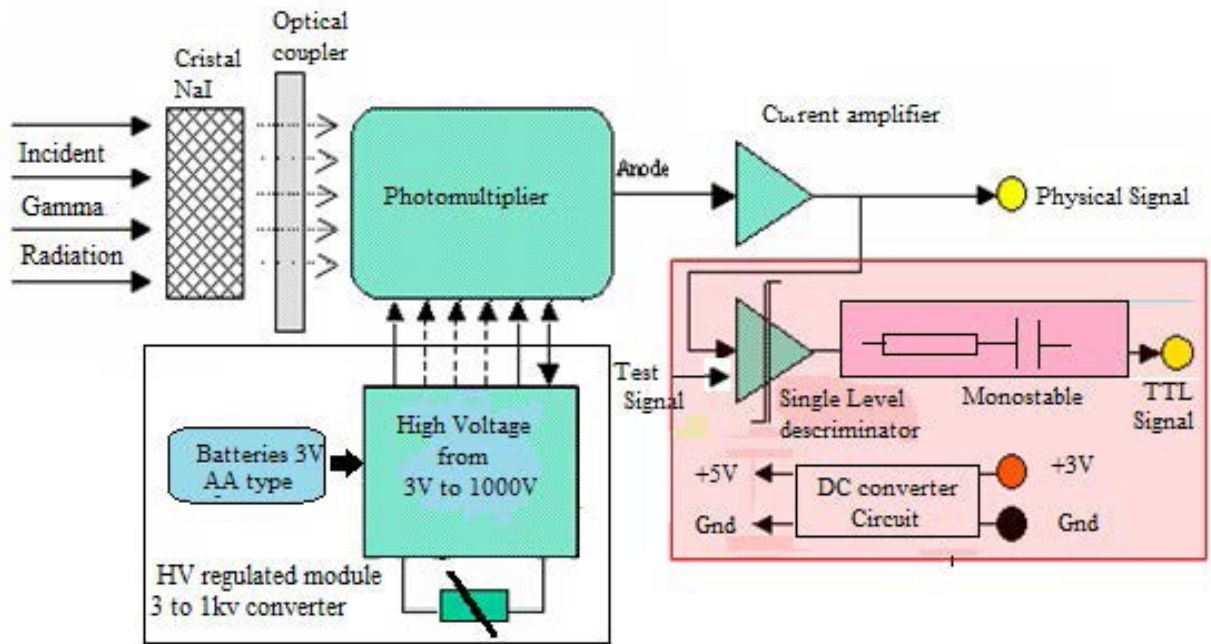


Figure 3.5: Nibras System circuit (Source: Benahmed and Alami, 2012).

The high voltage output of the converter and is used to power the detector. This data acquisition system is based on a microprocessor at 20MHz (Bora and Sarma., 2012). It has a Liquid crystal display connected to a notebook, as shown in Fig.3.6.

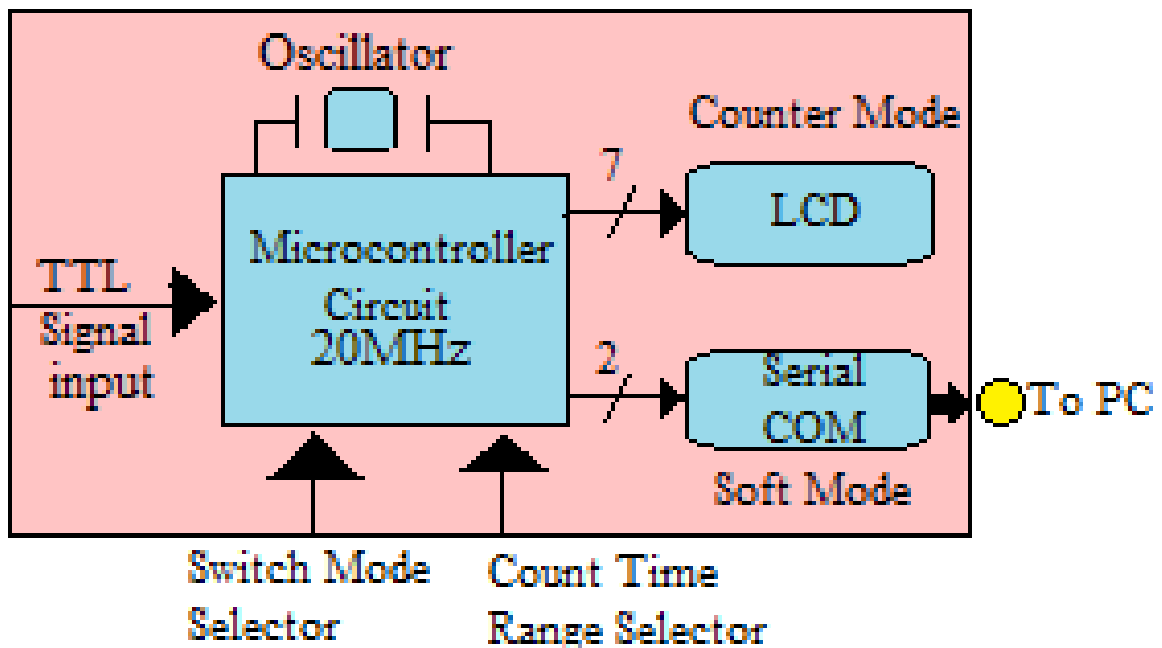


Figure 3.6: Nibras System display (Source: Benahmed and Alami, 2012).



A clock module is incorporated to control the acquired data. The system is connected to a PC using coaxial cables, as shown in Fig.3.7 from which it is operated while the detector and source is moved simultaneously by two individuals. If there are no power outlets in the column, it becomes hard to set the scanning and cumbersome.

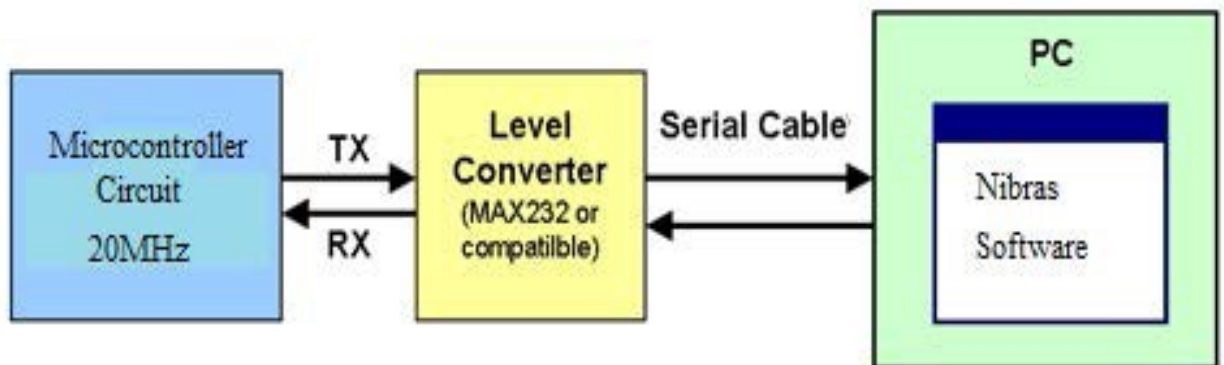


Figure 3.7: PC connection for Nibras system (Source: Benahmed and Alami., 2012).

From the Nibras system, it is evident that the system cannot run without the PC connected since the software which initiates the data acquisition is installed in the PC. It therefore becomes easy for the scanning process if the persons doing the scan on top of the column are the ones to initiate the acquisition system. Also, to reduce the long cable connections between the computer, data logger and detector, it could be necessary to have wireless transmission of the acquired data during scanning exercise. This makes the acquired data safer and fast in terms of transmission and reduces mistakes due to poor coordination between the expertise.

## CHAPTER FOUR

### METHODOLOGY

#### 4.1 Chapter Overview

This chapter discusses the development of the system from design to its operation. It is divided into several sub-sections. Section 4.2 brings out the design and construction layout of the system and discusses the conceptualized initial model of the scanner, and the final model of the scanner developed. Section 4.3 discusses the components used and the various circuits and component connections.

#### 4.2 System Design and Construction

In this study, the design and construction of the automated gamma scanning system prototype model approach assumed, as shown in Fig. 4.1.

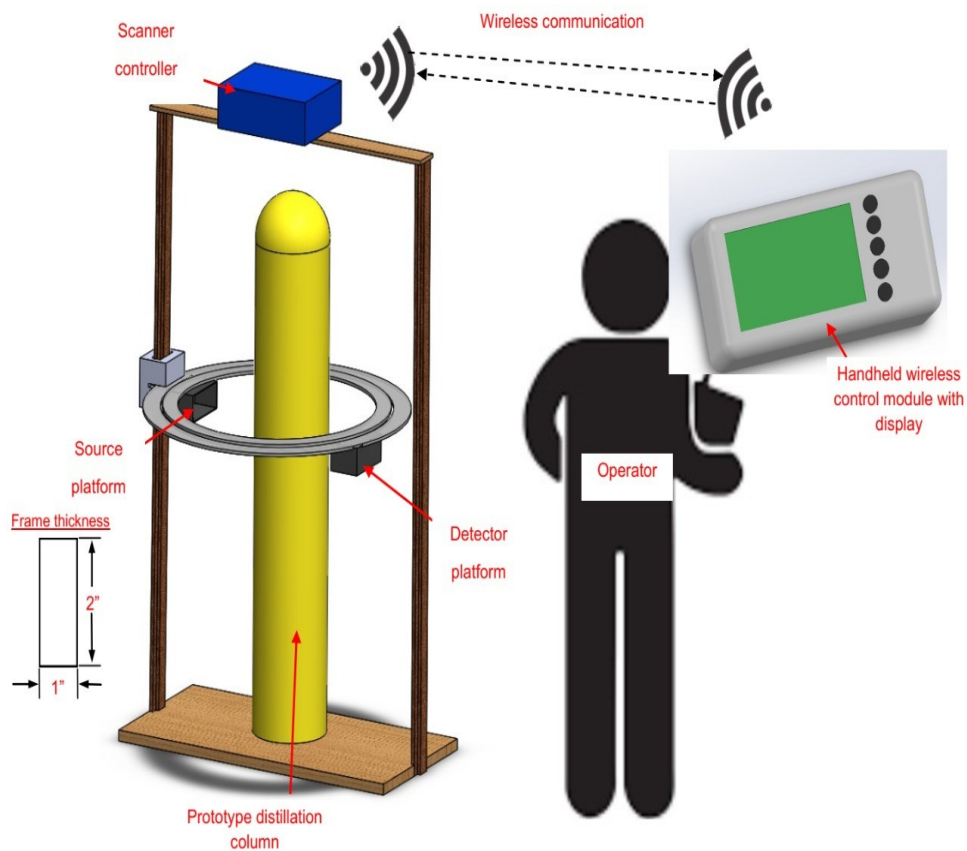


Figure 4.1: Outlook of the conceptualised automated gamma scanning system prototype.

The system consists of three modules: the scanner head, which consists of the components assembled for control of detector-source dual-axis movement and scanning; the communication module, which consist of the radio frequency transmitters and receivers and the controller module. The system configures the scan input parameters for display of scan profiles, as shown in Fig.4.2.

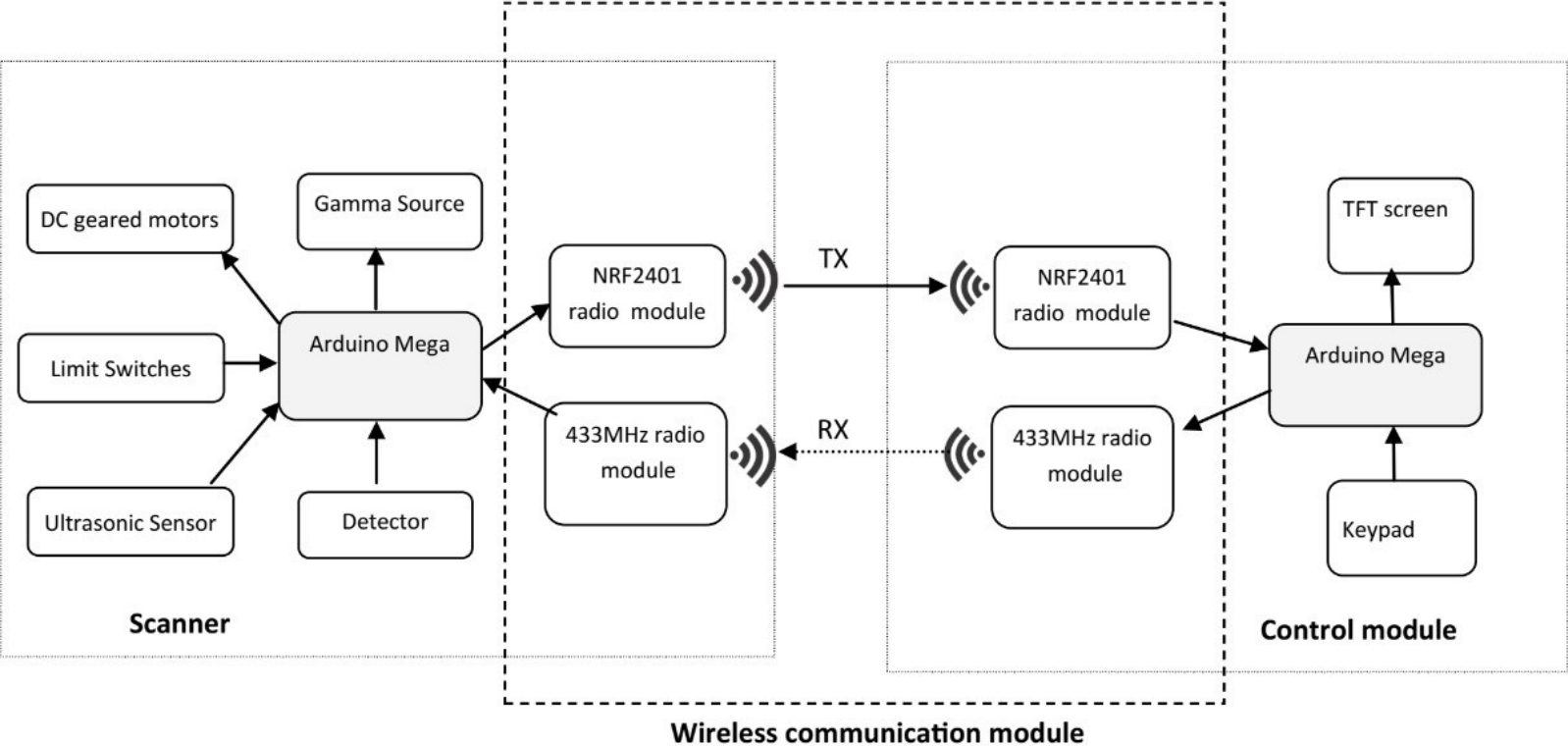


Figure 4.2: Block diagram of the designed gamma scanning system prototype.

The Scanner head module consists of three main functional units; source-detector unit, electromechanical movement drive unit and the communication unit and operated by two 12 VDC geared motors. All these units are connected to the Arduino Mega for control and scanning. The drive unit consists of a belt-driven motor system for vertical and horizontal plane movement and an ultrasonic sensor used to determine vertical distance. The communication modules are used to transmit bi-direction scan data between the controller module and the scanner module.

The control module consists of an LCD display screen (3.5" x 2.5") for displaying scan parameters and results. It has a five-button keypad for the input of commands and a pair of 2.4 GHz nRF12401 radio modules for transmission and receiving radio units. Two radio modules reduce latency, making the system as accurate as possible since no data is lost during acquisition and transmission.

The physical system was modelled using SolidWorks® design software to conceptualize its operation. Fig. 4.3 shows the initial conceptualized design of the scanner head, which was later redesigned to add a second stand in the support frame to distribute the weight of the scanner equally.

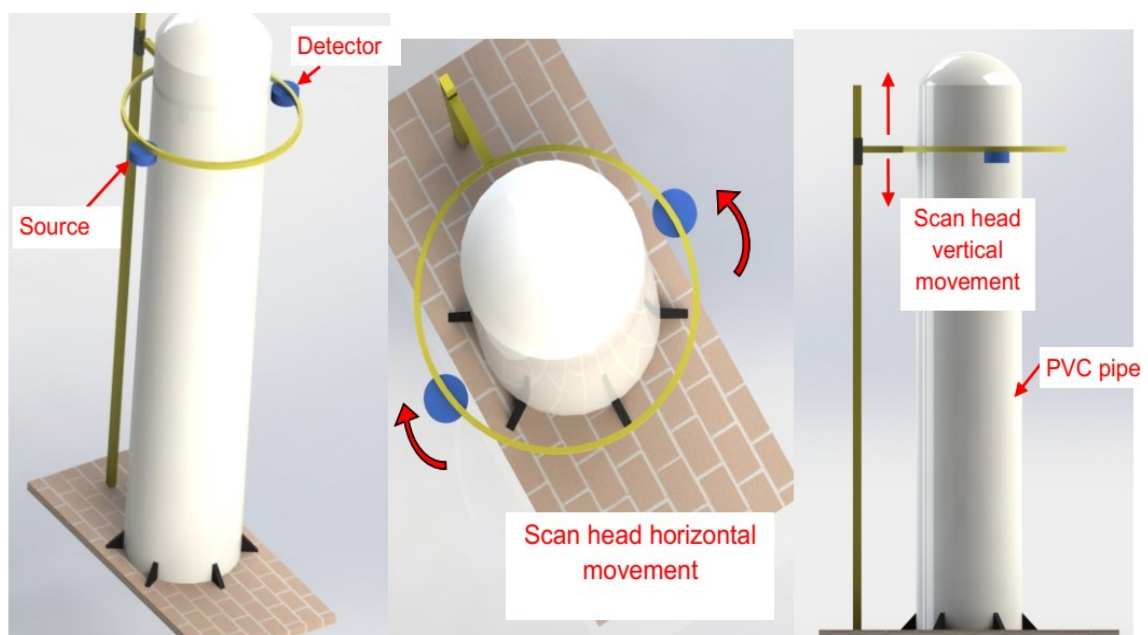


Figure 4.3: The conceptualised graphical model of the physical system.

The actual frame constructed used to support the Scanner unit is shown in Fig. 4.4. It was constructed using a 2"×1" wood frame, 1 m in length and 0.5 m in width.

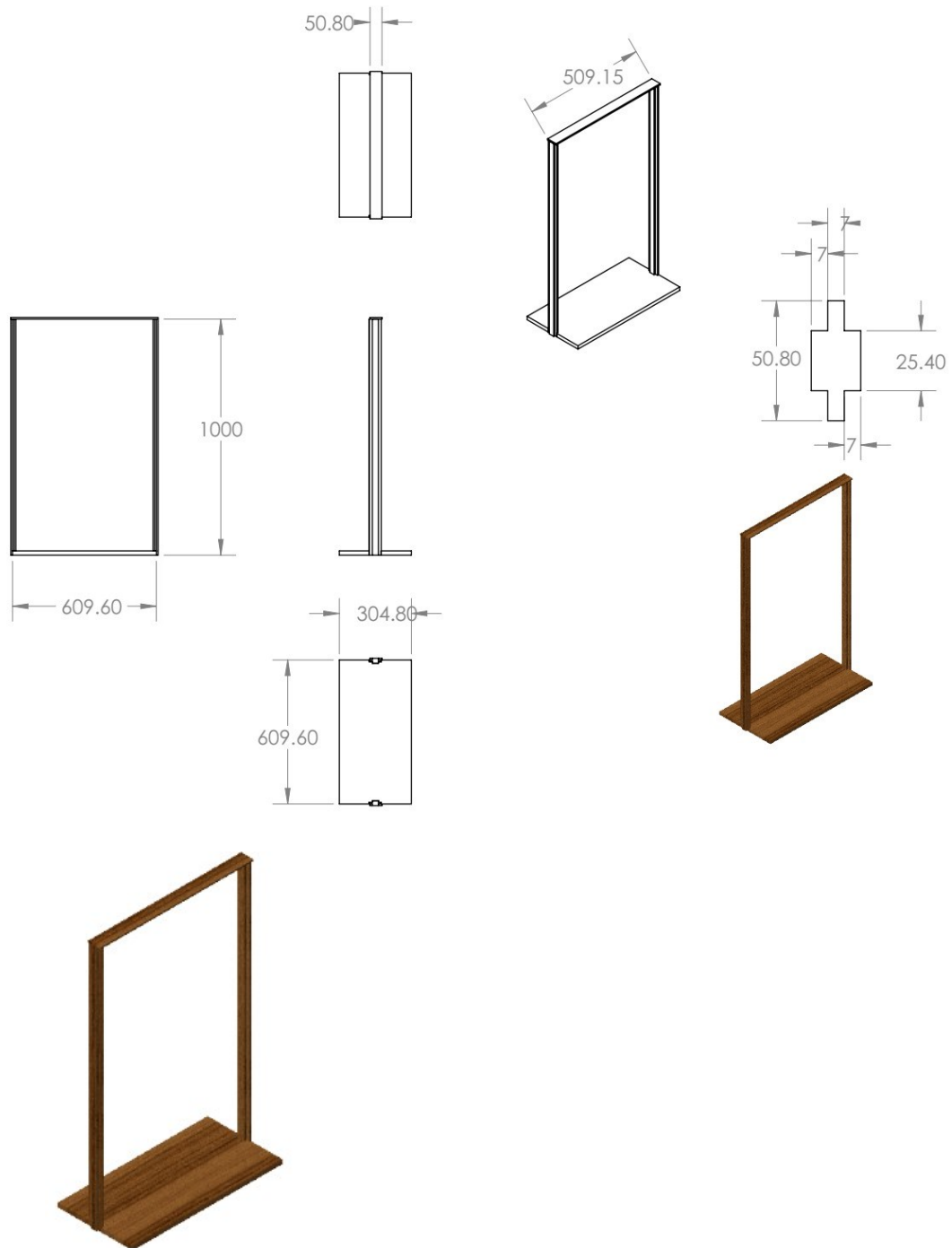


Figure 4.4: Scanner frame assembly unit.

### 4.3 Components used in the designed system development

To transform the conceptualized system into a real functional system, the following materials and components whose specifications are highlighted in detail were used in this study. Figs. 4.5 and 4.6 show the schematic diagrams for the Scanner head and handheld control modules respectively. They represent the power and signal circuitry interconnections of the components used in the system. Since the components consume different power, they are supplied each with the required voltage supply.

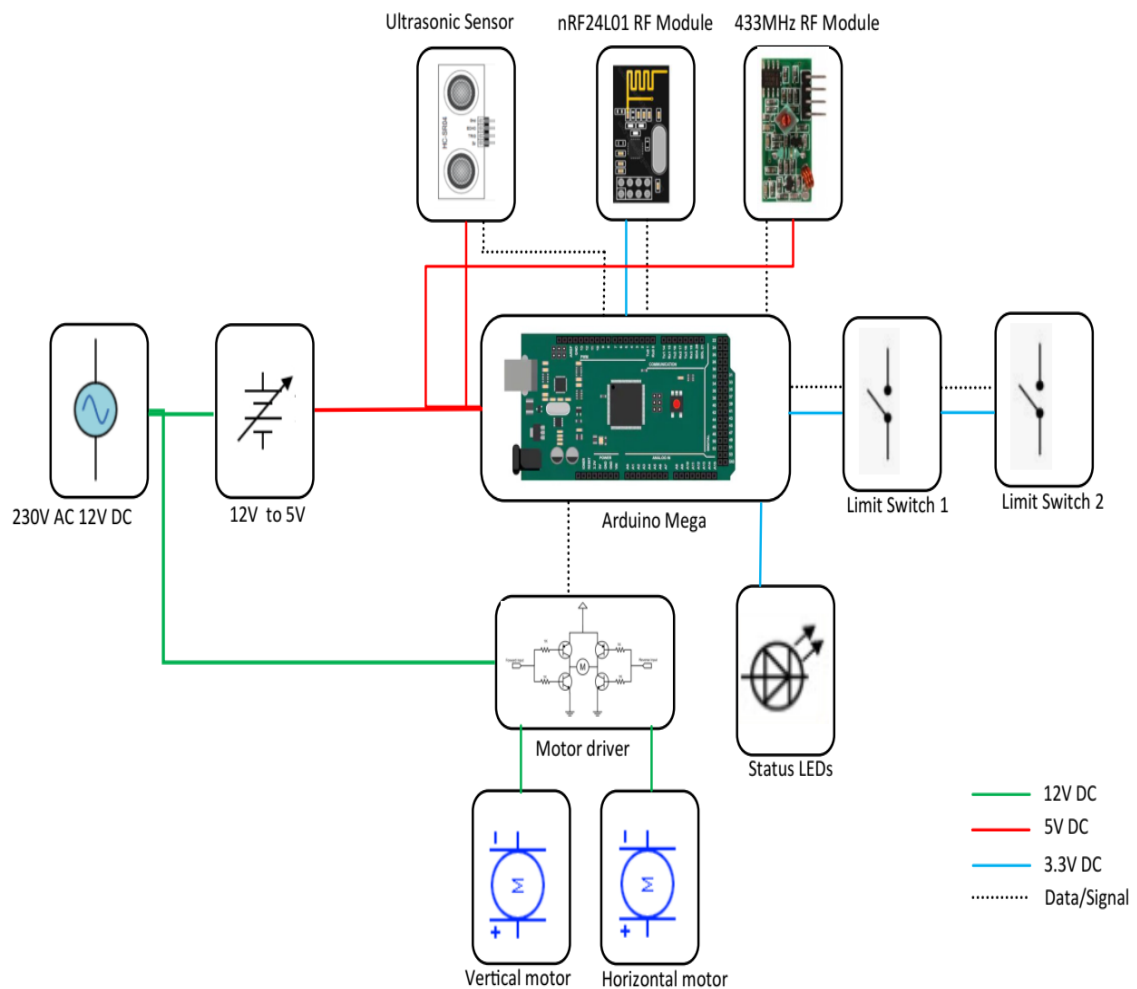


Figure 4.5: Schematic diagram of scanner head module components.

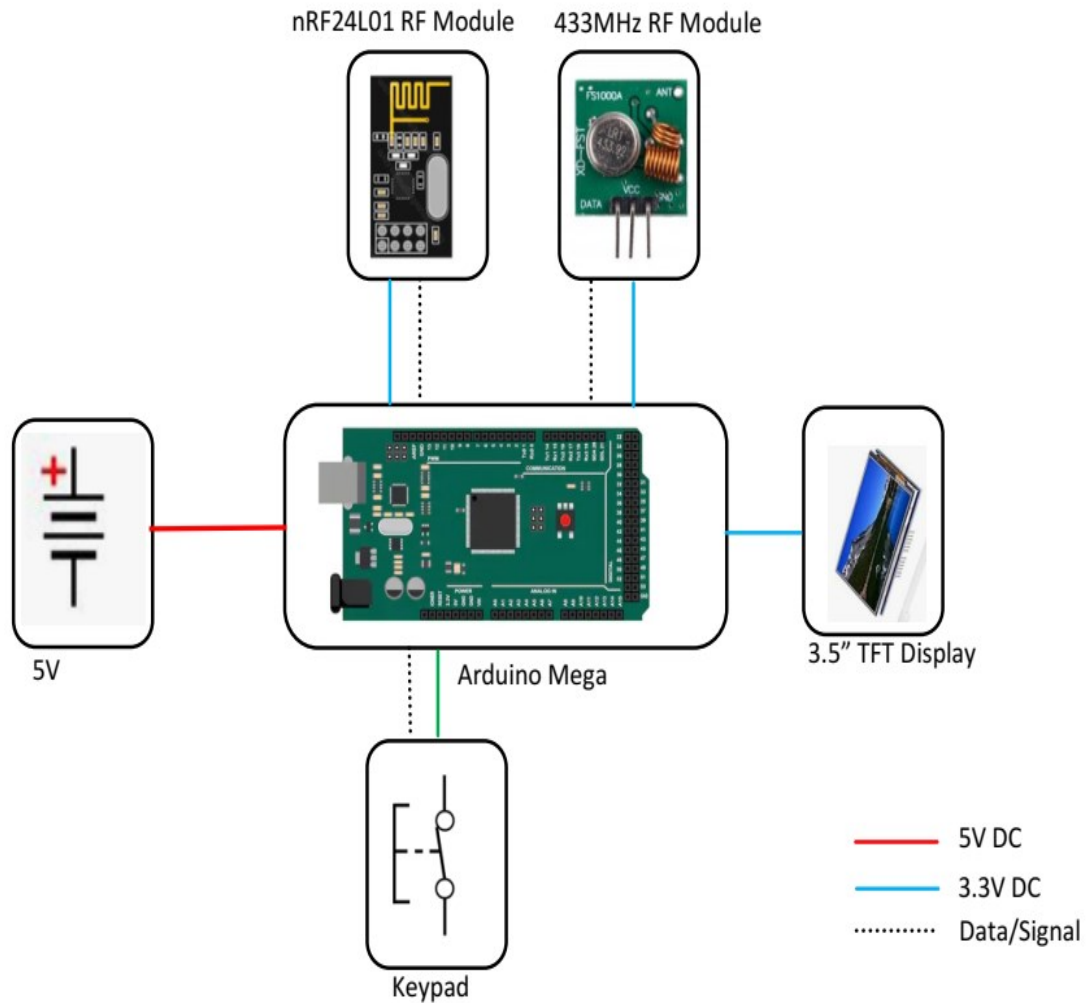


Figure 4.6: Schematic diagram of handheld control module components.

### 4.3.1 Arduino Mega

Arduino being an open-source microcontroller hardware and software platform was preferred in this project due to its easy use. It was programmed to function as a standalone system able to communicate with another system. It was capable of reading inputs and controlling output parameters. The board was assembled on a prototyping board, and its programming environment downloaded. Arduino Mega used is based on an Atmega2560 microcontroller chip (see specifications in Table A1.1 in Appendix I).

### 4.3.2 DC Geared Motors

In this project, two DC motors were used. They are made up of an armature whose movement was initiated by poles created by supplied power via brushes at one end. They were capable of producing either high-speed rotations or high torque rotation or in

combination depending on the need. To increase the motor's torque, a series of gears was connected to the motor's output shaft making the motor geared. This gearbox reduced the motor's speed to a manageable value while increasing torque (Warren, 2011). The speed reduction for the DC geared motors was in a gear ratio of a 100:1, which means that the motor output shaft spin 100 times for the gearbox output shaft to make one complete revolution. These DC motors were used to move the scanner setup and down and in a circular motion. Based on  $T_L$  value of 1.5kg.cm, a 12V DC Geared Motor with 100rpm and 2Kg.cm torque was selected for the project, providing a safety margin of 0.5Kg.cm torque.

#### **4.3.3 A 3.5" Thin Film Transistor LCD Screen**

The choice of the display screen used in this project was based on the need for a TFT module that is compatible with an Arduino Mega and large enough to display a graph with clarity. This led to the selection of the 3.5" TFT LCD screen, which operated on 5V DC, has a micro-SD card module and has a resolution of 480x320 with a refresh rate of up to 50Hz.

#### **4.3.4 nRF2401 Radio Module**

An nRF2401 radio module was operated at 2.4GHz-2.4835GHz ISM radio band. To initiate communication with this radio module from a microcontroller, Serial Peripheral Interface (SPI) was used based on the module's operation modes as per available configuration registers based on the First In First Out order (Mahbub, 2019). The module's baseband protocol was based on packet transfer communication which supported several modes like manual operation and advanced autonomous mode. For modulation, the nRF24L01 module used the GFSK module, which allowed the user to configure several parameters such as data rate and frequency channel output. This module was capable of transmitting data at a rate of 2Mbps and concurrently executing Cyclic Redundancy Checks to detect data transmission errors. (See specifications for the radio module presented Table A1.3 in Appendix I).

#### **4.3.5 433 MHz Transceiver and Receiver Pair Module**

This module consisted of a radio frequency transmitter and receiver, which operated as a pair at 433MHz frequency using Amplitude Shift Keying. To transmit data, the transmitter subjected the data through an HT12E encoder before transmitting. This encoder converted parallel data into serial for transmission. The receiver then decoded the



data received using an HT12D decoder into signal data which was easy to process. (Ahmed *et al.*, 2006). Specifications for the 433 MHz transmitters and receiver are presented in Tables A1.4 and A1.5, respectively in Appendix I.

#### **4.3.6 MPU6050 3- Axis Accelerometer**

An accelerometer uses electromechanical interactions to determine plane acceleration forces. The MPU6050 3-Axis Accelerometer used, contained a proof mass and sensing plate with support circuitry. As the acrylic assembly changed orientation, the proof mass moved and consequently causing a change in capacitance due to gravitational pull, which was amplified by the support circuitry to generate a voltage signal. A I2C communication protocol was used for communication between the accelerometer and the microcontroller to transmit digitized outputs of the resultant orientation value. Values for X (roll), Y(pitch) and Z(yaw) axis orientation can be obtained from the digitized values (ECEE, 2020).

#### **4.3.7 Ultrasonic Proximity Sensor**

Based on the radar principle, an ultrasonic sensor was used to determine the proximity of the detector-source in vertical movement, along the column height during gamma scanning operations. This principle involved the transmission of a signal at a high frequency and determining the distance depending on duration of the echo. The sensor reading was then used to generate the scan profile. The frequency used is usually beyond the audible range by human beings. The Ultrasonic sensor used could measure up to distances between 2 cm and 500cm with high accuracy.

#### **4.3.8 Power supply**

Two power supply modules, namely, 230V AC to 12V DC stepdown power supply and 12V to 5V DC buck converter, were used in this project. The step-down power supply is made up of a stepdown transformer, power conditioning capacitors, diodes, and variable potentiometer to adjust the output. The buck converter based on LM2596 power supply IC stepped down supplied voltage from 40V DC to a variable value between 12V and 5V DC. This was suitable for supplying the various loads in the system (see Table A1.6 in Appendix I).

#### **4.3.9 Velleman K2645 Geiger Muller Detector**

The Velleman K2645 GM kit was used for detecting nuclear radiations. It has a buzzer that beeped whenever radiation was detected; more beeping indicated high radiation

levels. This kit was powered by a 9V battery and had a current consumption of  $200\mu\text{A}$ . Incident radiation rays on the GM tube caused a high current which biased the transistor circuit producing a beep. For this project, a voltage divider circuit was added across the buzzer circuit to form the detector circuit (Fig. 4.7).

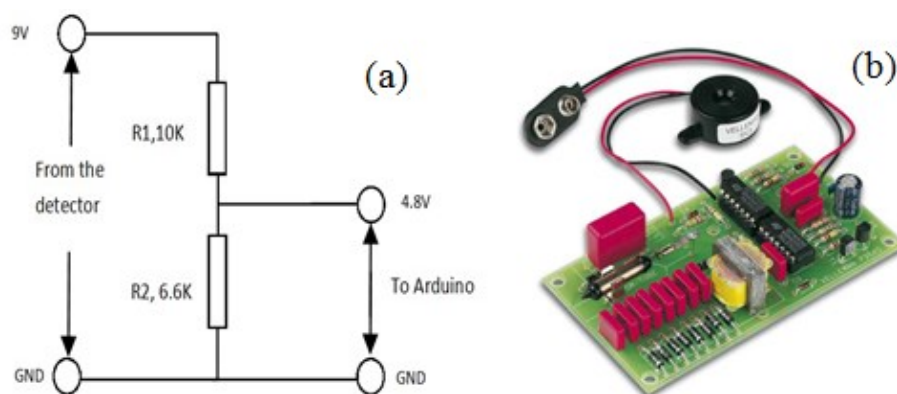


Figure 4.7: (a) Voltage divider schematic, (b) Velleman K2645 GM (Source:<https://www.velleman.eu/products/view/?country=nl&lang=en&id=9091>).

#### 4.3.10 DC Motor driver

An L298N driver module was used in this project to control the rotation speed and torque in DC motors. Using Pulse Width Modulation, which involves controlling the on and off duration of a power pulse, the required speed and torque of the DC geared motor used in this project, was achieved.

#### 4.3.11 A Limit switch

A limit switch is a simple switch that is used as a position sensor when the electrical contacts are closed. Limit switches are made in two types: the Normally Closed (NC) type, where the contacts are closed when not pressed and the Normally Open (NO) type, where the contacts remain open when not pressed. Use and application vary depending on the type of switch and whether it has a lever or not (Warren, 2011). In this project, a normally closed switch was used to limit the detector-source assembly in the vertical movement.

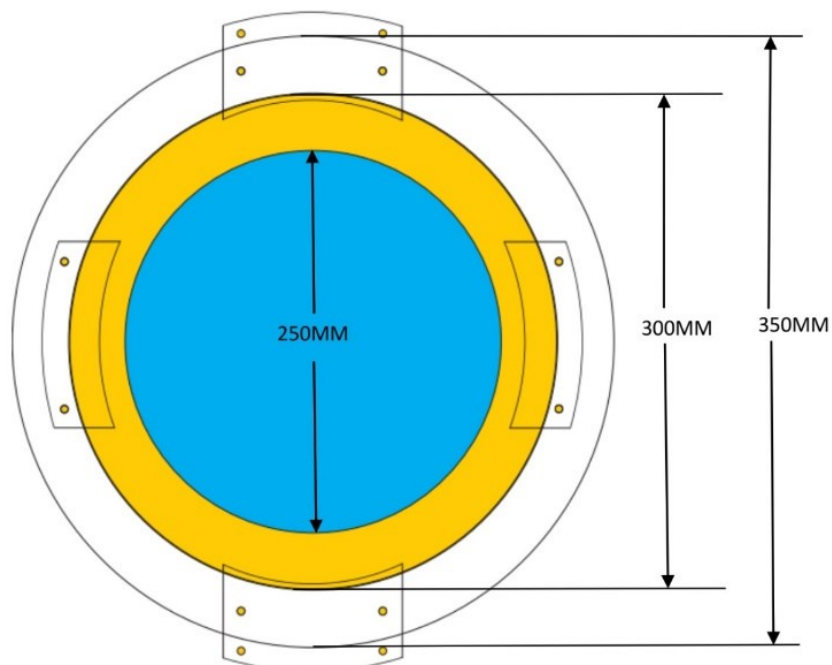
#### 4.3.12 Circular acrylic detector-source assembly

The detector-source assembly is a circular acrylic material (dia 25 cm and radial width 10 cm), that houses a low activity source, 114 mCi  $^{241}\text{Am}$  radiation source (NER-492, S/N: A-377) and Velleman K2645 GM detector. It was designed to slide along the vertical axis

guided by the vertical system support frame and consists of a hollow cuboid with rollers for smooth movement (see Fig. 4.7).

For source-detector rotation movement along the horizontal axis, the setup uses acrylic material for support, which was cut to precision using a laser CNC machine. The scanner is operated by two 12 VDC geared motors and two-timing belts to drive the source-detector assembly, in both vertical and horizontal plane, along with the column frame for preset increment heights of 5-10 cm movements for auto counting at a preselected preset time between 1-10 s.

The scanner head consists of the components assembled to control the detector –source dual-axis movement and scanning. Other features include; Ultrasonic sensor, 3-axis accelerometer, source and GM detector, all connected to a microcontroller-based circuit housed inside air-cooled ABS plastic inspection box and mounted on top of the scanner's frame.



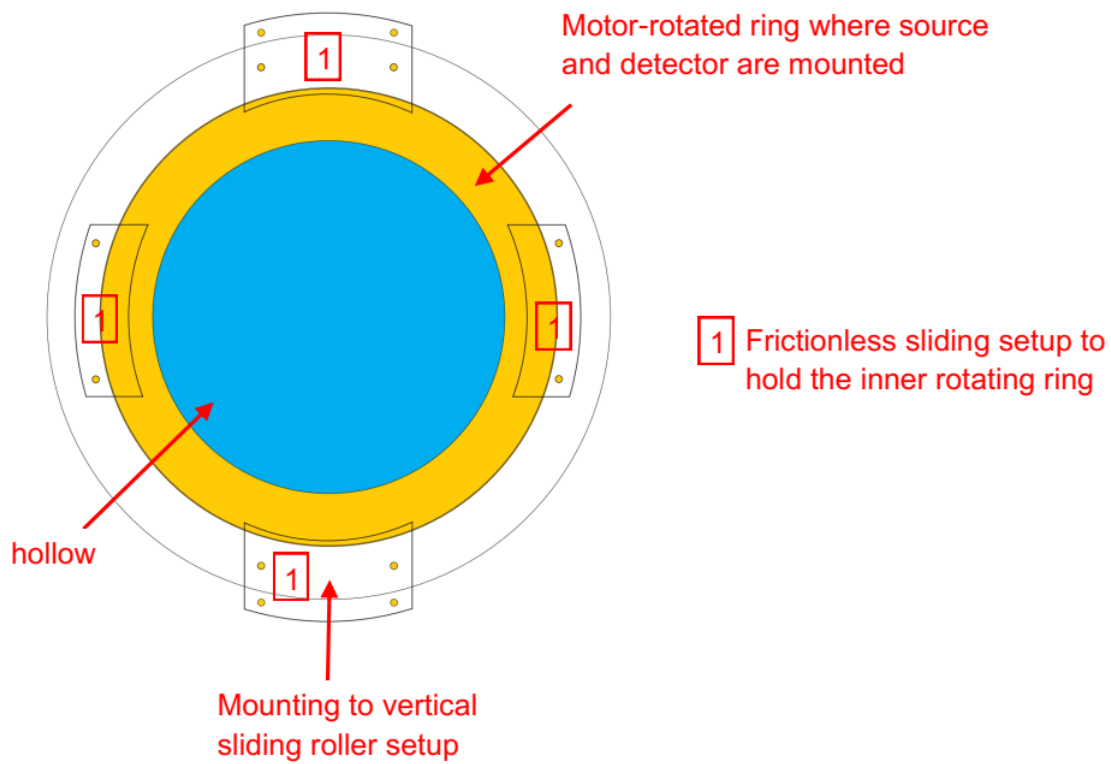


Figure 4.7: Design sketch of the Circular acrylic assembly.

## **CHAPTER FIVE**

### **RESULTS AND DISCUSSION**

#### **5.1 Chapter Overview**

This chapter discusses the results obtained in several sections. Section 5.2 discusses the hardware structure of the developed prototype system, software development and the system validation and reliability. Section 5.3 elaborates the system configuration for Gamma column scanning and the scanned profiles and closes the chapter with radiotracer configuration and the scan profile.

#### **5.2 Overview of the developed prototype system**

##### **5.2.1 Scanner head system**

The Gamma Scanner head prototype was constructed using a wood frame; 1 m length and 0.5 m width and a circular acrylic material (dia 25 cm and radial width 10 cm) that houses a low activity source, 114 mCi <sup>241</sup>Am radiation source (NER-492, S/N: A-377) and Velleman K2645 GM detector as shown in fig.5.1. The scanner is operated by two 12 VDC geared motors and two-timing belts to drive the source-detector assembly, in both vertical and horizontal plane, along with the column frame for preset increment heights of 5-10 cm movements for auto counting at a preselected preset time between 1-10s.

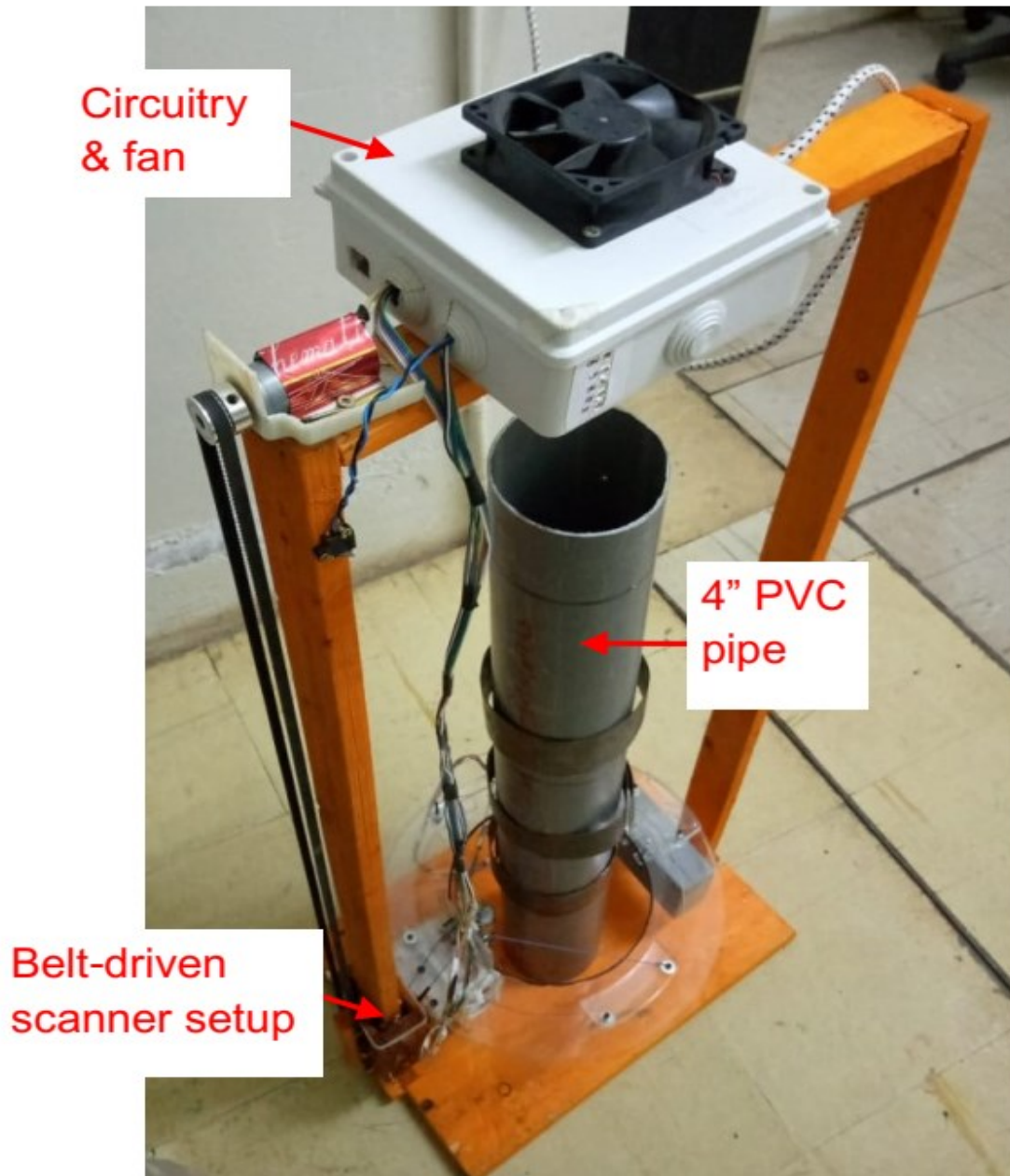


Figure 5.1: Constructed Scanner Head Features assembly – top view.

The GM detector was connected to a microcontroller-based circuit, as shown in Fig.5.2, which also controls the movement motors. Four 2.4 GHz nRF12401 radio modules facilitated wireless communication between the scanner and the handheld module within a radius of 100 m.

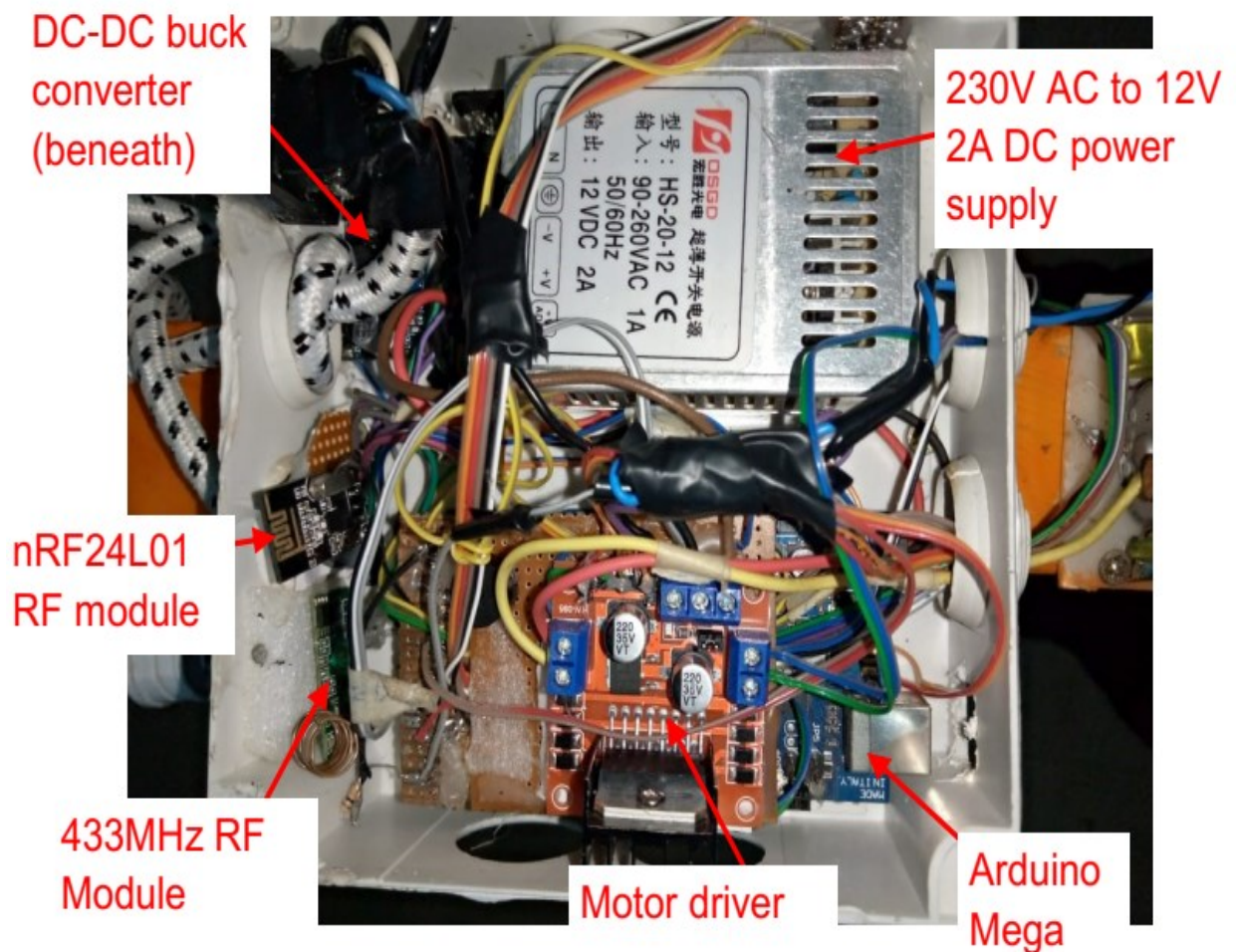


Figure 5.2: Scanner Head circuitry component layout.

### 5.2.2 Control Module system

A handheld controller module consists of an LCD screen (3.5" x 2.5"), Arduino Mega, 5 tactile push-button keypads, and an interface system. An nRF24L01 module for receiving data and a 433MHz module was connected to the Arduino to create the Radiofrequency transmission system, as shown in Figs 5.3 - 5.6. The control module assembly was housed in an ABS plastic enclosure (5"x3") and aesthetics with an extruding battery power connection USB port on the shorter side.

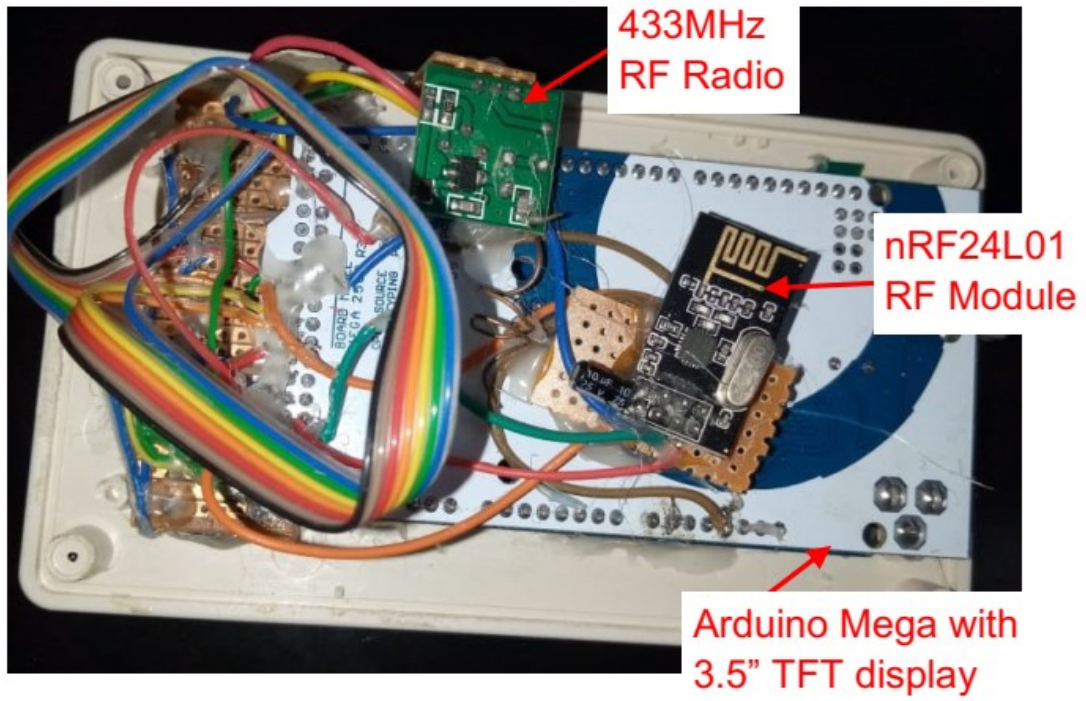


Figure 5.3: Wireless controller circuitry component layout.

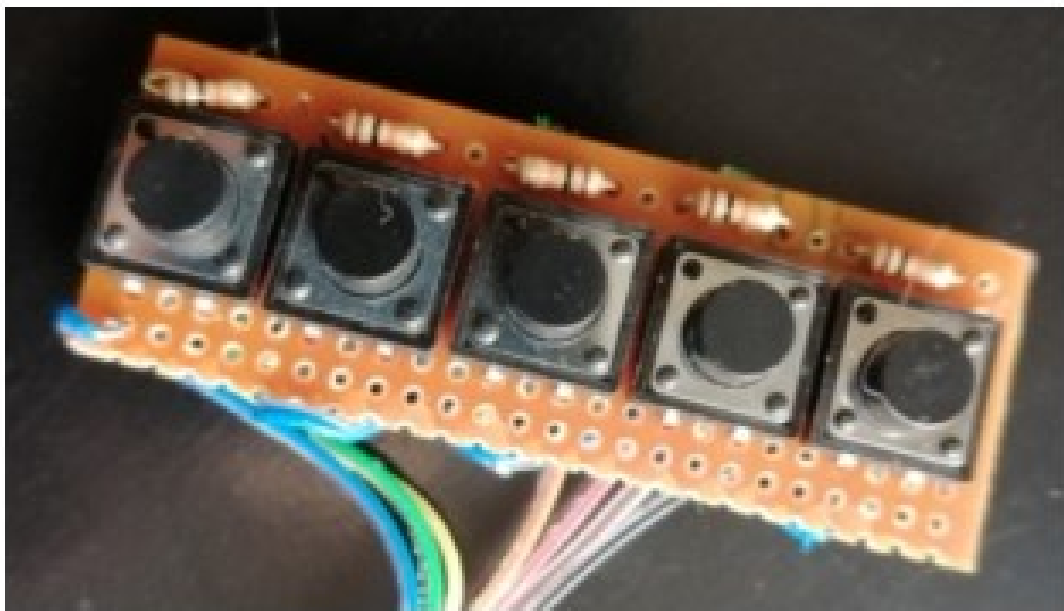


Figure 5.4: Keypad buttons component layout.



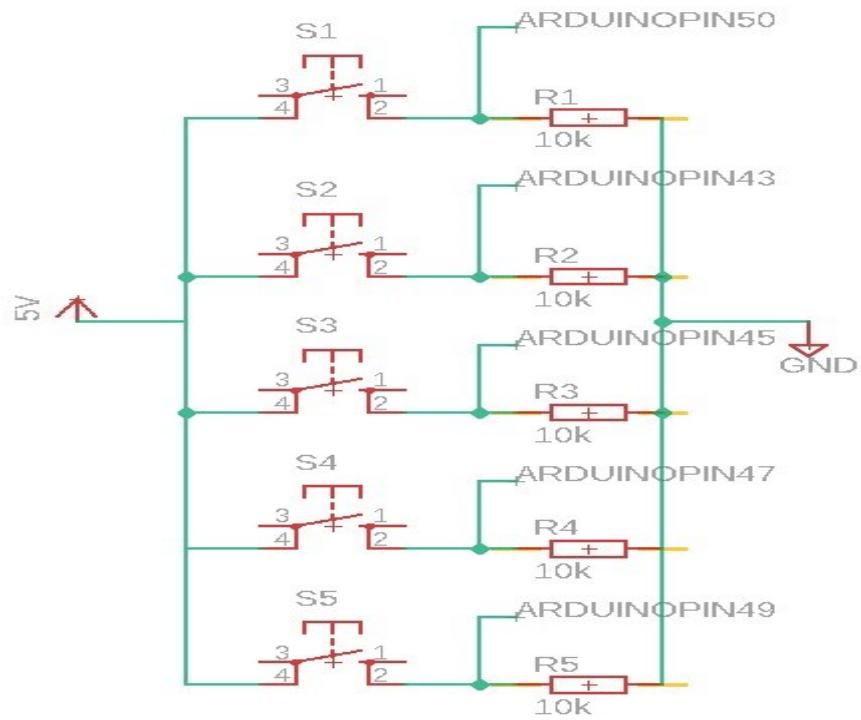


Figure 5.5: Keypad buttons electrical design circuit.

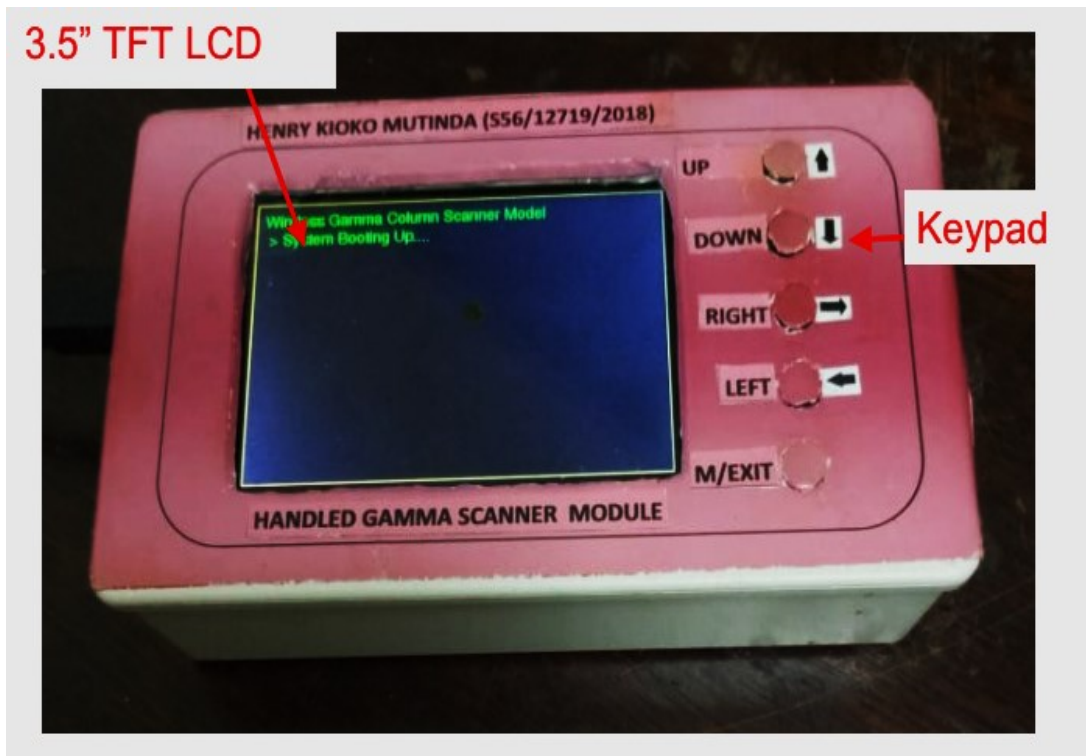


Figure 5.6: Constructed Wireless Handheld Control module.

### 5.2.3 Software Development

The system's software development was done for the; scanner and for the controller modules. The development of this code was done using the Arduino Integrated Development Environment, an open-source software using C++. For the scanner, the state-machine coding technique was used. The system's operation was configured into states which are dependent on user predetermined actions and user commands as shown in Fig. These states are described as follows:

- 1) **eFirstTime**: This is a type of gamma automatic mode that is executed the first time the command is received. In this state, the scanner is moved to the home position before scanning starts and all parameters are reset as in Appendix I.
- 2) **eCyclic**: This is a gamma automatic scanning mode that repeatedly executes the scan based on travel distance and pause time. During this state, scan parameters are transmitted to the wireless controller for plotting, as in Appendix II.
- 3) **eMovingHomeDown**: When the scanner moves to the bottom home position, 10cm from the base as in Appendix III.
- 4) **eMovingHomeUp**: When the scanner moves to the upper home position 70cm from the base as described by appendix IV.
- 5) **eIdle**: When the system is done with any particular scan, and no command is received (Appendix V).
- 6) **eGammaManual**: Executes gamma manual mode commands of movement and transmitting scan parameters to the remote for plotting, as shown in Appendix VI.
- 7) **eRadioTracer**: When the user configures the system in radiotracer operation mode as shown in Appendix VII.

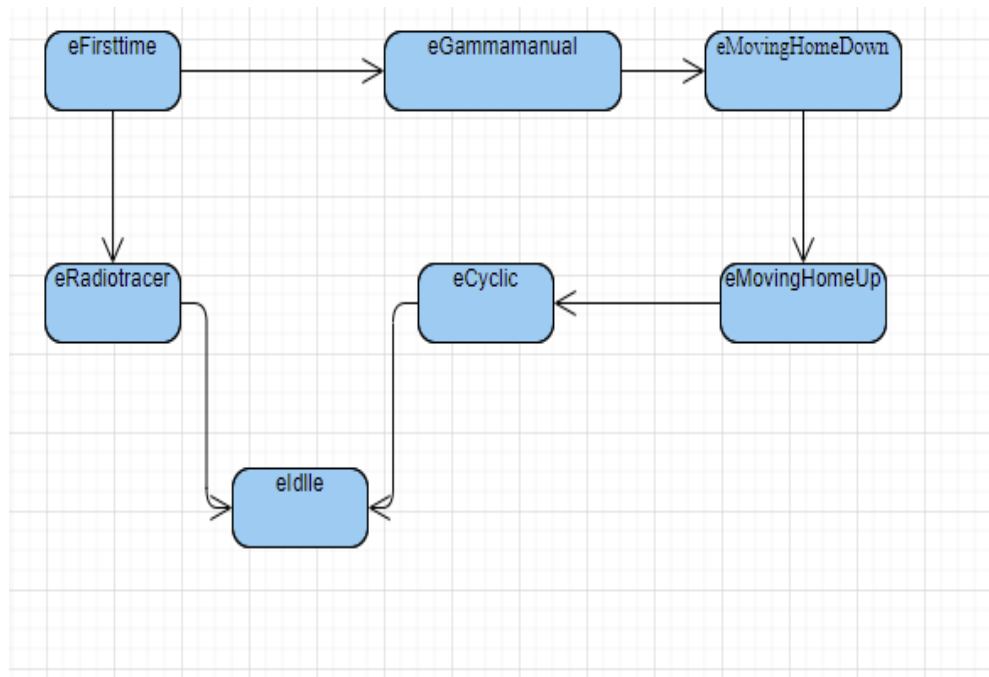


Figure 5.7: State machine coding flowchart for the developed system.

In the background mode, angle, proximity, and radiation sensor readings are sampled into arrays and smoothed by averaging to remove errors and in readiness for transmission when a mode requires data like radiotracer gamma mode. Radio commands are scanned every 1/16th of a second to ensure no command is missed and if any command is received, it is analyzed and executed. Fig. 5.8 describes the flowchart of software execution for the Scanner module /Control module / User interaction.

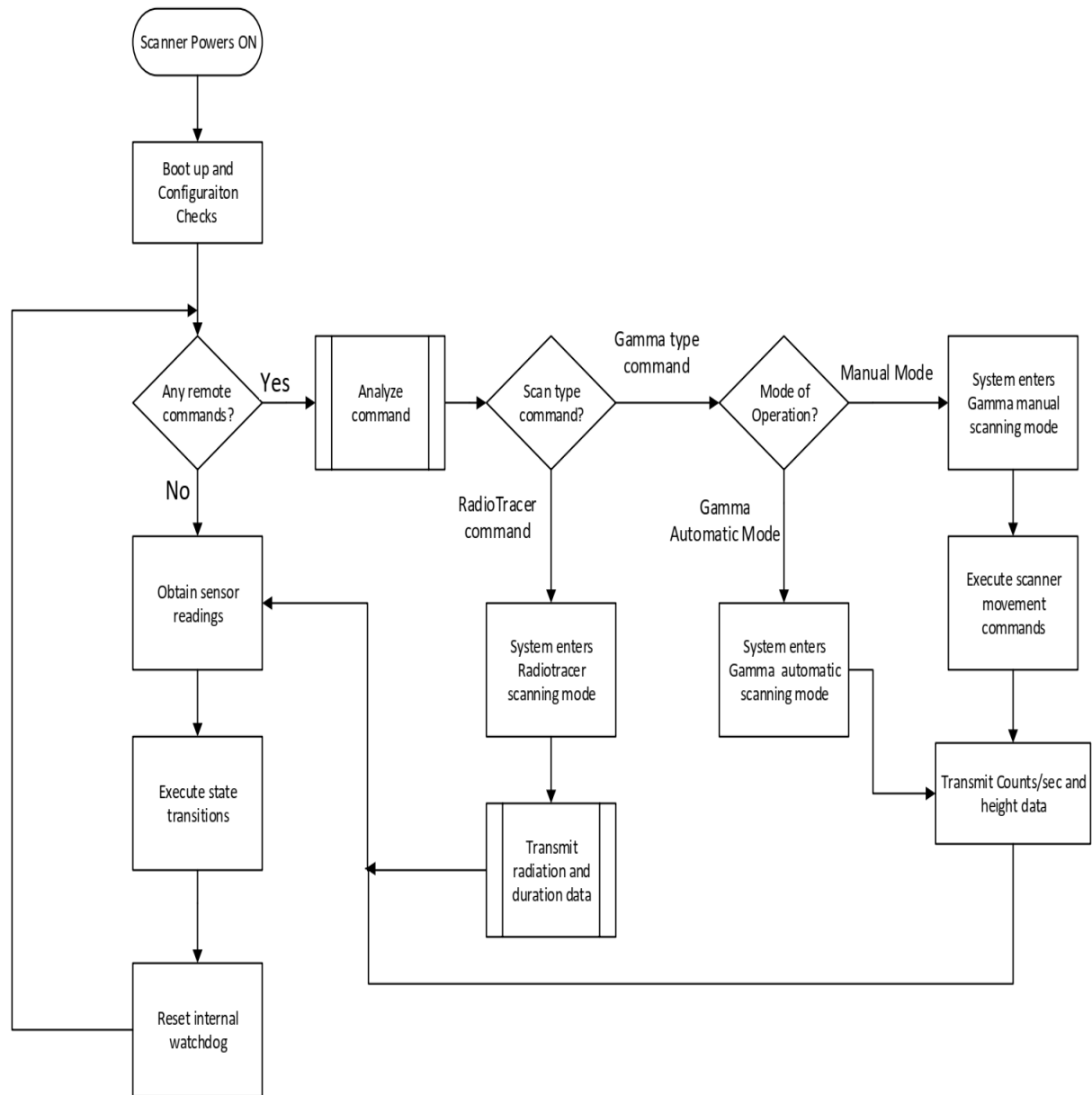


Figure 5.8: Software execution for scanner head/control module/user flowchart.

For the handheld wireless controller, the commands for checking user input via the keypad, were programmed to be executed in a loop. This enabled transition into various screens where parameters were captured and transmitted to the scanner and scan parameters received, analyzed, and plotted. Received scan data was also logged to the SD card for future retrieval. Fig. 5.9 describes the flowchart for the Control module / User interaction.

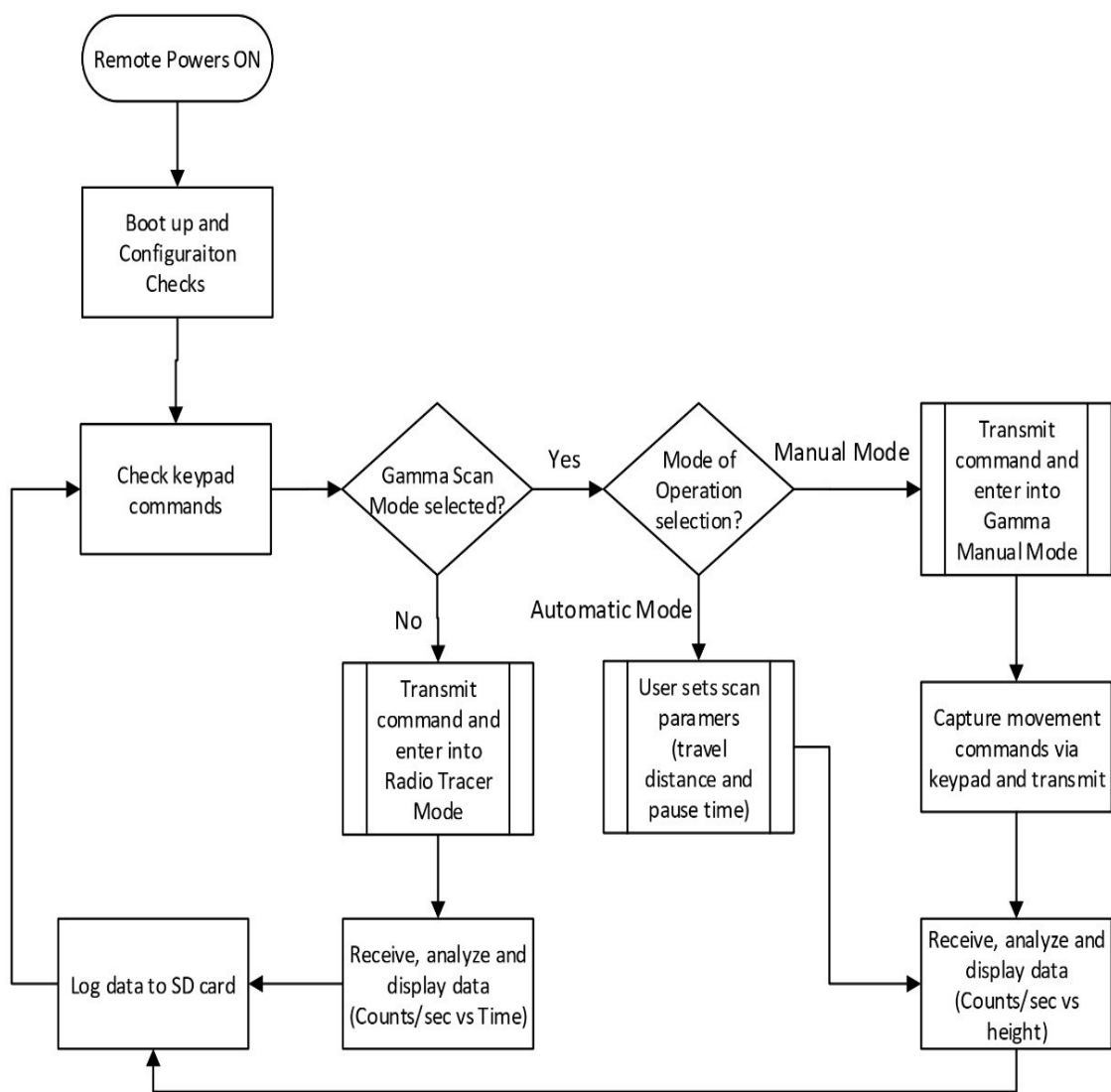


Figure 5.9: Software execution for operator/control module flowchart.

#### **5.2.4 Scanner Data Acquisition**

The scanner head movement was operated by two 12 VDC geared motors and two-timing belts to drive the source-detector assembly in both vertical and horizontal plane and the column frame for preset increment heights of 5-10 cm movements for auto counting at a preselected preset time between 1-10s. For the detector, the voltage divider was connected to an interrupt pin on the Arduino where pulses were detected, initiating an interrupt service routine which added up the number of pulses and stored these values in a buffer.

The Scanner position readings were sampled every  $\frac{1}{16} \text{ s}^{-1}$  and stored in an array that held 16 values, irrespective of orientation. For transmission or execution of a command, an average of the current 16 values was done, returning a 'smoothened' value.

#### **5.2.5 Scanner Data Transmission**

After acquisition for a preset time, the stored buffer data was serialized and then converted into a character array that is then automatically transmitted via the nRF24L01 module to the wireless controller. The procedure is implemented using specific software codes developed in C++ for the Arduino platform (Fig.5.10). For example, DS:100.0; RD:150; AG:30.0; to represent a height of 100 cm, 150 counts per second- and 30-degrees' orientation.

```

void sendGAData(bool bIsHeartbeat)
{
    wdt_reset();
    String statusMsg = "";
    if (bIsHeartbeat == true) {
        float fDist = getDistanceReading();
        float fAngle = 0;
        statusMsg += "DS:" + String(fDist); //Distance "DS" getDistance
        statusMsg += ";RD:" + String(ObtainRadiations()); //Radiation count/sec "RD"
        statusMsg += ";AG:" + String(fAngle); //Accelerometer value "AG"
        statusMsg += ";";
        statusMsg.toCharArray(ConvStatusMsg, 25); //convert serial data into a char array
        wdt_reset();
        radio.writeFast(&ConvStatusMsg, 25);
        delay(30);
        Serial.println(statusMsg);
        wdt_reset();
        delay(500);
        wdt_reset();
    }
}
}

```

Figure 5.10: Sample of scanner data serialization code.

### 5.2.6 Data Treatment in Control module: Reception, Buffering, Arraying and Plotting

The control module was by default idle when no scan and no command was received. During scanning, the presence of data on the nRF24L01 radio module was continuously checked. If data was available, it was subjected to CRC checks and read into a string before being stored on the EEPROM from where it was re-read for processing.

Since data stored in the EEPROM is in string form, it was read and individual parameters extracted depending on the keyword, such as RD, to imply radiations. After extraction, data was converted into an integer and stored into respective cyclic buffer arrays awaiting final extraction for plotting and display. To display or plot the data on the 3.5" TFT screen, values were read from the arrays using a common pointer to ensure the right data relation, for example, to ensure height corresponds to the respective counts/second before plotting.

Plotting the scan profile on the screen was carried out after obtaining the right data for the arrays and plugging the values into a tailored function. This function translated data into

pixel position and coloured that position. Points were connected as the scan progressed, values read, and pixels coloured, creating a continuous line.

### **5.2.7 System Validation for Reliability and Safety**

Several features have been incorporated in the design and development of this system for the safety of operation. These include the following: In order to guarantee a reliable and robust system, several features were incorporated in the design and development of this system. The operation of a microprocessor was run by the software uploaded on it, which defined the operations between specified states. By programming the system, it ensured user inputs, sensor readings, and defined triggers, are the only means of changing between the various states, which guaranteed seamless operation. This ensured no distraction while in operation.

A watchdog timer was incorporated into microcontrollers to trigger a reset, if not serviced periodically through the execution of a servicing command. This prevented any unexpected hanging in the system due to faults during operations.

To guarantee data integrity during the radio frequency transmission, Cyclic Redundancy Checks (CRC) were done whenever data was received before use (Doherty, 2020). CRC involved subjecting data to a polynomial function to generate a checksum which changed if there was any data change. By carrying out a CRC execution before and after transmission and comparing the two values, errors in data could be detected. In software development, a CRC function was carried by the nRF24L01 modules before transmission and after reception to ensure data integrity (Doherty, 2020).

In this study, an HC-SRO4 ultrasonic sensor was used to measure the vertical distances moved by the source-detector assembly during gamma column scanning for distances between 0 cm and 100 cm. Prior to measurements, the HC-SRO4 ultrasonic sensor was calibrated. Calibration of the sensor involved obtaining actual measurements and comparing them with the respective sensor readings. The data was analysed for correlation using Microsoft Excel.

The MPU-6050 Accelerometer was programmed to limit the extend of circular movement and avoid 360 degrees' rotation of the scanner, which potentially risks twisting and tangling cablings. In this study, rotation was limited to between 0° and 180° reducing the risk of damage.



There exists a great risk of moving the detector- source assembly beyond the preset limits due to operator error and motor errors. To avoid this, limit switches were installed at the topmost, corresponding to 80 cm and bottom-most part (10 cm) of the detector-source rail, and interrupts movement when the switch is enabled.

Prior to measurements, the HC-SRO4 ultrasonic sensor was calibrated. Calibration of the sensor involved obtaining actual measurements using tape measure and comparing them with the respective sensor readings at defined positions along the height of the scanner head. The data was the used to plot a graph of tape measurements against sensor reading at the corresponding heights. The graph was analyzed for correlation using Microsoft Excel, and from line of best fit a correlation equation was obtained (see Fig. 5.11).

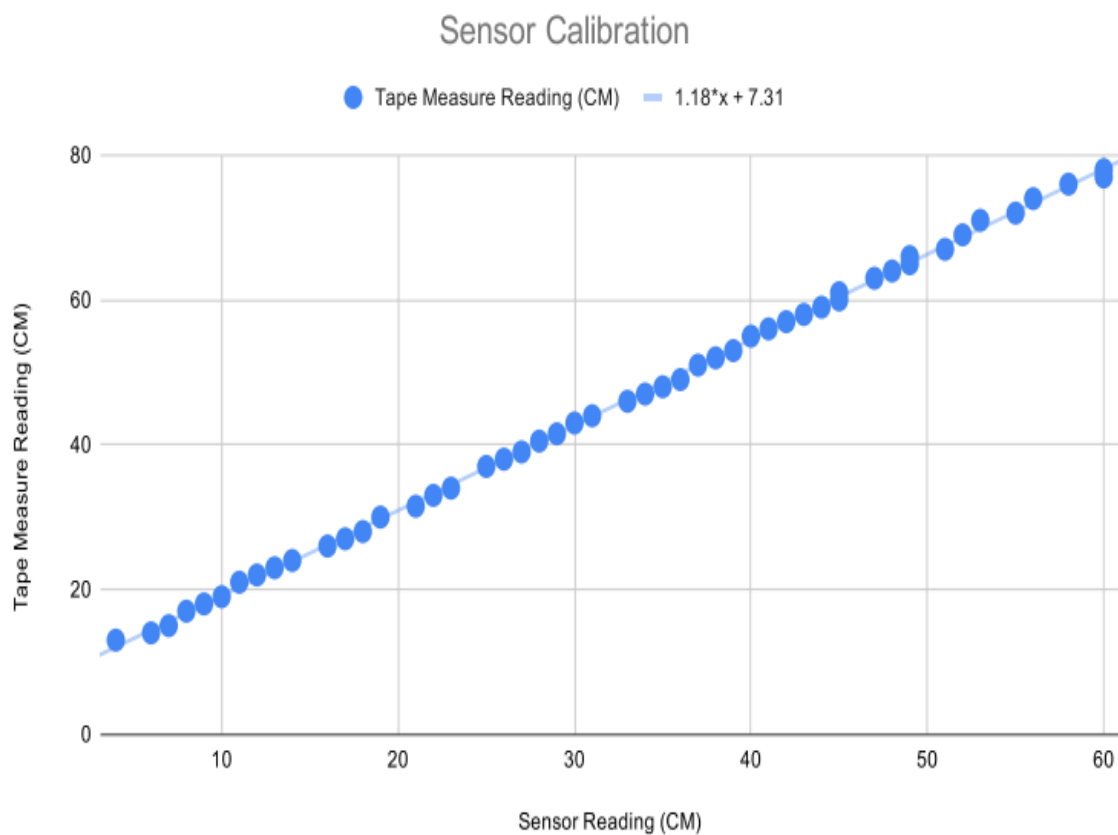


Figure 5.11: HC-SRO4 -Ultrasonic sensor calibration.

The resultant equation that is coded in the scanner software for correcting all values read by the sensor is;

$$\text{Accurate Reading(CM)} = (1.18 * \text{Sensor Reading}) + 7.31 \quad (5.1)$$

This Accurate Reading (CM) value is used in plotting scan profiles.

Percentage error of the sensor = 23%

This obtained equation of data fit was incorporated in the developed code for the scanner software for correction of all sensor readings to read actual values of detector-sources distances.

### **5.3 Test Results**

#### **5.3.1 Gamma column scanning setup**

In this study, the model column was an 8” diameter PVC pipe fitted with two 5 cm wide, 6 mm mild steel rings at 10 cm intervals; the bottom tray is positioned at 23cm mark while the upper tray is positioned at 50cm mark. An 8cm wide wooden piece was placed at 37cm mark to represent a simulated malfunction within the distillation column. The set up shown in Fig.5.12 represents distillation column and separation trays in a typical industrial process and was tested for faults using the prototype. The handheld gadget shown in Fig.5.13 was used as data logger and to control the scanner head movement.

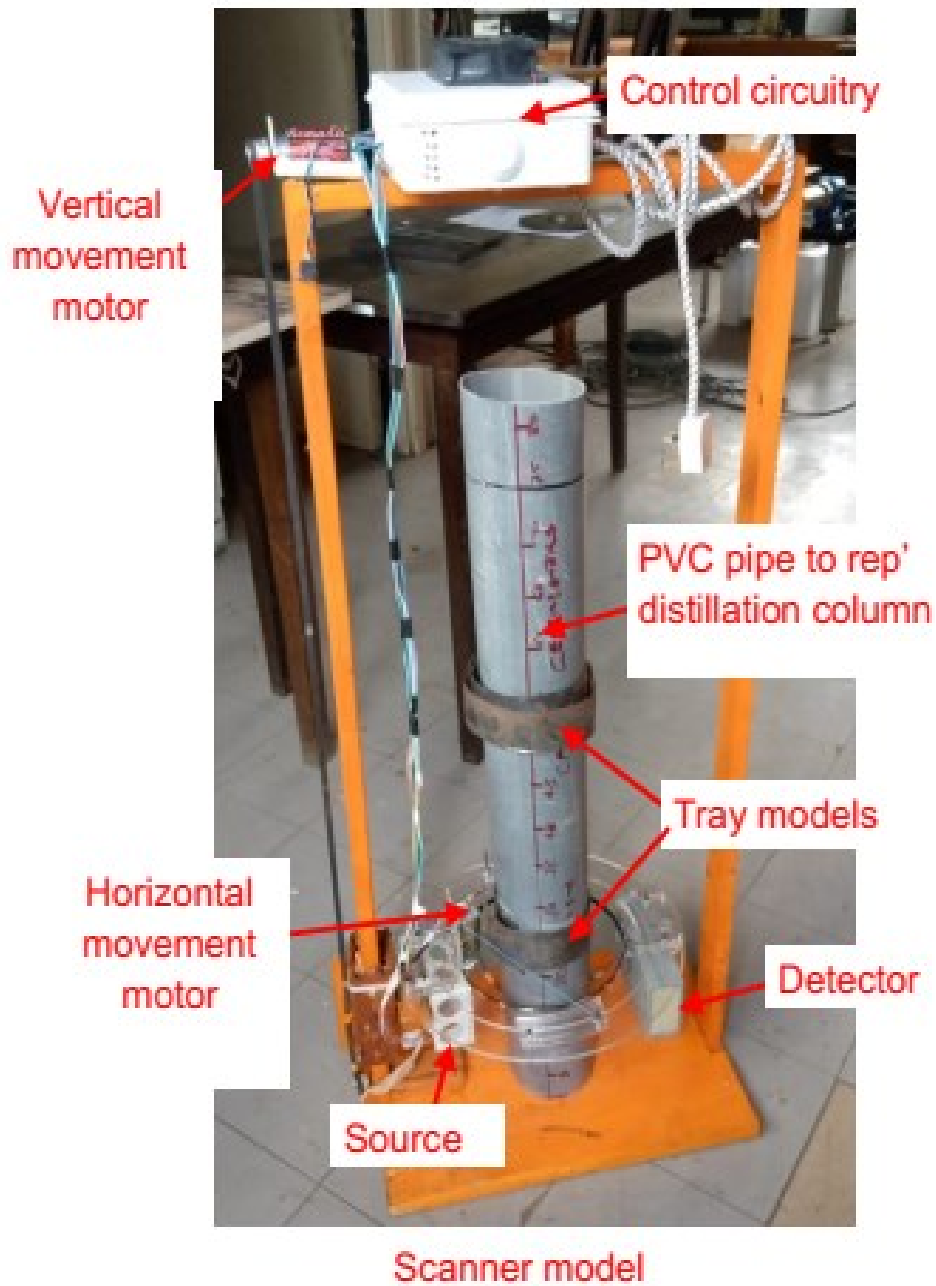


Figure 5.12: Scanner head system outlook with the model column.

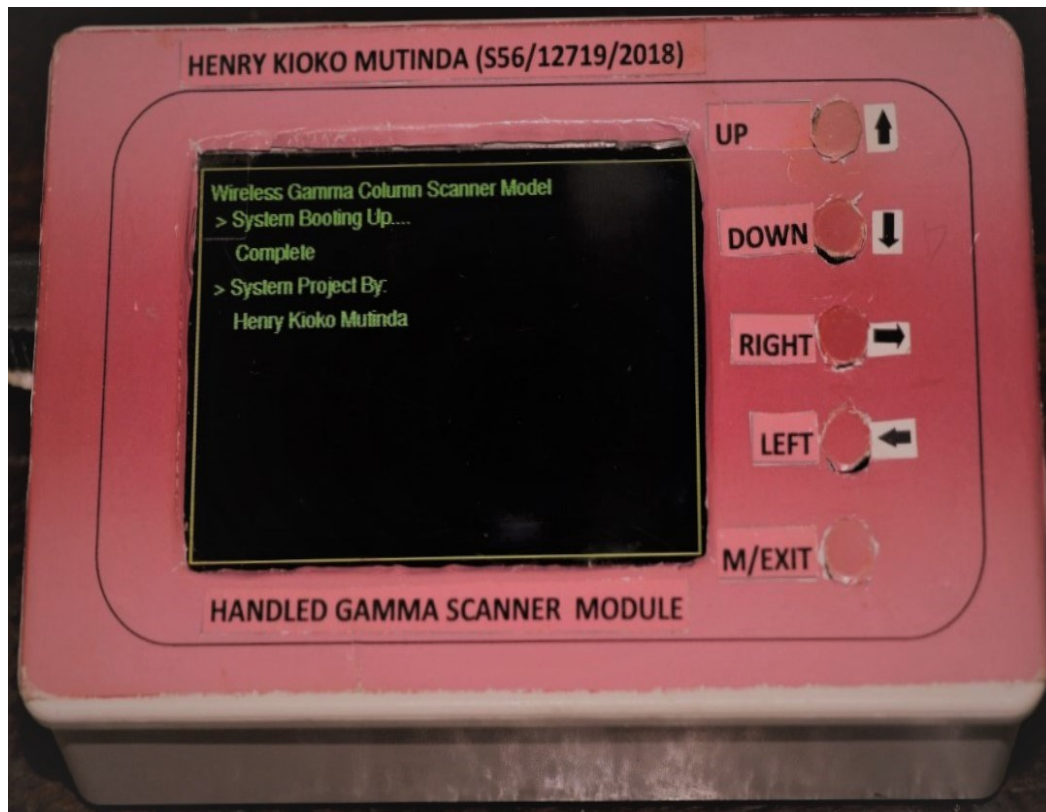


Figure 5.13: The developed wireless handheld controller module.

### 5.3.1.1 Gamma Scanning System Configuration and Laboratory Measurements

After boot up, the system was configured for measurements via the wireless controller and prompted user to select either of the operation modes available for radiotracer or gamma scanning measurements (see Fig. 5.14). In this case, Gamma operation mode was selected. Under Gamma mode, the system prompted you to select either to run in automatic Gamma scanning mode or manual. Under automatic mode, the scanning parameters which included desired pause time and travel distance were entered as shown in Fig.5.15, while under manual mode, it was possible to move the scanner up or down remotely during the scan process depending on the preset conditions.



Figure 5.14: Menu options display for mode on the wireless controller.

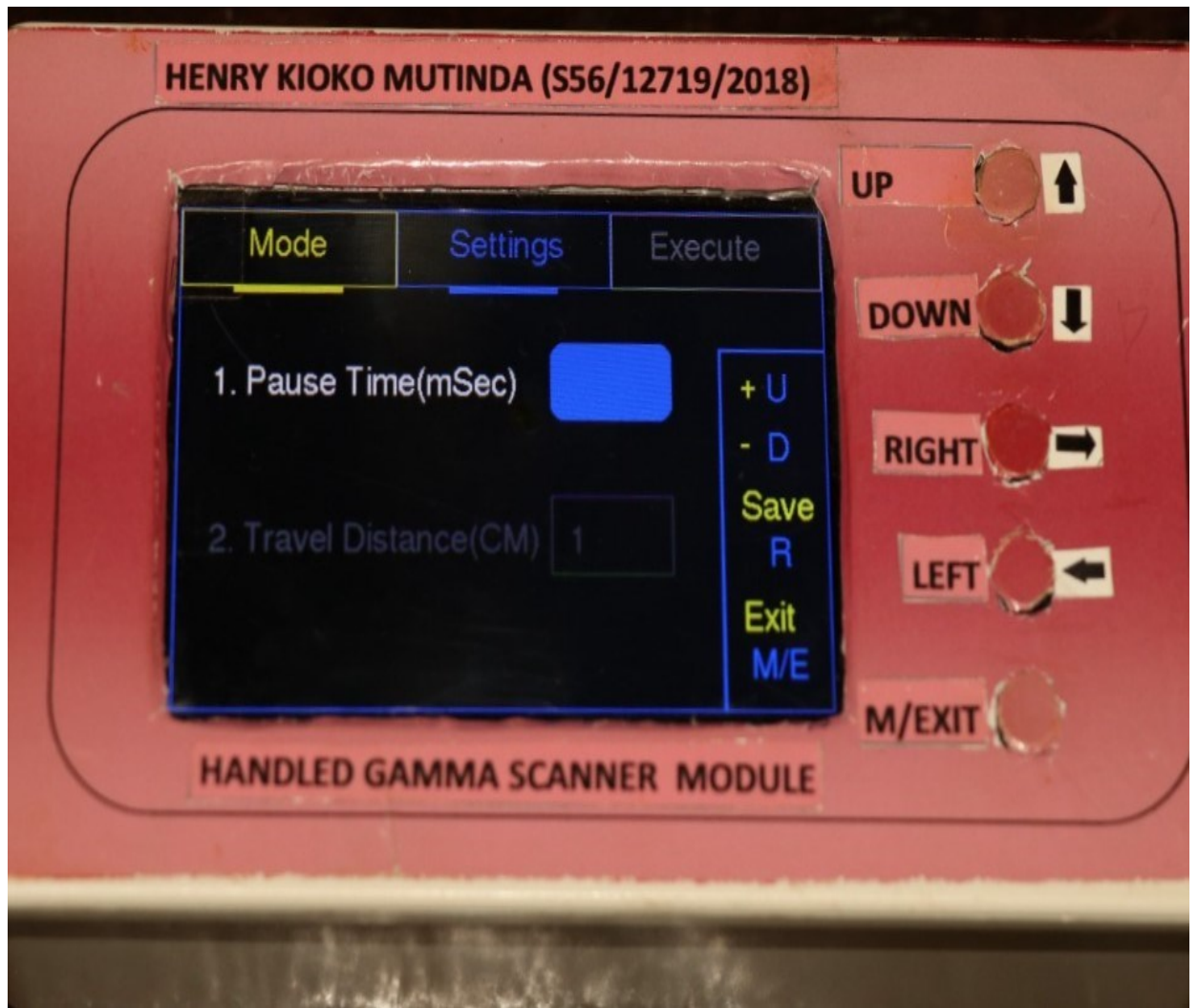


Figure 5.15: Menu options display for parameter selection on the wireless controller.

### 5.3.1.2 Results of Gamma Column Scan measurements

After running the test to scan the model column, a scan profile was generated in real time and the scan results stored as a file copy in the handheld module. The data of the archived file copy of results of gamma column scan data in Microsoft Excel file format obtained during Gamma column scanning exercise using the developed system was used to plot a profile of radiation intensity received against the height of model column using MATLAB. A screenshot of the scan profile drawn on the TFT miniature screen in real time was taken and compared with the scan profile generated with MATLAB as shown in Fig. 5.16.

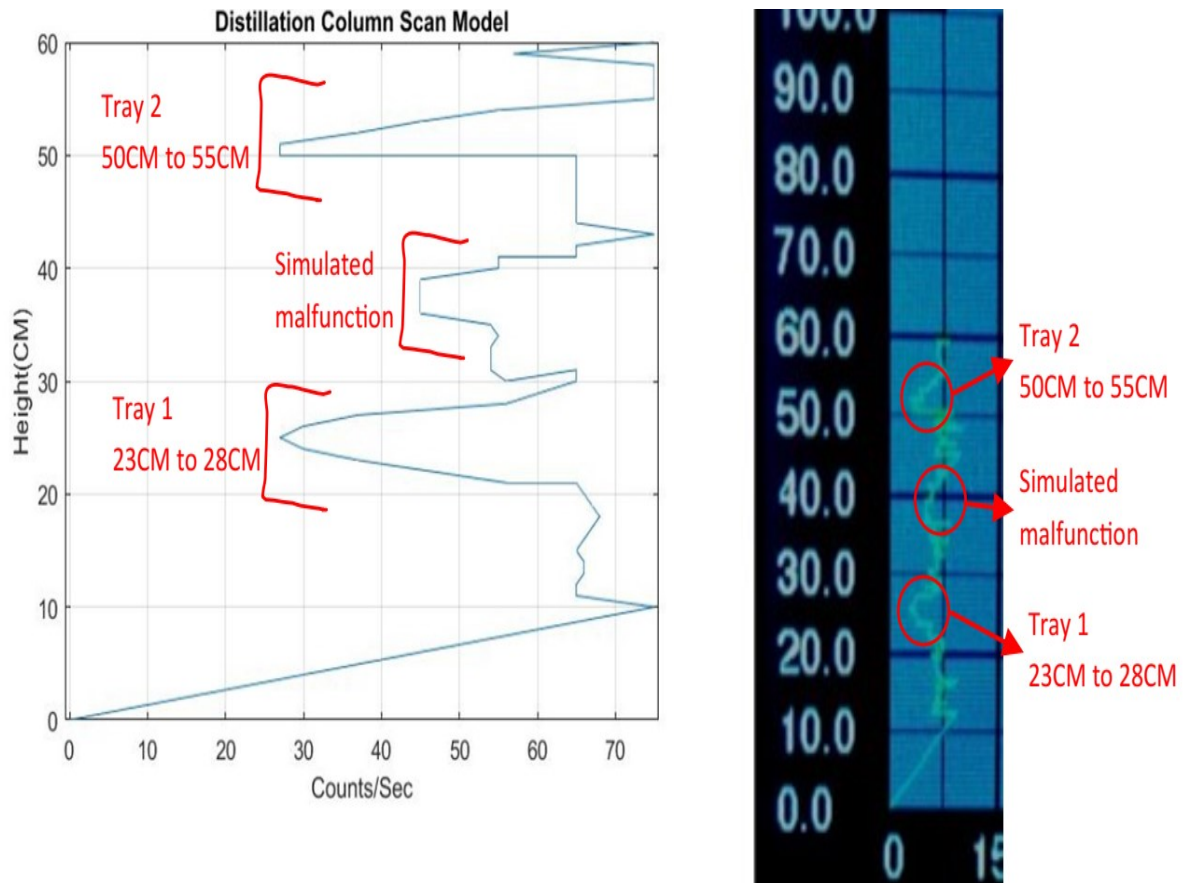


Figure 5.16: MATLAB vs LCD column profile plots.

### 5.3.1.3 Observation and Discussion

From both scan profiles generated, the two 5 cm wide, 6 mm mild steel rings representing the separation trays were identified. The first tray was located between 23 cm and 28 cm, and the second tray between 50 cm and 55 cm along the column model. It was also possible to identify the location of the 8 cm piece of wood representing the simulated column malfunction at a height of 37 cm. The location of these trays and malfunction was identified at the regions of low radiation count rates which represented areas of high density. The results were then validated through comparison with actual physical measurements of the column model. Repeated measurements at the different control module distances from the scanner showed the same results. These repetitive similar results at varying distances verified high accuracy, reliability and consistency of the system.

In general, high accuracy of the scan profile has been achieved. The following were observed of the system; system proved to be reliable for use in gamma column scanning, wireless communication has been effective through the use of radiofrequency,

Automation was possible through remote-controlled motorized movement, data was stored automatically for future retrieval and reference, data analysis in real-time and dual-axis movement were possible.

### **5.3.2 Radiotracer Configuration for Laboratory Measurements**

When powered on, the control requires approximately 30s to boot. Two options are thereafter available for selection; radiotracer and gamma operation modes. Upon selecting the radiotracer option on the control module, the system was automatically configured for radiotracer measurements mode. It displays an X-Y axis window on the LCD screen, where a profile of counts vs time begins to plot. For this study, the assumption is that the detector is fixed. In this test, a radiation emitting source was instantaneously introduced to and withdrawn from the detector three times at different intervals of time. A stop watch was used to monitor the time intervals at which the introduction was done.

After the instantaneous introduction and withdrawal of the radioactive source at different time intervals, a screenshot of the profile drawn in real-time display on the LCD screen was taken as shown in Fig.5.17 (a) and the data of the archived file copy of results, in Microsoft Excel file format, which was stored in an SD card during the study, used to plot a profile using MATLAB as shown in Fig.5.17 (b).



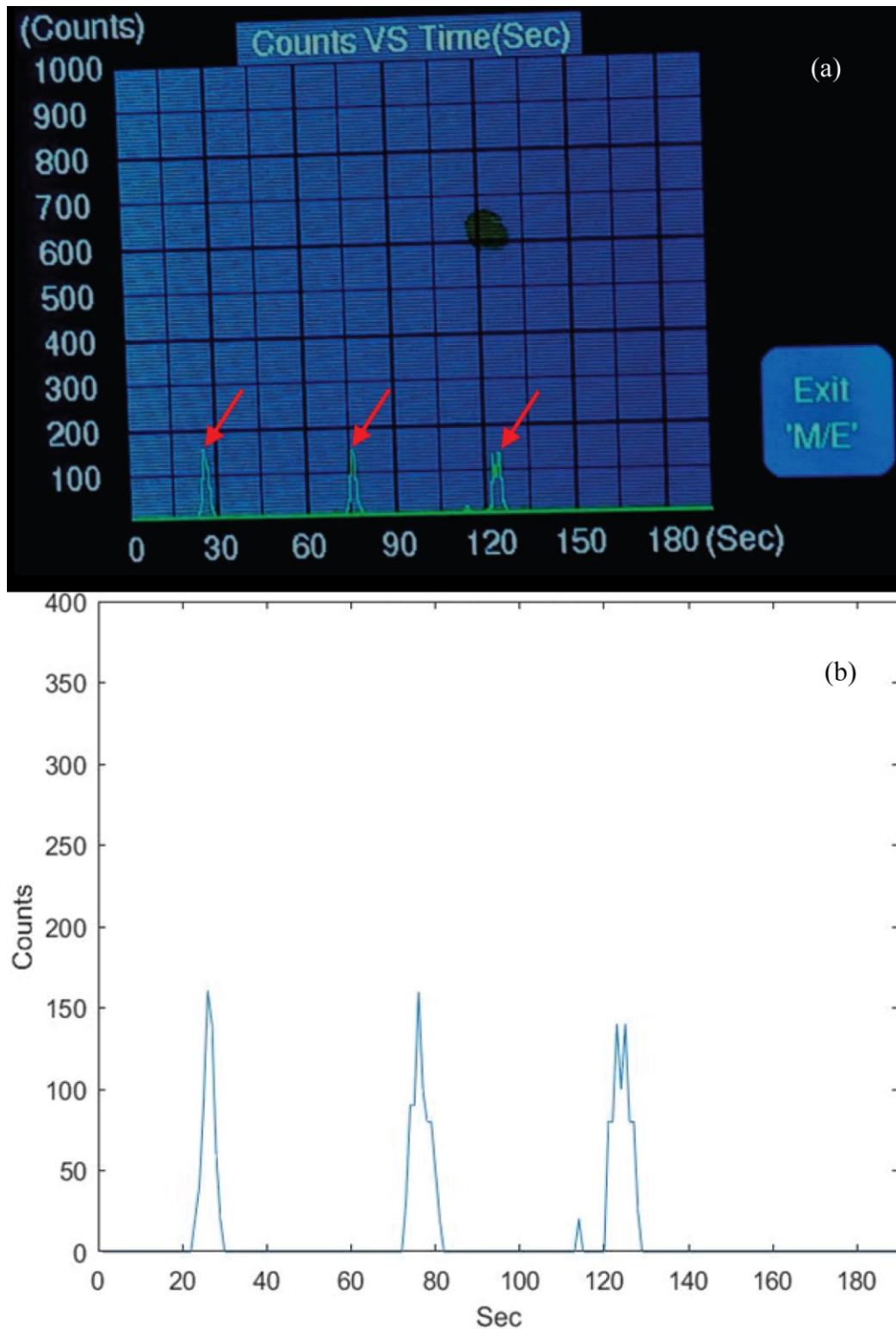


Figure 5.17: (a) Radiotracer scan simulation results on display and (b) Scan profile of radiotracer data using MATLAB.

From observation and comparison of the two profiles generated, three spikes corresponding to the three introduction and withdrawal of the radioactive source are observed to happen at 23sec, 73sec and 120sec marks consecutively in the horizontal axis.

These precisely matches the time intervals at which the introduction was done as recorded on the stop watch. This has demonstrated the possibility of using three or more detectors located at various positions in a radiotracer scanning measurement with this module and obtain reliable results.

In general, gamma column scanning technique has significantly contributed to effective on-line diagnosis of distillation columns in many industrial processes. In many scanning systems in use, the data is transmitted from the detector to the detection circuitry by use of coaxial cable. This data is transmitted as an analog signal. Being a guided medium, data transmission by coaxial cable are affected by noise, length of cables and environmental conditions which wear off the cables. This consequently leads to distortion of information. Wireless data transmission is necessary in modern measuring systems due to decentralization and the need for remote monitoring. This has led to the development of telemetry systems capable of transmitting information in two directions, enabling cross-communication between systems over a long distance while maintaining high degree of data integrity.

Data integrity in this project has been achieved by converting analog detector output into digital signal before transmission. This data in turn is subjected to Fourier transformations to ensure no variance before and after transmission. Use of radio frequency wireless transmission has seen elimination of interference which occurs in guided data transmission. This helped maintain data accuracy over distances beyond 100m. The detector -source movement for these existing systems has been manually done or mechanically driven. This calls for increased manpower and consequently increasing the cost of the exercise with increased chances of human error. With modern electro-mechanical systems, its possible to have an automated drive system with great torque and high precision capabilities. To reduce the manpower and its costs due, this project embraced nanotechnology through replacement of non-electronics with microsystems which has provisions for control from programmed micro-controller. This automated Electro-mechanical movement drive system facilitated detector-source movement in horizontal and vertical orientations with increased positioning accuracy.

The components used in the existing system are big in size with low performance capabilities. Modern technology, has seen the production of microelectronics with robust performance capabilities. In this project, the circuitry systems were developed using

components produced in the nanotechnology world. This resulted into substitution of the data logger and laptop with a highly portable handheld gadget, which has the capability to perform all the operations of the data logger and laptop combined.

In conclusion, this system has realized an improved performance, precision, data transmission and processing and movement automation. This has resulted to reduced cost of running, manpower and time, and highly portable system with increased accuracy over the existing system.

## CHAPTER SIX

### CONCLUSIONS AND RECOMMENDATIONS

#### 6.1 Conclusions

At a total cost of approximately KSHs. 70,000, the following was achieved; a gamma scanning head system was constructed with capabilities for dual-axis movement, vertical and horizontal, using motors and timing belts. Movement in both axes was controlled wirelessly via the handheld control module. In general, there was an instantaneous response in the system following the issuance of commands.

Using the selected nRF24L01 and 433MHz radio frequency communication modules, there was instant data transmission between the handheld controller and the scanner. The choice and use of a pair of radio frequency communication modules for one-way communication increase the system reliability as data can be transmitted in a full-duplex mode without delays during the transition in the case of one pair.

The use of wireless communication in gamma scanning operation increased flexibility as the wireless handheld controller works on a 4.5V power bank or battery. Since the controller has a 3.5" TFT LCD, which displays configurations, scan profiles and parameters, data treatment and representation is done using one microcontroller, which is cheap and robust. Data Cyclic Redundancy (CRC) checks and storage were added to increase reliability as errors in data can be detected and discarded. The movement of the scanner was executed from the wireless controller by pressing the respective travel direction buttons. This eliminates the need for more than one person carrying out the scan and uses computer software for plotting scan profiles.

#### 6.2 Recommendations

This project was limited to testing the possibility of integrating microcontroller-based wireless data transmission, motorized scanner movement control and plotting of gamma scan data. Development can be carried out to have cloud-based data storage for retrieval using a GSM enabled module.

Since the system has proved to be reliable with one detector, a more advanced system with provisions for more detectors can be developed and also incorporate a communication network that is more reliable in transmissions beyond a radius of 100m.

Since the interpretation of scan profile was left to human interpretation, artificial intelligence and machine learning algorithms can be added to the controller software. This will enable a machine-based decision on scan results in real-time as the scan is carried out, eliminating human errors and delays brought in by profile analysis after scanning.

## REFERENCES

- Adegbija, T., and Tandon, R. (2017, July). Coding for Efficient Caching in Multicore Embedded Systems. In *2017 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)* (pp. 296-301). IEEE.
- Adwet, W. M., Pant, H. J., Mangala, M. J., and Masinza, S. A. (2019). Evaluation of hydraulic performance of an anaerobic pond using radiotracer technique. *Applied Radiation and Isotopes*, *145*, 101-105.
- Ahmed.F., Alim, S., Islam, S., Bhusan.K, and Shafiul, I., (2016) “433 MHz (Wireless RF) communication between Two Arduino UNO”, *American Journal of Engineering Research (AJER)*, Vol-5, Issue-10, pp-358-362.
- Alami R., Bensitel A., Benahmed A., Saadaoui A., “Contrôle Par Gamma-Ray Scanning de la Colonne de Distillation Sous Vide 31-C-101 de la SAMIR –Mohammedia, Morocco,” Technical Report, CNESTEN-DIAIRI, March 2012.
- Alami, Rachad and Bensitel, Abdeslam. (2012). Radioisotope Technology as Applied to Petrochemical Industry. 10.5772/38387.
- Anger, H. O. (1968). Multiplane tomographic gamma-ray scanner.
- Åström, K. J., and Hägglund, T. (2000). Benchmark systems for PID control. *IFAC Digital Control: Past, Present and Future of PID Control*, 5-7.
- Austerlitz, H. (2003). *Data Acquisition Techniques Using PCs*. Academic Press, NewYork.
- Bachuwar, V. D., Shaligram, A. D., and Deshmukh, L. P. (2017). Low Cost Wireless Data Acquisition System for Multisensor Applications. *International Journal of Development Research*, *7(08)*, 14346-14349.
- Baljit Kaur, Madhvi Verma, Satbir Singh (2015) *Overview of Bluetooth Technology and its Communication Applications*, INPRESCO, India.
- Bao, XJ and Wei, WS and Liu, YS and She, G. and Shen, F. (2002). Troubleshooting distillation column by gamma ray scanning technique. *Chinese Journal of Chemical Engineering*. 10. 52-55.

- Barati, H., Khorsandi, M., and Fegghi, S. A. H. (2019). A new approach for profile interpretation of a tray-type distillation column using gamma-ray scan and template matching techniques. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 916, 183-189.
- Barr, M., and Massa, A. (2006). *Programming embedded systems: with C and GNU development tools*. " O'Reilly Media, Inc."
- Bentley, J. (2008). *Principles of measurement systems*. Harlow: Pearson Prentice Hall.
- Bernhard, D., and Gurevych, I. (2008, June). Answering learners' questions by retrieving question paraphrases from social Q and A sites. In *Proceedings of the third workshop on innovative use of NLP for building educational applications* (pp. 44- 52).
- Bhagwat, M. R., Chandrakant, M. R., Nitin, M. Y., and Adhau, R. (2017). Using real time communication provide emergency treatment and hospital searching via IOT Based system. *International Research Journal of Engineering and Technology (IRJET)*, 4(10), 1471-1473.
- Billingsley, J.(2006). *Essentials of mechatronics* John Wiley and Sons, Inc., Hoboken, New Jersey. Canada
- Bora A., Sarma K.C., "Design of a USB based Multichannel, Low Cost Data Acquisition System using PIC Microcontroller," *International Journal of Computer Applications* (0975 – 8887) Volume 59– No.6, December 2012.
- Borchers, J.(2013) , *Arduino in a Nutshell*.
- Bouck, D. (2018). Inspecting Distillation Towers Part 1: Turnarounds. *Chemical Engineering Progress*, 114(9), 26-35.
- Brooks, R. A., and Di Chiro, G. (1976). Principles of computer assisted tomography (CAT) in radiographic and radioisotopic imaging. *Physics in Medicine & Biology*, 21(5), 689.
- Brown, G. G. (1950). *Unit operations*. New York: Wiley

- Carney, D., Ghosh, S., Krnjaic, G., and Taylor, J. M. (2019). Gravitational Direct Detection of Dark Matter. *arXiv preprint arXiv:1903.00492*.
- Cheremisinoff, N. P. (2000). *Handbook of chemical processing equipment*. Elsevier.
- Chmielewski, A. G., Smoliński, T., and Rogowski, M. (2019). Radiotracers and Nucleonic Control Systems Applied in Industry---Polish Case. *World Journal of Nuclear Science and Technology*, 9, 27-66.
- Chuong, H. D., Hung, N. Q., Le, N. T. M., Nguyen, V. H., and Thanh, T. T. (2019). Validation of gamma scanning method for optimizing NaI (Tl) detector model in Monte Carlo simulation. *Applied Radiation and Isotopes*, 149, 1-8.
- Cool, D. A. (2018). The new mandate and work of ICRP Committee 4. *Annals of the ICRP*, 47(3-4), 214-220.
- De Koeijer, G.M. and Kjelstrup, S. (2000) Minimizing Entropy Production Rate in Binary Tray Distillation. *Int. J. Appl. Thermodyn.* 3:105-110
- De Koeijer, G.M. and Kjelstrup, S. (2000) Minimizing Entropy Production Rate in Binary Tray Distillation. *Int. J. Appl. Thermodyn.* 3:105-110.
- De Swart, J., and Groenevelt, P. (1971). Column Scanning with 60 Kev Gamma Radiation. *Soil Science*, 112(6), 419-424. doi: 10.1097/00010694-197112000-00006
- Diehl R. (2001) Gamma- Production and Absorption Processes. In: Schönfelder V. (eds) *The Universe in Gamma Rays*. Astronomy and Astrophysics Library. Springer, Berlin, Heidelberg. [https://doi.org/10.1007/978-3-662-04593-0\\_2Ray](https://doi.org/10.1007/978-3-662-04593-0_2Ray)
- Do, R. K., and Dick, D. W. (2018). PET/MRI: Safety Considerations. In *PET/MRI in Oncology* (pp. 115-130). Springer, Cham.
- Doherty, B. (2020). *nRF24L01 Library Documentation* [Ebook] (1st ed.). Retrieved from <https://circuitpython-nrf24l01.readthedocs.io/en/latest/>
- Drummer, D., Ehrenstein, G. W., Hopmann, C., Vetter, K., Meister, S., Fischer, T., ... and Prokop, J. (2012). Innovative process technologies for manufacturing



- thermoplastic micro parts—analysis and comparative assessment. *Journal of Plastics Technology*, 8(5), 439.
- ECEE (2020). Using the MPU6050. Inertia Measurement Systems Gyroscopes & Accelerometers Sensor fusion I2C MPU-6050
- El Korchi, K., Alami, R., Bensitel, A., Outayed, R., Echchelh, A., and Chaouch, A. Comparative Study of Radiotracer and Sealed Source Techniques for Detecting the Coking in Distillation Columns' Packings.
- Electric Electronic Technology-*Step and Servo Motors*, SVET, 2007. Retrieved from <http://www.ieeexplore.ieee.org/document/1570123/>
- Gitau, J., Gatari, M. J., and Pant, H. J. (2019). Investigation of flow dynamics of porous clinkers in a ball mill using technitium-99m as a radiotracer. *Applied Radiation and Isotopes*, 154, 108902.
- Gyorki, R. (2004). *Signal Conditioning and PC-Based Data Acquisition Handbook*. Iotech.
- Hammad, M. and Kasban, Hany & Elaraby, Sayed and Dessouky, M.I. & Zahran, O. & Abd El-Samie, Fathi. (2014). Distillation Column Malfunctions Identification using Higher Order Statistics. *International Journal of Computer Applications*. 108. 7-13. 10.5120/18873-0129.
- Haraguchi, M. I., Calvo, W. A. P., and Kim, H. Y. (2018). Tomographic 2-D gamma scanning for industrial process troubleshooting. *Flow Measurement and Instrumentation*, 62, 235-245.
- Hartig, J. U., Bieberle, A., Engmann, C., and Haller, P. (2019). Investigation of density variations in molded wood tubes using gamma-ray CT and correlation with load-bearing behavior. *CompWood*, 66-66.
- Jaafar, A. (2005). Gamma ray scanning for troubleshooting, optimization and predictive maintenance of distillation columns. *Hydrocarbon Asia*, Jan/Feb, 62–65.
- Charlton, J. S. (1985). *Radioisotope techniques for problem-solving in industrial process plants*. Glasgow: Leonard Hill.

- James, K. (1997). *The PC Programmer's Data Acquisition Handbook*. International Thomson Computer Press.
- Jaouen, G., Top, S., Vessières, A., and Alberto, R. (2000). New paradigms for synthetic pathways inspired by bioorganometallic chemistry. *Journal of Organometallic Chemistry*, 600(1-2), 23-36.
- Ketheeswaren, Sivapackiyathan and Rosilinmary, S and Visvanath, B. (2009). Models of transmission Media for a Library Network: A Comparative Study.
- Khiter, A., and Khechiba, K. (2016). Design and implementation of microcontroller based controller for direction and speed of a robot.
- Kim, J., Jung, S., and Kim, J. (2007). The identification of flooding phenomenon in a column of refinery process by gamma absorption technique. *J. Label. Compd. Radiopharm.*, 50(5/6), 605–606. doi: 10.1002/jlcr.1299.
- Kim, J., Jung, S., Moon, J., and Cho, G. (2011). Industrial gamma-ray tomographic scan method for large scale industrial plants. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 640(1), 139-150.
- Kister, H., (1992), *Distillation Design*, McGraw-Hill, New York
- Klar, V. (2017), *Introduction to Raspberry Pi* Aalto University. Finland.
- Laraki K., et al., “Diagnosis of Distillation Column Problems Using New Generation Gamma-Ray Scanning Gauge,” *The International Arab Journal of Information Technology*, Vol. 5, No. 1, January 2008.
- Laraki, Khalid and Alami, Rosma and Morsli, Rajaa and Bensitel, Abdessalam and EL Badri, Lhachemi and Hamzaoui, El-Mehdi. (2008). Diagnosis of Distillation Column Problems Using New Generation Gamma-Ray Scanning Gauge. *Int. Arab J. Inf. Technol.* 5. 75-79.
- Lee, Z., Ophus, C., Fischer, L. M., Nelson-Fitzpatrick, N., Westra, K. L., Evoy, S., and Mitlin, D. (2006). Metallic NEMS components fabricated from nanocomposite Al– Mo films. *Nanotechnology*, 17(12), 3063.

- Ma, C. C., Ho, H. W., Chou, W. H., Chen, D. R., and Chang, C. Y. (2020). In-situ measurements of strain distribution by coupling digital image correlation and an optical microscope. *Microsystem Technologies*, 1-7.
- Madhvi, V., Satbir, S., and Baljit, K. (2015). An Overview of Bluetooth Technology and its Communication Applications. *International Journal of Current Engineering And Technology*, 5(3), 1-7.
- Mahbub, Mobasshir. (2019). Design and Implementation of Multipurpose Radio Controller Unit Using nRF24L01 Wireless Transceiver Module and Arduino as MCU. *International Journal of Digital Information and Wireless Communications*. 9. 61-72. 10.17781/P002598.
- Manshoori Yeganeh, A., Movahhedy, M. R., and Khodaygan, S. (2019). An efficient scanning algorithm for improving accuracy based on minimising part warping in selected laser sintering process. *Virtual and Physical Prototyping*, 14(1), 59-78.
- Marin-Palomo, P., Kemal, J. N., Karpov, M., Kordts, A., Pfeifle, J., Pfeiffer, M. H., ... and Koos, C. (2017). Microresonator-based solitons for massively parallel coherent optical communications. *Nature*, 546(7657), 274-279.
- Nerokas Engineering Ltd, Thika Kenya.
- Ngo, T. D. (2019). Open-Source Electronics Platforms: Development and Applications.
- Ngui, D., Chege, J., and Kimuyu, P. (2016). Kenya's industrial development. *Manufacturing Transformation*, 72.
- Nguyen, T. C., Van Nguyen, Q., and Lakosi, L. (2019). Determination of nuclear material content of items originating from damaged spent fuel assemblies by using collimated gamma-spectrometric scanning. *Progress in Nuclear Energy*, 110, 103-109.
- Nureni, Yekini. (2015). DATA COMMUNICATION & NETWORKING, Lagos, Nigeria.
- Olujic, Z., Fakhri, F., Rijke, A. de., Graauw J. de. And Jansens, P. J. (2003) Internal Heat Integration – The Key to an Energy Conserving Distillation Column. *Journal of Chemical Technology and Biotechnology*, 78:241–248.

- Osaka, K., Yokozawa, Y., Torizuka, Y., Yamada, Y., Manota, M., Harada, N., ... and Sato, M. (2019, January). Versatile high-throughput diffractometer for industrial use at BL19B2 in SPring-8. In *AIP Conference Proceedings* (Vol. 2054, No. 1, p. 050008). AIP Publishing.
- Pant, H. J., Kundu, A., and Nigam, K. D. P. (2001). Radiotracer applications in chemical process industry. *Reviews in chemical engineering*, 17(3), 165-252.
- Patel, V. P., Walker, L. A., and Feinstein, A. (2017). Deconstructing the symbol digit modalities test in multiple sclerosis: The role of memory. *Multiple sclerosis and related disorders*, 17, 184-189.
- Patil, Nilesh and Patil, Vilas. (2016). Operational and economic assessment of distillation column from the performance of tray. *Int. J. Eng. Trend. Technol.* 4. 500-505.
- Pilling, Mark and Holden, Bruce. (2009). Choosing Trays and Packings for Distillation. *Chemical Engineering Progress*. 105. 44-50.
- Putra, I. P. E. S., Brusey, J., Gaura, E., and Vesilo, R. (2018). An event-triggered machine learning approach for accelerometer-based fall detection. *Sensors*, 18(1), 20.
- Reilly, D., Ensslin, N., Smith, H., and Kreiner, S. (1991). *Passive nondestructive assay of nuclear materials* (pp. 43 - 62). Springfield, VA: US Department of Commerce, National Technical Information Service.
- Richardson, J.F., Harker, J.H, Backhurst, J. R., *Coulson and Richardson's CHEMICAL ENGINEERING Volume 2*. 2002: p. 559-575.
- Saengchantr, D., Srisatit, S., and Chankow, N. (2019). Development of Data Acquisition Technique for Inspection of Pipes in Vessel using Gamma Scanning Coupled with Computed Tomography. *Engineering Journal*, 23(2), 85-95.
- Sanches, M. P., Hanaguchi, M. I., Beckmann, F. D. S., & Calvo, W. A. P. (2007, July). Radiological safety in the gamma scan procedures.
- Senani, R., Bhaskar, D. R., and Singh, A. K. (2014). *Current conveyors: variants, applications and hardware implementations*. Springer.

- Shahabinejad, H., Fegghi, S. A. H., and Khorsandi, M. (2014). Structural inspection and troubleshooting analysis of a lab-scale distillation column using gamma scanning technique in comparison with Monte Carlo simulations. *Measurement*, 55, 375-381.
- Spero, Y. E. (2018). *U.S. Patent No. 9,955,551*. Washington, DC: U.S. Patent and Trademark Office.
- Stoddart, H. F., and Stoddart, H. A. (1979). A new development in single gamma transaxial tomography Union Carbide focused collimator scanner. *IEEE Transactions on Nuclear Science*, 26(2), 2710-2712.
- Stokic, D., Kirchhoff, U., and Sundmaeker, H. (2006, May). Ambient intelligence in manufacturing industry: control system point of view. In *The eighth IASTED international conference on control and applications* (pp. 24-26).
- Tan, L. (2008), *Digital Signal Processing, Fundamentals and Applications*. Elsevier Academic Press, Georgia.
- Theraja, B.L and Theraja, A.K (2005). *A Textbook of Electrical*(24<sup>th</sup> Ed). India: S. Chand and Company Ltd.
- Tinti, F., Kasmaee, S., Elkarmoty, M., Bonduà, S., and Bortolotti, V. (2018). Suitability evaluation of specific shallow geothermal technologies using a GIS-based multi criteria decision analysis implementing the analytic hierarchic process. *Energies*, 11(2), 457.
- Walinjkar, P., Kumar, U., Singh, B. P., and Singh, G. (2009). Wireless acquisition of transmitted gamma radiation for condition monitoring of distillation columns. In *Proceedings of DAE-BRNS symposium on nuclear and radiochemistry*.
- Warren, J.D, Adams, j. (2011), *Arduino Robotics*. Apress Publishers, CA. USA.
- Zhirnov, V. V., and Cavin, R. K. (2013). Future microsystems for information processing: limits and lessons from the living systems. *IEEE Journal of the Electron Devices Society*, 1(2), 29-47.

## APPENDICES

### Appendix I: Components Specifications

Table A1.1: Arduino Mega Specifications

Feature	Details
Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (14 PWM)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB (8 KB used by bootloader)
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

Table A1.2: 12V DC motor Specifications (Source: Warren, 2011)

Parameter	Value
Power rating	1.31W
Operating Voltage	12VDC
Rated Current (No Load)	300mA
Mechanical Output (Torque)	2000 g.cm
Nominal Tolerance Torque	2.0 kgf-cm
Momentary (Max) Tolerance Output	3.5 kgf-cm
Speed	100rpm
Efficiency	65%

Table A1.3: nRFL2401 Specifications (Source: Brendan, 2020)

Parameter	Value
RF transceiver Module	2.4GHz
Operating Voltage	3.3V
Nominal current	50Ma
Transmission range	200 feet
Operating current	250mA
Communication Protocol	SPI
Baud Rate	250Kbps – 2Mbps
Channel Range	125
Maximum Pipelines/node	6

Table A1.4: 433 MHz RF Transmitter Module Specifications (Source: Ahmed *et al.*, 2006)

Parameter	Value
Modulation Mode	ASK
Working Voltage	3V to 12V
Working Current	9mA to 40mA
Operating Frequency	315MHz to 433MHz
Transmission power	25mW
Resonance	SAW
Data transmission rate	10Kbps
Transmission range	90Meters in open areas

Table A1.4: 433MHz RF receiver module specifications (Ahmed *et al.*,2006)

Parameter	Value
Modulation Mode	ASK
Working Voltage	5.0V
Working Current	5.5mA
Operating Frequency	315MHz to 433MHz
Bandwidth	2MHz
Sensitivity	-100dBm at 50Ω
Data transmission rate	10Kbps

Table A1.5: Power supply rating calculation

<b>Component</b>	<b>No</b>	<b>Current Rating (A)</b>	<b>Cumulative Current(A)</b>
12V DC Geared Motor	2	0.3	0.6
433MHz RX Radio Module	1	0.0055	0.0055
nRFL2401L Radio Module	1	0.05	0.05
LEDs	6	0.01	0.06
MPU-6050 Accelerometer	1	0.01	0.01
HC-SR04 Ultrasonic Sensor	1	0.02	0.02
<b>Total Current</b>			<b>0.7455</b>



## Appendix II: System Booting and Distance Reading

```
void loop() {
  //1. check all sensors
  serviceSensors();
  //2. check if any new commands and process them
  serviceCommands();
  //3. check for any state follow on state transitions
  runStateTransitions();
  //4. Reset the internal watchdog
  wdt_reset();}

Distance reading
// -----
// Calculate a ping median using the ping_timer() method.
const static unsigned int DistanceSensorAveragingPeriodSeconds = 1;
const static unsigned int paramDistanceISRFrequencyHz = 16;//5
const static unsigned int DistanceSensorSampleSize =
(paramDistanceISRFrequencyHz*DistanceSensorAveragingPeriodSeconds);
static unsigned int DistanceSamples[DistanceSensorSampleSize] = {0};
//unsigned long ulCurrentDistanceAverage = 0;
long duration;
float distance;float fDistance = 0.0;
void setupDistanceSensor() {
  //define and initialise inputs
  if (bDistanceEmulation == true) {}
  else { pinMode(TRIGGER_PIN, OUTPUT);
    pinMode(ECHO_PIN, INPUT);
  }}void sampleDistanceSensor() {
  static int sampleIndex = 0;
  //if index reaches array size set to zero again
  if ( sampleIndex == DistanceSensorSampleSize ) {
    // getDistanceReading(); //calculate avg distance when array is full.
    sampleIndex = 0; }
  DistanceSamples[sampleIndex] = getDistancesampleIndex ++;
  // Serial.println(sampleIndex);
}
//call from anywhere to get latest smoothed value
float getDistanceReading() {
  //wdt_reset();
  float currentDistanceReading = 0.0;
  float EmulatedCurrentDistanceReading = 80.0;
```

```

float currentDistanceAverage = 0.0;
int counterff;
unsigned long ulCurrentDistanceAverage = 0;

if (bDistanceEmulation == true) {
    EmulatedCurrentDistanceReading = 80;
}
else {
    //compute the current average ADC level
    for (int i = 0; i < DistanceSensorSampleSize; i++) {
        // wdt_reset();
        currentDistanceReading += DistanceSamples[i];
    }
    // wdt_reset();
    currentDistanceAverage = ceil((float)currentDistanceReading / (float)DistanceSensorSampleSize);
    currentDistanceAverage = (1.18*currentDistanceAverage)+7.31; //equation to compensation for detector
location
    ulCurrentDistanceAverage = currentDistanceAverage;
}
wdt_reset();
return ulCurrentDistanceAverage;
}

float getDistance() {
// Clears the trigPin
//Serial.println("was here");
wdt_reset();
digitalWrite(TRIGGER_PIN, LOW);
delayMicroseconds(2);
// Sets the trigPin on HIGH state for 10 micro second
digitalWrite(TRIGGER_PIN, HIGH);
delayMicroseconds(10);
digitalWrite(TRIGGER_PIN, LOW);
// Reads the echoPin, returns the sound wave travel time in microseconds
duration = pulseIn(ECHO_PIN, HIGH);
distance= duration*0.034/2;
wdt_reset();
return distance;
}

```

### Appendix III: Obtaining Scanner Orientation

```
const static unsigned int AngleSensorAveragingPeriodSeconds = 1;
const static unsigned int paramAngleISRFrequencyHz = 16;
const static unsigned int AngleSensorSampleSize =
(paramAngleISRFrequencyHz*AngleSensorAveragingPeriodSeconds);
static unsigned int AngleSamples[AngleSensorSampleSize] = {0};
#include <Wire.h> // Wire library - used for I2C communication
const int MPU = 0x68; // MPU6050 I2C address
float AccX, AccY, AccZ;
float GyroX, GyroY, GyroZ;
float accAngleX, accAngleY, gyroAngleX, gyroAngleY, gyroAngleZ;
float roll, pitch, yaw;
float AccErrorX, AccErrorY, GyroErrorX, GyroErrorY, GyroErrorZ;
float elapsedTime, currentTime, previousTime;
int c = 0;
void setupAngleSensor() {
  if (bAngleEmulation == true) {
  }
  else {
    Wire.begin(); // Initialize communication
    Wire.beginTransmission(MPU); // Start communication with MPU6050 // MPU=0x68
    Wire.write(0x6B); // Talk to the register 6B
    Wire.write(0x00); // Make reset - place a 0 into the 6B register
    Wire.endTransmission(true); //end the transmission
    // Call this function if you need to get the IMU error values for your module
    // calculate_IMU_error();
    delay(20);
  }
}
void sampleAngleSensor() {
static int sampleIndex = 0;

//if index reaches array size set to zero again
if ( sampleIndex == AngleSensorSampleSize ) {
  sampleIndex = 0;}
//add sample to array
AngleSamples[sampleIndex] = readSensor();
delay(20);
```

```

//increment index for next time
sampleIndex++;}
//call from anywhere to get latest smoothed value
float getAngle() {
  wdt_reset();
  unsigned long currentAngleReading = 0;
  float currentAngleAverage = 0.0;
  if (bAngleEmulation == true) {
    currentAngleAverage = 10; }
  else {
    //compute the current average ADC level
    for (int i = 0; i < AngleSensorSampleSize; i++) {
      // wdt_reset();
      currentAngleReading += AngleSamples[i]; }
    currentAngleAverage = ceil((float)currentAngleReading / (float)AngleSensorSampleSize); }
  wdt_reset();
  return currentAngleAverage;}
float readSensor(){
  float fCurrentDistanceAngle = 0.0;
  wdt_reset();
  Wire.beginTransmission(MPU);
  Wire.write(0x3B); // Start with register 0x3B (ACCEL_XOUT_H)
  Wire.endTransmission(false);
  Wire.requestFrom(MPU, 6, true); // Read 6 registers total, each axis value is stored in 2 registers
  //For a range of +-2g, we need to divide the raw values by 16384, according to the datasheet
  AccX = (Wire.read() << 8 | Wire.read()) / 16384.0; // X-axis value
  AccY = (Wire.read() << 8 | Wire.read()) / 16384.0; // Y-axis value

  AccZ = (Wire.read() << 8 | Wire.read()) / 16384.0; // Z-axis value
  // Calculating Roll and Pitch from the accelerometer data
  accAngleX = (atan(AccY / sqrt(pow(AccX, 2) + pow(AccZ, 2))) * 180 / PI) - 0.58; // AccErrorX ~(0.58)
  See the calculate_IMU_error() custom function for more details
  accAngleY = (atan(-1 * AccX / sqrt(pow(AccY, 2) + pow(AccZ, 2))) * 180 / PI) + 1.58; // AccErrorY ~(-
  1.58)
  // wdt_reset();
  // === Read gyroscope data === //
  previousTime = currentTime; // Previous time is stored before the actual time read
  currentTime = millis(); // Current time actual time read
  elapsedTime = (currentTime - previousTime) / 1000; // Divide by 1000 to get seconds

```

```

Wire.beginTransmission(MPU);
Wire.write(0x43); // Gyro data first register address 0x43
Wire.endTransmission(false);
Wire.requestFrom(MPU, 6, true); // Read 4 registers total, each axis value is stored in 2 registers
GyroX = (Wire.read() << 8 | Wire.read()) / 131.0; // For a 250deg/s range we have to divide first the raw
value by 131.0, according to the datasheet
GyroY = (Wire.read() << 8 | Wire.read()) / 131.0;
GyroZ = (Wire.read() << 8 | Wire.read()) / 131.0;
// Correct the outputs with the calculated error values
GyroX = GyroX + 0.56; // GyroErrorX ~(-0.56)
GyroY = GyroY - 2; // GyroErrorY ~(2)
GyroZ = GyroZ + 0.79; // GyroErrorZ ~(-0.8)
// Currently the raw values are in degrees per seconds, deg/s, so we need to multiply by seconds (s) to
get the angle in degrees
gyroAngleX = gyroAngleX + GyroX * elapsedTime; // deg/s * s = deg
gyroAngleY = gyroAngleY + GyroY * elapsedTime;
yaw = yaw + GyroZ * elapsedTime;
fCurrentDistanceAngle = lround (yaw);
updateCurrentDistanceValueToEEPROM(fCurrentDistanceAngle);
// Serial.println(yaw);
return yaw; }

float readDistance() {
float fCurrentDistance = 0;
getCurrentDistanceValueFromEEPROM(fCurrentDistance);
return fCurrentDistance;
}

```

## Appendix IV: Scanner State Transitions

```
boolean bNotifiedFaultState = false;
boolean bStartCyclic = false;
//process any state transitions that should occur due to events in the FuelPoint
// unsigned long ulParamPT;
// unsigned long ulParamTD;

void runStateTransitions() {
    wdt_reset();
    //switch to radiotracer mode
    if (gState == eRadioTracer ) {
        digitalWrite(UPLD_PIN, HIGH);
        digitalWrite(DOWNLED_PIN, HIGH);
        digitalWrite(TXLED_PIN, HIGH);
        digitalWrite(MODELED_PIN, HIGH);
    }
    //if we are scanning (obviously in the automatic mode, check on the scan progress
    if (gState == eFirstTime) {
        if ( CheckFirstScanProgress(true) == true ) { // && (checkStopPoint()== true )
            CheckFirstScanProgress(false);
            // Serial.println("Done with first scan but moving to the next position");
            initialiseScannerReporting(true);
            gState = eCyclic;
        }
        // sendGMData(true);
    }
    if (gState == eGammaManual) {
        // wdt_reset();
        digitalWrite(UPLD_PIN, LOW);
        digitalWrite(DOWNLED_PIN, LOW);
        digitalWrite(TXLED_PIN, HIGH);
        digitalWrite(MODELED_PIN, HIGH);
        gState = eGammaManual;
    }
    if (gState == eCyclic) {
        // Serial.println("eCyclic");
        if (checkStopPoint() == false) {
            moveScannerUp();
            // moveScannerUpBits();
            // delay(300);
        }
    }
}
```

```

//stopMovement();
//sampleDistanceSensor();
}
if (checkStopPoint() == true) {
    stopMovement();
    CyclicScan(true);
    if ((CheckCyclicScanProgress() == false)) {
        CyclicScan(true);
    }
    if ((CheckCyclicScanProgress() == true)) {
        initialiseScannerReporting(true);
        checkStopPoint();
    }
}
if (ulNextPausePoint > ulMaxDistance) {
    stopMovement();
    gState = eIdle;
}
}
if (gState == eMovingHomeUp ) {
    digitalWrite(UPLED_PIN, HIGH);
    digitalWrite(DOWNLED_PIN, HIGH);
    digitalWrite(TXLED_PIN, HIGH);
    digitalWrite(MODELED_PIN, HIGH);
    float fDist = getDistanceReading();
    if ((fDist >= 65 ) || (digitalRead(LIMITSWITCH_PIN) == HIGH)) {
        stopMovement();
    }
}

if (gState == eMovingHomeUp) {
    gState = eIdle;
}
}
else {
    moveScannerUp();
}
}
if (gState == eMovingHomeDown ) {
    // Serial.println(getDistanceReading());
}

```

```

digitalWrite(UPLED_PIN, HIGH);
digitalWrite(DOWNLED_PIN, HIGH);
digitalWrite(TXLED_PIN, HIGH);
digitalWrite(MODELED_PIN, HIGH);
float fDist = getDistanceReading();
if ((fDist <= 15 ) || (digitalRead(LIMITSWITCH_PIN))) {
    stopMovement();
    if (gState == eMovingHomeDown) {
        gState = eIdle;
    }
}
else {
    // Serial.print("Distance="); Serial.println(getDistanceReading());
    moveScannerDown();
}
}

if (gState == eIdle ) {
    digitalWrite(UPLED_PIN, LOW);
    digitalWrite(DOWNLED_PIN, LOW);
    digitalWrite(TXLED_PIN, LOW);
    digitalWrite(MODELED_PIN, LOW);
}
return;
}
/**
Returns the current controller state

*/
State getState() {
    return gState;
}
/**
Returns the controller state corresponding to the sState parameter
@param sState String representation of a state.
@param eState enumeration of the state per global state definition
@return true if the conversion was successful
*/
bool getStateFromString(String sState, State &eState) {

```



```
if (sState == "RT") {
    eState = eRadioTracer;
}
else if (sState == "FR") {
    eState = eFirstTime;
}
else if (sState == "MP") {
    eState = eMovingHomeUp;
}
else if (sState == "MD") {
    eState = eMovingHomeDown;
}
else if (sState == "GM") {
    eState = eGammaManual;
}
else if (sState == "CC") {
    eState = eCyclic;
}
else {
    eState = eIdle;
    return false; }
return true;
```

## Appendix V: Scanner Movement

```
#include <AFMotor.h>
unsigned long ulParamPauseTime = 5000;
unsigned long ulParamTravelDistance = 200;
unsigned long ulParamMD = 0;
unsigned long ulParamPT = 0;
float fParamTD = 0.0;
float fParamPT = 0.0;
unsigned long ulScanType = 0;
String sParamTD = "";
boolean bFirstScanStarted = true;
float currentDistanceAverage = 0.0;
bool bReset = false;
boolean bFirstScan = true;
boolean LimitState = true;    // variable for reading the limitswitch status
int Scan = 1;
boolean iResetState = true;
boolean iLastResetState = true;
boolean iStartState = true;
boolean iLastStartState = true;
float ulNextPausePoint = 0;
float ulNextNextPausePoint = 0;
static uint32_t time_now = 0;
unsigned long ulMaxDistance = 70;
void setupMotors() {
    pinMode(EN_A, OUTPUT);
    pinMode(IN1, OUTPUT);
    pinMode(IN2, OUTPUT);
    pinMode(IN3, OUTPUT);
    pinMode(IN4, OUTPUT);
    pinMode(EN_B, OUTPUT);
    pinMode(LIMITSWITCH_PIN, INPUT);
    pinMode(PWRLED_PIN, OUTPUT);
    pinMode(UPLED_PIN, OUTPUT);

    pinMode(DOWNLED_PIN, OUTPUT);
    pinMode(TXLED_PIN, OUTPUT);
    pinMode(MODELED_PIN, OUTPUT);
    //digitalWrite(PWRLED_PIN, LOW);
```

```

digitalWrite(UPLED_PIN, LOW);
digitalWrite(DOWNLED_PIN, LOW);
digitalWrite(TXLED_PIN, LOW);
digitalWrite(MODELED_PIN, LOW);
stopMovement();}
void executeGammaScannerAutoMode(bool bStartCyclic) {
  if (bStartCyclic == true) {
    // checkStopPoint();
    Serial.println("CYclci");
  }
  if (bStartCyclic == false)
  {
    // initialiseScannerReporting();
    Serial.println("out");
  }
}
bool CheckFirstScanProgress(bool bStartFirstScan) {
  iStartState = bStartFirstScan;
  fParamPT = getPauseTimeParam(fParamPT);
  time_now = millis();
  if ( time_now - iScanStartTime <= getPauseTimeParam(fParamPT)) {
    return false;
  }
  else {
    Serial.println("First scan is done");
    return true;
  }
  //initialiseScannerReporting(true);
  // return true;
}

bool CheckCyclicScanProgress() {

  if (CyclicScan(true) == false) {
    return false;
    //Serial.println("Scan goin on");
  }
  else {
    // Serial.println("scan is done,moving to the next point");
    return true;
  }
}

```

```

    //initialiseScannerReporting(true);
}
}
bool CyclicScan(bool bStartCyclic) {
if ( millis() - iScanStartTime <= fParamPT) {
    // sendGMDData(true);
    //delay(500);
    //wdt_reset();
    // Serial.print("Cyclic Scan Progress = "); Serial.println(millis() -iScanStartTime);
    return false;
}
else {
    return true;
}
}
bool checkStopPoint() {
    fParamTD = getTravelDistanceParam(fParamTD); // from EEPROM
    //sampleDistanceSensor();
    setSampleDistanceFlag();
    delay(30);
    float fDistance = getDistanceReading();
    if ((ulNextPausePoint) > ulMaxDistance) {
        stopMovement();
    }
    if (fDistance >= ulNextPausePoint) {
        Serial.print("stopped at: "); Serial.println(fDistance);
        initialiseFirstTimeScannerReporting(true);
        return true;
        initialiseScannerReporting(true);
    }
    return false;
    //return false;
}
void executeManualMode()
{
    gState = eGammaManual;
}
void initialiseScannerReporting(bool bReset) {
    //reporting variables

```

```

float fDistance = getDistanceReading();
iResetState = bReset;
Serial.println("resetting params.");
ulNextPausePoint = (fDistance + getTravelDistanceParam(fParamTD));
Serial.print("Current Distance = "); Serial.println (fDistance);
Serial.print("next pause = "); Serial.println(ulNextPausePoint);
if ( ulNextPausePoint > ulMaxDistance) {
    Alert_Tone(); delay(100); Alert_Tone();
    gState = eIdle;
}
}
void initialiseTimeScannerReporting(bool bReset) {
    //reporting variables
    gState = eCyclic;
    float fDistance = 0.0;
    iResetState = bReset;
    iScanStartTime = millis();
}
void initialiseFirstTimeScannerReporting(bool bReset) {
    //reporting variables
    gState = eFirstTime;
    float fDistance = 0.0;
    iResetState = bReset;
    iScanStartTime = millis();
}
void stopScanner() {

    digitalWrite(IN3, LOW);
    digitalWrite(IN4, LOW);
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, LOW);
}
void moveScannerUpBits() {
    digitalWrite(UPLED_PIN, HIGH);
    digitalWrite(DOWNLED_PIN, LOW);
    analogWrite(EN_B, 230);
    digitalWrite(IN3, HIGH);
    digitalWrite(IN4, LOW);
    Beep();
    delay(70);
}

```

```

analogWrite(EN_B, 0);
digitalWrite(IN3, LOW);
digitalWrite(IN4, LOW);
delay(30);
// setSampleDistanceFlag();
// sampleDistanceSensor();
// delay(100);
//Serial.println(getDistanceReading());
// delay(50);

}
void moveScannerUp() { //use EN_A, 255 , IN3 &IN4
  //Serial.println(getDistanceReading());
  analogWrite(EN_B, 240);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
  digitalWrite(UPLD_PIN, HIGH);
  digitalWrite(DOWNLED_PIN, LOW);
  Beep();
}
void moveScannerDown() {
  getDistanceReading();
  digitalWrite(DOWNLED_PIN, HIGH);
  digitalWrite(UPLD_PIN, LOW);
  analogWrite(EN_B, 150);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH); //REVERT
  delay(5);
}
void moveScannerRight() {
  digitalWrite(DOWNLED_PIN, LOW);
  digitalWrite(UPLD_PIN, LOW);
  analogWrite(EN_A, 255);
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  Beep();
  // Serial.println("right");
}
void moveScannerLeft() {
  digitalWrite(DOWNLED_PIN, LOW);

```

```
digitalWrite(UPLD_PIN, LOW);
analogWrite(EN_A, 255);
digitalWrite(IN1, HIGH);
digitalWrite(IN2, LOW);
// Serial.println("left");
Beep();
}
bool stopMovement()
{
  analogWrite(EN_B, 0);
  analogWrite(EN_A, 0);
  digitalWrite(UPLD_PIN, LOW);
  digitalWrite(DOWNLED_PIN, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, LOW);
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);
  return;
}
```

## Appendix VI: Data Receiver and Utility Checks

```
#include <SPI.h>
#include "nRF24L01.h"
#include "RF24.h"
#include "printf.h"

char cad[100];
int i=0;

/***** USER Configuration *****/
RF24 radio(34,37);          // Set up nRF24L01 radio connected on SPI bus plus pins 34 and 37
unsigned long timeoutPeriod = 3000; // Set a user-defined timeout period.
/*****/

const uint64_t pipes[2] = { 0xABCDABCD71LL, 0x544d52687CLL }; // Radio pipe addresses for the 2
nodes to communicate.
byte data[40] = {"_2,20,440,20,0,1"}; //Data buffer
volatile unsigned long counter;
unsigned long rxTimer,startTime, stopTime, payloads = 0;
bool transferInProgress = 0;
unsigned int TX=1,RX=0,role=1,lastrole=0,SKIP=2,RXPRINT=3,RXDUMP=4;
unsigned int offset=0;

char inData[50];
int newmessage = 0;
String sRxBufferRadio = "";

void setupRemote() {
  setupTXRadio();
  setupRXRadio();
}

void setupTXRadio()
{
  // Serial.begin(9600); // Debugging only
  if (!driver.init()){
    Serial.println("Receiver init failed");
  }
  else {
    Serial.println("Receiver Initialized Success");
  }
}
```



```

    }
}

void setupRXRadio() {
    printf_begin();
    radio.begin();           // Setup and configure rf radio
    radio.setChannel(1);     // Set the channel
    radio.setPALevel(RF24_PA_HIGH);
    radio.setDataRate(RF24_250KBPS);
    //radio.setDataRate(RF24_2MBPS);
    radio.setAutoAck(1);     // Ensure autoACK is enabled
    radio.setRetries(2,15);  // Optionally, increase the delay between retries. Want the number of
auto-retries as high as possible (15)
    radio.setCRCLength(RF24_CRC_16); // Set CRC length to 16-bit to ensure quality of data
    radio.openWritingPipe(pipes[0]); // Open the default reading and writing pipe
    radio.openReadingPipe(1,pipes[1]);
    radio.startListening();  // Start listening
    radio.powerUp();        //Power up the radio

    Serial.println(F("**----- REMOTE STARTED*****"));
}

void showData(void)
{
    printf("Data: ");
    for(int i=0; i<sizeof(data); i++){
        if(isprint(data[i]))

printf("%c", data[i]);
    }
    printf("\n\r");
}

bool ReadRF(){
    if(radio.available()){
        bDataPresent = true;
        radio.read(&data,sizeof(data)); //Read any available payloads for analysis
        showData();
    }
}

```

```

    writeMessageStringToEEPROM(data);
    return bDataPresent;
}
bDataPresent = false;
return bDataPresent;
}
//Extracts the radiations parameter from the received message string and stores it in the array
void exParamRD(String sMessage, unsigned long &ulParamRD) {
    String sParamName = "RD";
    String sParamValue = "";
    int iParamNameIndex = sMessage.indexOf(sParamName);
    if ( iParamNameIndex > -1 ) {
        int iStartOfValueIndex = sMessage.indexOf(":", iParamNameIndex); //parameter value starts after the :
        if ( iStartOfValueIndex > -1 ) {
            int iEndOfValueIndex = sMessage.indexOf(".", iParamNameIndex); //and ends with the ;
            if ( iEndOfValueIndex > - 1 ) {
                sParamValue = sMessage.substring(iStartOfValueIndex + 1, iEndOfValueIndex);
            }
        }
        ulParamRD = sParamValue.toInt();
        GBL_TdataNewPtr_RD++;
        if (GBL_TdataNewPtr_RD>324) GBL_TdataNewPtr_RD=0; //increment pointer and wrap it back to zero
        if needed
        GBL_Tdata_RD[GBL_TdataNewPtr_RD]=ulParamRD; //Store the sample into the trend array);
        Xaxiscounter = GBL_TdataNewPtr_RD;
    }
}

//Extracts the Distance parameter from the received message string and stores it in the array
void exParamDS(String sMessage, unsigned long &ulParamDS) {
    String sParamName = "DS";
    String sParamValue = "";
    int iParamNameIndex = sMessage.indexOf(sParamName);
    if ( iParamNameIndex > -1 ) {
        int iStartOfValueIndex = sMessage.indexOf(":", iParamNameIndex); //parameter value starts after the :
        if ( iStartOfValueIndex > -1 ) {
            int iEndOfValueIndex = sMessage.indexOf(".", iParamNameIndex); //and ends with the ;
            if ( iEndOfValueIndex > - 1 ) {
                sParamValue = sMessage.substring(iStartOfValueIndex + 1, iEndOfValueIndex);
            }
        }
    }
}

```

```

    }
    ulParamDS = sParamValue.toInt();
    GBL_TdataNewPtr_DS++;
    if (GBL_TdataNewPtr_DS>320) GBL_TdataNewPtr_DS=0; //increment pointer and wrap it back to
zero if needed
    GBL_Tdata_DS[GBL_TdataNewPtr_DS]=lroundf(ulParamDS); //Store the sample into the trend array
    Serial.println(ulParamDS);
}

```

//Extracts the angle parameter from the received message string and stores it in the array

```

void exParamAG(String sMessage, unsigned long &ulParamAG) {
    String sParamName = "AG";
    String sParamValue = "";
    int iParamNameIndex = sMessage.indexOf(sParamName);
    if ( iParamNameIndex > -1 ) {
        int iStartOfValueIndex = sMessage.indexOf(":", iParamNameIndex); //parameter value starts after the :
        if ( iStartOfValueIndex > -1 ) {
            int iEndOfValueIndex = sMessage.indexOf(";", iParamNameIndex); //and ends with the ;
            if ( iEndOfValueIndex > - 1 ) {
                sParamValue = sMessage.substring(iStartOfValueIndex + 1, iEndOfValueIndex);
                // return true;
            }
        }
    }
}

```

```

    ulParamAG = sParamValue.toInt();
    // Serial.println(ulParamDS);
    GBL_TdataNewPtr_AG++;
    if (GBL_TdataNewPtr_AG>320) GBL_TdataNewPtr_AG=0; //increment pointer and wrap it back to
zero if needed
    GBL_Tdata_AG[GBL_TdataNewPtr_AG]=lroundf(ulParamAG); //Store the sample into the trend
array
}

```

//Extracts the scan duration parameter from the received message string and stores it in the array

```

void exParamTS(String sMessage, unsigned long &ulParamTS) {
    String sParamName = "TS";
    String sParamValue = "";

    int iParamNameIndex = sMessage.indexOf(sParamName);

```

```

if ( iParamNameIndex > -1 ) {
    int iStartOfValueIndex = sMessage.indexOf(":", iParamNameIndex); //parameter value starts after the :
    if ( iStartOfValueIndex > -1 ) {
        int iEndOfValueIndex = sMessage.indexOf(";", iParamNameIndex);    //and ends with the ;
        if ( iEndOfValueIndex > - 1 ) {
            //parameter exists, return true
            sParamValue = sMessage.substring(iStartOfValueIndex + 1, iEndOfValueIndex);
            // return true;
            ulParamTS = sParamValue.toInt();
            GBL_TdataNewPtr_TS++;
            if (GBL_TdataNewPtr_TS>320) GBL_TdataNewPtr_TS=0; //increment pointer and wrap it back to zero
if needed
            GBL_Tdata_TS[GBL_TdataNewPtr_TS]=(ulParamTS); //Store the sample into the trend array
        }
    }
}
/**Takes a string value and returns its floating point numerical value. e.g, "19.9" -> 19.9
    @param sParam Value to be converted to a numerical value.
*/
float getFloatingPointNumericalValue(const String sParam) {
    return sParam.toFloat();
}
/**
    Takes a string value and returns its integer numerical value. e.g, "19" -> 19
    @param sParam Value to be converted to a numerical value.
*/
long getIntegerNumericalValue(const String sParam) {
    return sParam.toInt();
}

```

## Appendix VII: Gamma Chart Plotting Code

```

void PlotGammaAUTOTrendChart(int xStart, int yStart, int xEnd, int yEnd, int Horiz_GridSize, int
Vert_GridSize, float DataAcqPeriodMs, float TotalPlotMs) {
    for (int i = 0; i < GBL_TdataSize; i++) {
        GBL_Tdata_RD[i] = NoData; //random(600,700);
        GBL_Tdata_TS[i] = NoData; //random(600,700);
        GBL_Tdata_DS[i] = NoData; //random(600,700);
    }
    //===== Variables and Constants we use to plot data to the screen =====
    int PriorX = NoData;
    int PriorY = NoData;

```

```

//Define Just the Trend Graph Plot Area & Boundary Check Limits
int xGraphStart = xStart;    //Accommodate room for Vert-Axis Labels on Left Side of grid
//int xGraphStart=xStart+52;    //Accommodate room for Vert-Axis Labels on Left Side of grid
//int xGraphStart=xEnd-int(18*float(Horiz_GridSize));
int xGraphStart2 = xEnd - int(17.5 * float(Horiz_GridSize));
int yGraphTop = yStart;
int xGraphSize = xEnd - (xGraphStart) - 100;
int xGraphSize2 = xEnd - (xGraphStart2) - 100;
int yGraphSize = yEnd - yGraphTop - 22; //Accommodate room for Horz-Axis Labels on bottom of grid
int yGraphBottom = yGraphTop + yGraphSize; //y-TopLimit is the same as yStart; y-Bottom Limit =
Bottom of graph - (Character Ht + 3)
int xGraphLeft = xGraphStart2;

int xGraphRight = xGraphStart + xGraphSize;
int NumPointsToPlot = xGraphRight - xGraphLeft; //NumPointsToPlot on the graph
float MilliSecPerPixel = TotalPlotMs / NumPointsToPlot; //Used for Horz Scaling
float PlotTimeVsDataTimeRatio = (TotalPlotMs / NumPointsToPlot) / DataAcqPeriodMs; //Calc
PlotTimeRatio

int X, Y; //Coordinates of points to plot

//Define Trend_variables to hold starting coordinates and Scan Min/Max values
int TxStart, TyStart; //General purpose "X,Y Text-Starting-Coordinates" for use within this routine

float Tmax, Tmin, Tstep; //Reserve and Initialize variable to hold the Max & Min values we are currently
plotting

int numV_lines = xGraphSize / Horiz_GridSize;
float vGridSpacing = float(xGraphSize) / float(numV_lines);
int numH_lines = yGraphSize / Vert_GridSize - 1;
float hGridSpacing = float(yGraphSize) / float(numH_lines);

//Define variables needed during data scanning to find max, min, and average.
int ScanMax = 100;
int ScanMin = 0;
float Yrange = 10.;
float Yavg = ScanMax;
int TdataPlotPtr;

```

```

//===== Scan TrendData array to find min and max data values for the points we will be showing
for (int i = 0; i < NumPointsToPlot; i++) {
    TdataPlotPtr = GetDataPointer(i, PlotTimeVsDataTimeRatio); //Get the new data pointer
    if (TdataPlotPtr > 0) { //Error check...Did we get a valid pointer value returned?
        if (GBL_Tdata_RD[TdataPlotPtr] != NoData) { //Only look at valid data entries
            if (ScanMax == NoData) { //If this is the first data point we've found, set ScanMax and ScanMin to
that value
                ScanMax = GBL_Tdata_RD[TdataPlotPtr];
                ScanMin = ScanMax;
            }
            if (GBL_Tdata_RD[TdataPlotPtr] > ScanMax) ScanMax = GBL_Tdata_RD[TdataPlotPtr]; //Update
ScanMax and ScanMin values as needed
            if (GBL_Tdata_RD[TdataPlotPtr] < ScanMin) ScanMin = GBL_Tdata_RD[TdataPlotPtr];
        }
    }
}
Yavg = float(ScanMax + ScanMin) / 2.; //Calc the average value
//
//=====BEGIN AUTO SCALING ALGORITHM
//
// Here is where we select the correct vertical scale based on Largest and Smalled temps found in data scan
// This is where we determine the vertical plot-scaling we will be using

switch (ScanMax - ScanMin) {

    case 0 ... 100: //Range = 5.6-10.0 DegC P-P, 18.0 DegF
        Yrange = 100.;
        Tmax = 100;
        Tstep = 10;
        Tmin = Tmax - Yrange;
        break;
}

#ifdef DiagPrintEnabled
    DiagPrint("    Post Scaling-scan, MAX=", Tmax, ","); DiagPrint(" MIN=", Tmin, ""); DiagPrint("
Range=", Tmax - Tmin, ""); DiagPrintln(" Avg=", Yavg, "");
#endif

```

```

//Set axis font and then make the plot area background to grey...
//TODO // myGLCD.setFont(SanSerif_8x14);
tft.setFont(); //Select default font just in case... TODO Remove this backup code

//Paint Graphics Plot area background Color

tft.fillRect(xGraphStart2, yGraphTop, xGraphSize2, yGraphSize, GBL_Graph_BG);

int H_Adjust = 2;
const int GridLineColor = BLACK;
for (int i = 0; i < numH_lines + 1; i++) {
    int Vert_Axis_Lbl_Width = 28;
    tft.drawFastHLine(xGraphStart2, yGraphTop + i * hGridSpacing, xGraphSize2, GridLineColor);
    if (i % 2 == 0) { //Make every other line a slightly wider. Is "i" an EVEN number?
        //Yes i==EVEN, draw a single-wide line
        tft.drawFastHLine(xGraphStart2, yGraphTop + i * hGridSpacing + 1, xGraphSize2, GridLineColor);
    }
}
//===== DRAW VERTICAL GRID

H_Adjust = 2; //Minor fixed hori-offset determined empirically

for (int i = 0; i < numV_lines + 1; i++) { //Draw Single-Wide Grid Line
    tft.drawFastVLine(xGraphStart + i * vGridSpacing, yGraphTop, yGraphSize, GridLineColor);
    if (i % 2 == 0) { //Make every other line a slightly wider. Is "i" an EVEN number?
        //Yes i==EVEN, draw a second line to make it double-wide line
        tft.drawFastVLine(xGraphStart + i * vGridSpacing + 1, yGraphTop, yGraphSize, GridLineColor);
    }
}
//===== PRINT VERTICAL AXIS LABELS
float Grid_Ymax = -10000.;
float Grid_Ymin = 10000.;
tft.setTextColor(WHITE, GREY); //Foreground,Background text color
tft.setFont(&FreeSans9pt7b);

```

```

for (int i = 0; i < numH_lines + 1; i++) {

    TxStart = xStart + 1;

    float GridLabel = (Tmax - i * Tstep); //Grid Label value
    int DisplayedValue = GridLabel;
    //Determine X-starting coordinate of label

    TxStart = xStart + 1 + 8;

    TyStart = yGraphTop + 6;
    TxStart = xStart + 1; //This eliminates the centering-stuff in the above switch()...
    tft.setCursor(TxStart, TyStart + i * hGridSpacing);
    String NewText;
    NewText = String(GridLabel, 1); // 1 decimal point of precision...

    tft.fillRect(2, TyStart + i * hGridSpacing + 2, xGraphStart2 - 2, -(getTextHeight(NewText) + 2),
TFT_BLACK);

    tft.print(NewText);
    tft.setCursor(TxStart, 25);
    tft.print("CM");
    //Find Graph Max and Min values (in DegC)
    if (GridLabel > Grid_Ymax) Grid_Ymax = GridLabel;
    if (GridLabel < Grid_Ymin) Grid_Ymin = GridLabel; }
    //Capture and scale Y Min/Max Grid values so we can plot things properly
    Grid_Ymin = Grid_Ymin * 10;
    Grid_Ymax = Grid_Ymax * 10; //This is needed because incoming data is in 10ths of a degree C
#ifdef DiagPrintEnabled
    // DiagPrint(" Grid_Ymin=",Grid_Ymin," ");
    // DiagPrintln("Grid_Ymax=",Grid_Ymax," ");
#endif
    //===== PRINT NEW HORIZONTAL AXIS GRAPH LABELS
    int H_GridLabel = 1;
    float H_GridLabel_F = 0.0;
    float H_GridPeriodHrs_F = GBL_TotPlotHrs / (numV_lines - 1);
    DateTime now = rtc.now();
    int TOD_StartMinute = 1;

```



```

int LargestHourValue = 0;
tft.fillRect(xGraphLeft, ScrnHeight - 2, xGraphSize2, -(ScrnHeight - yGraphBottom - 2), TFT_BLACK);
int MaxTime = 300;
int TStep = 50;
for (int i = 0; i < 7; i++) {
//TOD_StartMinute=TOD_StartMinute+10;
float GridLabelX = (MaxTime - i * TStep); //Grid Label value (
    int DisplayedValue = GridLabelX;
    //Determine X-starting coordinate of label
    while (GridLabelX > 1000) {
        GridLabelX = 1000;
    }
    while (GridLabelX < 0) {
        GridLabelX = 0; //Range limit value to stay in 0-180 Hour range
    }
    TxStart = xStart + 1 + 8;

TyStart = yGraphTop + 6;
    TxStart = xStart + 1; //This eliminates the centering-stuff in the above switch()...
    String NewXText;
    NewXText = String(GridLabelX, 0); //1 decimal point of precision...

    tft.setCursor(350 - (i * 2 * vGridSpacing) - getTextWidth(NewXText) / 2, yEnd - 1);
    tft.print(NewXText); //xGraphRight

    tft.setTextColor(WHITE, GREY); //Foreground,Background text color
    tft.setFont(&FreeSans9pt7b);
    tft.setTextSize(1); //Make is 1X sized
    tft.setCursor(370, yEnd - 1);
    tft.print("(C/Sec)");
}

    GBL_CurDisplayHours = LargestHourValue; //Remember this value so other routines can detect a
change
    String TrendText = String(LargestHourValue, DEC);

    TrendText += "Scan Height(CM) vs Counts/Sec";

```

```

TrendText += " ";
//Now more or less center the text inside of the grid area; This is an approximation based on mono-spaced
characters...
//Get Width of the printed text string,,,
int TrendTextWidth = getTextWidth(TrendText);
//Center Text in GRAPH region
int X_TrendText = xGraphLeft + xGraphSize2 / 2 - TrendTextWidth / 2; //Calculate Starting X-coord to
center the text string (about 31 Characters)
X_TrendText = constrain(X_TrendText, xGraphLeft, xGraphRight - TrendTextWidth); //Make sure we're
in-bounds
tft.setFont(&FreeSans9pt7b);
tft.setTextSize(1); //Make is 1X sized
int16_t X_erase, Y_erase, dX_erase, dY_erase;

tft.getTextBounds(TrendText, X_TrendText, yStart + 20, &X_erase, &Y_erase, &dX_erase, &dY_erase);
//Clear background where label will be to get rid of grid lines
uint16_t const B_Spc = 3; //B_Spc (in pixels) = extra boarder space around text to be sure enough
background has been cleared
tft.fillRect(X_TrendText, 20, dX_erase + 2 * B_Spc, dY_erase + 2 * B_Spc, GBL_Graph_BG); //Make
text background GREY
tft.setCursor(X_TrendText + 8, 37);
tft.setTextColor(WHITE);
tft.print("Scan Height(CM) vs Counts/Sec"); //Print graph label
tft.setCursor(400, 15);
tft.setTextColor(WHITE);
tft.print("GammaA"); //Print graph label
int CurMarkerX = NoData;
int CurMarkerY = NoData;
GBL_TdataNewPtr_RD = 0;
Plot_Gamma_AUTO_OvrView_Buttons(TFT_BLUE, TFT_WHITE);
Menu = 8; while (Menu == 8) {
//Listen to buttons for Abort====
ReadRF();
getMessageStringFromEEPROM(sMessage);
if (digitalRead(MenuExitPB_Pin) == HIGH)
{ driver.send("-setST TY:4;PT:0;TD:0;", 27);
driver.waitPacketSent();
tft.fillScreen(BLACK);
Display_MODE_SELECTION_Screen(TFT_LIGHTGREY);
//Abort_Warning(); }

```

```

//===== PLOT ACTUAL TREND DATA POINTS TO SCREEN
if (bDataPresent == true) {
    exParamRD(sMessage, ulParamRD);
    exParamDS(sMessage, ulParamDS);
    exParamAG(sMessage, ulParamAG);
    PlotGammaAutoChart();
} }
//===== End Screen Plotting}

```

## Appendix VIII: Plotting Radio Tracer Chart Points

```

void PlotRTracerChart() {
    int PointColor = GBL_Outdoor_Temp_FG;
    Serial.print ("Pointer value: "); Serial.println(Xaxiscounter);
    X = Xaxiscounter + 54;
    Serial.print ("X value: "); Serial.println(X);
    y_pos_x = GBL_Tdata_RD[Xaxiscounter];
    Serial.print("Y Value: "); Serial.println(y_pos_x);
    if (X > 368) {
        driver.send("-setST TY:4;PT:0;TD:0;", 28);
        driver.waitPacketSent();
        delay(100);
    }
    if ( y_pos_x != NoData) {
        // Serial.println(X);
        Y = map(y_pos_x, 0, 500, 290, 42);
        Y = constrain (Y, 42, 291); //Final check to keep XY point inside the define graph area
        if (X > 55 && X < 375) {
            tft.drawLine(PriorXval, PriorYval, X, Y, PointColor);
            tft.drawLine(PriorXval, PriorYval + 1, X, Y + 1, PointColor); //Thicken line in Y
            PriorYval = Y;
            PriorXval = X;
        }
    }
    else {
        PriorXval = 0;
        PriorYval = 0;
    }
}

```

## Appendix IX: Detector Code

```

//radiation counts variables
volatile uint32_t nCounts = 0; //incremented by interrupt from radiation counts
uint32_t nTargetCounts = 0; //number of Counts we should get to when dispensing fixed folume
uint32_t nNewCounts = 0; //Counts received since last report from radiation counts
uint32_t nLastCounts = 0; //Counts received at last radiation counts report
uint32_t lastMeasurementTime = 0; //Counts received at last radiation counts report
uint32_t pulseIntervalMillis = 0; //calculated time between Counts
const uint32_t reportIntervalMilliseconds = 1000; //time between radiation reports
uint32_t currentRadiations = 0;

void setupRadiationSensor() {

    pinMode(SIGN_PIN, INPUT);
    attachInterrupt(digitalPinToInterrupt(SIGN_PIN), countsInterrupt, RISING);
}

float ObtainRadiations()
{
    wdt_reset();
    // uint32_t currentRadiations = 0;
    if (bDetectorEmulation == true) {
        currentRadiations = random (750,970);
    }
    else {
        if ( millis() >lastMeasurementTime + reportIntervalMilliseconds) {
            //flow rate
            currentRadiations = nCounts ;
            // return currentRadiations;
            // Serial.println(currentRadiations);
            nCounts = 0;
            lastMeasurementTime = millis();
        } }
    if (currentRadiations >=950){
        currentRadiations = 950;
    }
    //updateCurrentRadiationValueToEEPROM(currentRadiations);
    // Serial.println(currentRadiations);
    wdt_reset();
    // Serial.println(currentRadiations);

```

```

    return currentRadiations;}
//initialise radiation counts variables before a scan
void initialiseDetector() {
    nCounts = 0;
    nNewCounts = 0;
    nLastCounts = 0;
    CountsIntervalMillis = 0;
    lastMeasurementTime = millis();}
/* radiation counts code */
void countsInterrupt() {
    //low to high transition!
    nCounts++;
}
Send data

    void sendGMData(bool bIsHeartbeat)
    {
        wdt_reset();
        String statusMsg = "";
        if (bIsHeartbeat == true) {
            float fDist = getDistanceReading();
            float fAngle = 0;
            statusMsg += "DS:" + String(fDist); //Distance "DS" getDistance
            statusMsg += ";RD:" + String(ObtainRadiations()); //Radiatoin count/sec "RD"
            statusMsg += ";AG:" + String(fAngle);
            statusMsg += ";";
            statusMsg.toCharArray(ConvStatusMsg, 25); //convert serialdat the the msg char array
            wdt_reset();
            radio.writeFast(&ConvStatusMsg,25);
            delay(30);
            Serial.println(statusMsg);
            wdt_reset( )

```

## **Appendix X: nRFL2401 and 433MHz RF Code**

```

#include <SPI.h>
#include "nRF24L01.h"
#include "RF24.h"
#include "printf.h"
/***** USER Configuration *****/
RF24 radio(42,44); // Set up nRF24L01 radio on SPI bus plus pins 7 & 8

```

```

unsigned long timeoutPeriod = 3000; // Set a user-defined timeout period. With auto-retransmit set to
(15,15) retransmission will take up to 60ms and as little as 7.5ms with it set to (1,15). // With a
timeout period of 1000, the radio will retry each payload for up to 1 second before giving up on the
transmission and starting over
/*****/
const uint64_t pipes[2] = { 0xABCDABCD71LL, 0x544d52687CLL }; // Radio pipe addresses for the 2
nodes to communicate.
byte data[32] = {"_2,20,440,20,0,1"}; //Data buffer
volatile unsigned long counter;
unsigned long rxTimer,startTime, stopTime, payloads = 0;
//bool TX=1,RX=0,role=0, transferInProgress = 0;
bool transferInProgress = 0;
unsigned int TX=1,RX=0,role=1,lastrole=0,SKIP=2,RXPRINT=3,RXDUMP=4;
unsigned int offset=0;

//pinMode(TXLED_PIN, OUTPUT);
// pinMode(TXLED_PIN, OUTPUT);

boolean bAutomaticMode = false; //Boolean to indicate if system is on automode or manual mode
boolean bManualMode = true; //Boolean to indicate if system is on automode or manual mode
unsigned long ulParamOperationMode = 1; //Config parameter that is used to set bAutomaticMode
boolean bStartAutomaticModeOperation = false;
String sOperationMode = " ";
//sets the system operation mode
void setOperationMode() {
  if (ulParamOperationMode == 1 ) {
    bAutomaticMode = true;
    bStartAutomaticModeOperation = true;
    StartAutomaticModeOperation();

  } else {
    bAutomaticMode = false;
    bManualMode = true;
  }
}
void radioData(){
}
String checkForOperationMode() {
  if (ulParamOperationMode == 1 ) {
    return "1";
  }
}

```

```

} else {
  return "2";
}
}

void StartAutomaticModeOperation(){
  if (bStartAutomaticModeOperation == true ) {
    String sPauseTime;
    String sTravelDistance;
    // String scanStatus = executeGammaScannerAutoMode(sPauseTime,sTravelDistance);
    bStartAutomaticModeOperation = false;
  }
}

void setupTXRadio() {
  // printf_begin();
  radio.begin();           // Setup and configure rf radio
  role = TX;
  radio.setChannel(1);     // Set the channel
  radio.setPALevel(RF24_PA_HIGH);
  radio.setDataRate(RF24_250KBPS);
  //radio.setDataRate(RF24_2MBPS);
  radio.setAutoAck(1);     // Ensure autoACK is enabled
  radio.setRetries(2,15);  // Optionally, increase the delay between retries. Want the number of
  auto-retries as high as possible (15)
  radio.setCRCLength(RF24_CRC_16); // Set CRC length to 16-bit to ensure quality of data

  radio.openWritingPipe(pipes[1]); // Open the default reading and writing pipe
  radio.openReadingPipe(1,pipes[0]);
  radio.stopListening();// if (role == RX)
  radio.powerUp();         //Power up the radio
  Serial.println(F("*** RADIO STARTED IN TX MODE*****"));

}

void setupRXRadio()
{
  // Serial.begin(9600); // Debugging only
  if (!driver.init()){

```

```

    Serial.println("Receiver init failed");
}
else {
    Serial.println("Receiver Initialized Success") }}

```

### **Appendix XI: Command Checking and Execution.**

```

String sFaultCode = "None";
#include <RH_ASK.h>
// #include <SPI.h> // Not actually used but needed to compile
// boolean bSampleDistanceFlag = false;
// pinMode(2, OUTPUT);
RH_ASK driver;
char cad[50];
int pos = 0;
byte message[50];
// byte messageLength = sizeof(message);
byte buf[50];
byte buflen = sizeof(buf);
// called to check for any incoming messages on the 433MHz radio
void serviceCommands() {
    String sCommandUSB = GetRadioCommand();
    if (sCommandUSB != "") {
        processCommandUSB(sCommandUSB);
    }
}
String GetRadioCommand()
{
    byte messageLength = sizeof(message);
    String sRxBufferRadio = "";
    String sReceivedCommand = "";
    sRxBufferRadio.reserve(35);
    if (driver.recv(message, &messageLength))
    {
        for (int i = 0; i < messageLength; i++)
        {
            // sRxBufferRadio = (char)message;
            // sReceivedCommand = sRxBufferRadio;
            // Serial.println(messageLength);
            // Serial.write(message[i]);

```



```

delay(50);
//Serial.println(message[i]);Serial.println(" ");
//delay(2000);
char inChar = (char)message[i];    // read a byte and cast as a character
if ( inChar == '-') {              // if the incoming character is a newline the message is complete
    //if (gbNewCommand == false) {    // If the last message has been acknowledged (not pending
processing)
    // Serial.println("start of a new message");
    int startChar = sRxBufferRadio.indexOf('-'); // Finds the start character for a message (an exclamation
mark). If any crap on the serial port was added to the buffer before this, we will ignore it.
    if (startChar > - 1) {
        sReceivedCommand = sRxBufferRadio.substring(startChar + 1); //Start character present, remove all
characters upto and including it
    }
    else {
        sReceivedCommand = sRxBufferRadio;    // No start character, assign message to instruction
variable
    }
    //}
    sRxBufferRadio = "";    //wipe the receive buffer
}
//
else {    // If the character is not a newline character then message is incomplete
    sRxBufferRadio += inChar;    // add the char to the receive buffer
    sReceivedCommand = sRxBufferRadio;;
}
}
Serial.println(sReceivedCommand);
}
return sReceivedCommand;
}
//process commands received over the serial port from Fuel Point Manager
void processCommandUSB(String sCommand) {

//get the instruction part of the command (which API message is it)
String sInstruction = getInstruction(sCommand);
// boolean bSampleDistanceFlag = false;

```

```

//set_config command
if ( sInstruction == SETSCANTYPE_INSTRUCTION ) {
    String sScanTypeConfigString = getParameter(sCommand);
    applyReceivedScanTypeConfigString(sScanTypeConfigString);
    return;
}
//get_config command
if ( sInstruction == GETSETSCANTYPE_INSTRUCTION ) {
    sendCommandSuccessResponse(GETSETSCANTYPE_COMPLETED + getScanType());
    return;
}
//get the instruction part of the command (which API message is it)
// status instruction
if ( sInstruction == STATUS_INSTRUCTION ) {
    // getStatus(false);
    // sendRadioData(true);
    sendGMData(true);
    return;
}
//scanner instructions
if ( (sInstruction == MOVE_UP_INSTRUCTION) && (getDistanceReading() < 75)) {
    moveScannerUp();
    delay(250);
    stopMovement();
    return;
}
if ( (sInstruction == MOVE_DOWN_INSTRUCTION && (getDistanceReading() > 12)) {
    moveScannerDown();
    delay(100);
    stopMovement();
    return;
}
if ( sInstruction == MOVE_RIGHT_INSTRUCTION ) {
    moveScannerRight();
    delay(100);
    stopMovement();
    return;
}
if ( sInstruction == MOVE_LEFT_INSTRUCTION ) {
    moveScannerLeft();
}

```

```

delay(100);
stopMovement();
return;
}
if(( sInstruction == MOVE_HOME_UP_INSTRUCTION ) && (getDistanceReading() < 74)) {
    gState = eMovingHomeUp;
    // moveScannerHome(true);
    return;
}
if(( sInstruction == MOVE_HOME_DOWN_INSTRUCTION ) && (getDistanceReading() > 10)) {
    gState = eMovingHomeDown;
    // moveScannerHome(true);
    return;
}
if( sInstruction == ABORT_SCAN_INSTRUCTION ) {
    stopMovement();
    return;
}
//watchdog reset instruction
if( sInstruction == KPWD_RESET ) {
    resetSystemWatchdog();
    sendCommandSuccessResponse(KPWD_RESET_MESSAGE);
    return;
}
//timeout watchdog instruction
if( sInstruction == KPWD_TIMEOUT ) {
    timeoutSystemWatchdog();
    sendCommandSuccessResponse(KPWD_TIMEOUT_MESSAGE);
    return;
}
if(sInstruction == CONTROLLER_STATUS_INSTRUCTION) {
    delay(300);
    sendCommandSuccessResponse(CONTROLLER_STATUS_UPDATE_MESSAGE + gsState[gState]);
}
//set serial data to EEPROM
if( sInstruction == ABORT_AUTO_SCAN_INSTRUCTION ) {
    sendCommandSuccessResponse(EEPROM_SERIAL_SET_COMPLETED);
}
/*
//return serial data

```

```

if ( sInstruction == EEPROM_SERIAL_GET ) {
    sendCommandSuccessResponse(EEPROM_SERIAL + getKpcId());
    return;
}
//delete serial data from EEPROM
if ( sInstruction == EEPROM_SERIAL_DELETE ) {
    deleteKpcIdFromEEPROM();
    setupKpcId();
    sendCommandSuccessResponse(EEPROM_SERIAL_DELETE_COMPLETED);
    return;
}
*/
//fault codes
if ( sInstruction == FAULT_CODE_INSTRUCTION ) {
    if (sFaultCode != "None") {
        sendCommandSuccessResponse(FAULT_CODE_MESSAGE + sFaultCode);
    } else {
        sendCommandErrorResponse(ERROR_NO_FAULT, "Failed to get a fault code");
    }
}
return;
}
//else instruction process based on controller's state
switch (gState) {

case eCyclic:
    if (sInstruction == ABORT_SCAN_INSTRUCTION) {
        gState = eIdle;
    }
    else {
        sendCommandErrorResponse(ERROR_STATE_INVALID_COMMAND, "Invalid command " +
sInstruction + " received in " + gsState[gState] + " state");
    }
    break;
case eRadioTracer:
    if (sInstruction == ABORT_SCAN_INSTRUCTION) {
        gState = eIdle;
    }
    else {
        sendCommandErrorResponse(ERROR_STATE_INVALID_COMMAND, "Invalid command " +
sInstruction + " received in " + gsState[gState] + " state");

```

```

    }
    break;
    /* case eIdle:
    //set_config command
    if ( sInstruction == SETGAMMAMODECONFIG_INSTRUCTION ) {
        String sGammaModeConfigString = getParameter(sCommand);
        //Attempt to apply the config string from app. If all parameters are ok, they will be applied. Else the
default values will be used
        //and an alert will be raised for each invalid/missing parameter.
        applyReceivedGammaModeConfigString(sGammaModeConfigString);
        gState = eGammaManual;
    }
    break; */
}
wdt_reset();
}
//extract the instruction from the command
String getInstruction(String sCommand) {
    String sInstruction = "";
    int indexOfSpace = sCommand.indexOf(' ');
    //if no space then the instruction is simply the command
    if (indexOfSpace == -1) {
        sInstruction = sCommand;
    } else {
        sInstruction = sCommand.substring(0, indexOfSpace);
    }
    if (DEBUG_OUTPUT) {
        Serial.println("Parsed instruction " + sInstruction);
    }
    return sInstruction;
}
//extract the parameter from the command
String getParameter(String sCommand) {
    //bSampleDistanceFlag = false;
    String sParameter = "";
    int indexOfSpace = sCommand.indexOf(' ');
    //if no space then the instruction is simply the command
    if (indexOfSpace == -1) {
        sParameter = "";
    } else {

```

```

    sParameter = sCommand.substring(indexOfSpace + 1);
}
if (DEBUG_OUTPUT) {
    Serial.println("Parsed parameter " + sParameter);
}
return sParameter;
}
//prints the command success response
void sendCommandSuccessResponse(String successMessage) {
    Serial.println(successMessage);
}
//prints the command error response code and creates an alert for the error
void sendCommandErrorResponse(String errorCode, String errorMessage) {
    Serial.println(INSTRUCTION_ERROR_MESSAGE + errorCode);
    // sendAlert(errorCode, errorMessage);
}

```

## Appendix XII: EEPROM code

```

#include <EEPROM.h>

const static uint32_t eepromAddr_messageString = 100;

/**
 * Writes the sMessage string to EEPROM at address eepromAddr_messageString, and also updates the
 * length of the string to EEPROM.
 * @param sMessage The value to be written to EEPROM
 */
void writeMessageStringToEEPROM(String sMessage) {
    int iNextWriteAddress = eepromAddr_messageString;
    for (int i = 0; i < sMessage.length(); i++) {
        eepromWrite(iNextWriteAddress, sMessage.charAt(i));
        iNextWriteAddress++;
    }
    //Write null characters at each index after the last config string index up to the last config string allocated
    index
    while (iNextWriteAddress < eepromAddr_messageString + 50) {
        eepromWrite(iNextWriteAddress, 0);
        iNextWriteAddress++;
    }
}

```

```
/**  
    Reads the value of the string in the EEPROM and assigns it to the variable referenced by sMessage  
    pointer.
```

```
    @param sMessage A pointer to the variable to be assigned with the value read from the EEPROM  
*/
```

```
void getMessageStringFromEEPROM(String &sMessage) {  
    String sMessageFromEEPROM = "";  
    int iStartReadAddress = eepromAddr_messageString;  
    for (int i = iStartReadAddress; i < iStartReadAddress + 50 ; i++) {  
        char cNextChar = (char)EEPROM.read(i);  
        if (cNextChar == 0) {  
            break;  
        }  
        else {  
            sMessageFromEEPROM += cNextChar;  
        }  
    }  
    sMessage = sMessageFromEEPROM;  
    // Serial.println(sMessage);  
}
```

```
/**  
    Utility method that accepts any data type and writes the data to EEPROM.
```

```
    @param iAddressToWrite address to write the data value
```

```
    @param a reference to the data to write to EEPROM
```

```
*/  
template <class T> int eepromWrite(int iAddressToWrite, const T& value) {  
    const byte* p = (const byte*)(const void*)&value;  
    unsigned int i;  
    for (i = 0; i < sizeof(value); i++) {  
        EEPROM.update(iAddressToWrite++, *p++);  
    }  
    return i;  
}
```

```
/**  
    Utility method that reads data written in the EEPROM
```

```
    @param iAddressToRead address to start reading the data value from
```

```
    @param value a reference to the variable where the read value will be assigned.
```

```
*/  
template <class T> int eepromRead(int iAddressToRead, T& value) {
```

```
byte* p = (byte*)(void*)&value;
unsigned int i;
for (i = 0; i < sizeof(value); i++) {
    *p++ = EEPROM.read(iAddressToRead++);
}
return i;
```