

DESIGN AND DEVELOPMENT OF
IN SITU FPGA-BASED WATER QUALITY
MONITORING KIT

BY

ABDULRAHMAN BAHAA ZAIDAN

A dissertation submitted in fulfilment of the requirement for
the degree of Master of Science in Computer and
Information Engineering

Kulliyyah of Engineering
International Islamic University Malaysia

November 2022

ABSTRACT

In 2017, about 144 million people collected water from untreated water bodies, such as lakes, streams, and rivers. One of the major causes of death is consuming contaminated or polluted water. Measuring and monitoring water quality are usually done using two methods. The conventional method occurs by taking samples of water and then transferring them to the laboratory. The second method is real-time water quality by integrating the Internet of Things (IoT). This method is preferable as it only requires smart sensors and processors to monitor the water quality. Among the widely used processors are the Arduino and Raspberry Pi. However, these two processors have a limitation, including a limited number of hard-coded input/output pins, unlike the Field Programmable Gate Array (FPGA) processor, which has many input/output pins not hard-coded to allow different interfacing of multiple sensors. Based on the literature, an FPGA platform provides more flexibility and reconfigurability features when compared with the Arduino and Raspberry Pi. This research mainly focuses on designing a reconfigurable multi-core Smart Water Quality System (SWQS) measuring the pH, Total Dissolved Solids (TDS), and turbidity parameters. The hardware design was developed based on the system-on-chip (SoC) design methodology on an FPGA to parallelize the SWQS functionality. A Liquid-Crystal Display (LCD) display has been incorporated into the Raspberry Pi to show real-time data. The Platform Designer on Quartus II has been used to instantiate four cores to integrate all functions into one processor. The Eclipse tool on Quartus II, on the other hand, was used to program the sensors using embedded C language. The proposed design has been implemented on DE10 Nano FPGA-SoC consuming 9% of logic resources and 57% of internal memory. To verify the proposed system functionality, the sensors were tested on different liquids. To test the pH level, the pH sensor was tested on pure water, lemon juice, and milk to show the acidity and alkalinity. The pH sensor showed 7, less nearly 2, and less than 8 for pure water, lemon juice, and milk, respectively. The TDS sensor successfully detected the salt added to the water, and the TDS values increased to approximately 1800 ppm. Finally, the turbidity sensor revealed the dust inserted in the solution. The more dust in the liquid, the more TDS value there was recorded. Additionally, results showed that the processing time of all the sensors using FPGA is approximately 300 ms for ten readings; on the other hand, the processing time of using other processors, such as Arduino, took 2 s for ten readings. This is because FPGA is functioning at 100 MHz, while Arduino's frequency is not more than 24 MHz. All real-time sensor readings were shown on a Linux Terminal. In conclusion, the proposed FPGA-based system can be utilized as a heterogeneous multi-core system for many applications, including the SWQS.

ملخص البحث

أكثر من مائة وأربع وأربعون مليون إنسان يأخذون احتياجاتهم من المياه من مصادر غير معالجة مثل الأنهار والبحيرات. في بعض الدول النامية، معظم المصانع تقوم برمي الفضلات في المياه وهذا يجعل المياه بالقرب من المصانع أكثر تلوثاً لأنها تحمل مواد كيميائية بالإضافة إلى المواد الثقيلة. هذه المواد تؤثر بشكل سلبي على البيئة وعلى الكائنات الحية التي تعيش في المياه لأن المواد الثقيلة تنشر الفيروسات والبكتيريا. في هذه الأيام يتم قياس جودة المياه بطريقتين هما أما الطريقة التقليدية وهي عبارة عن أخذ عينات من المياه ونقلها إلى المختبر ومن ثم قياس جودتها، ولكن هذه الطريقة تحتاج إلى وقت وتكلفه أكثر. بالإضافة إلى ذلك، حالة المياه من الممكن أن تتغير خلال عملية النقل. أما الطريقة الثانية فهي قياس جودة المياه باستخدام الحساسات الذكية مع المعالجات ونقل البيانات بطرق مختلفة. هذه الطريقة مفضلة بشكل أكبر من الطريقة التقليدية لأنها فقط تحتاج إلى الحساسات الذكية بالإضافة إلى المعالج لقياس جودة المياه في كل وقت. هذه الطريقة ستساعد المستخدم على اتخاذ القرار بسرعة في الحالات المفاجئة. لهذا فإن هذا المشروع يركز بشكل أساسي على استخدام حساسات لقياس الأس الهيدروجيني والعكورة بالإضافة إلى كمية المواد الذائبة. بعد ذلك تقوم مصفوفة البوابات المنطقية القابلة للبرمجة بمعالجة البيانات وإرسالها لأجل عرضها على الشاشة. هذا المشروع سيقبل من تعقيد الأجهزة الأخرى المستخدمة في قياس جودة المياه. وسوف تقوم باستخدام الأجهزة مع البرمجة لتقليل الوقت المستخدم في تطوير جهاز متحسس المياه الذكي. أخيراً وليس آخراً، جهاز الراسبري باي سيستخدم فقط لعرض البيانات على الشاشة المتربطة به. سيقوم معالج الراسبري باي بأستلام البيانات عن طريق تطبيق مبرمج بداخله ومن ثم عرضها على الشاشة. الجهاز المصمم قد تم اختباره على سوائيل مختلفة مثل المياه النقية، عصير الليمون، الحليب لقياس الأس الهيدروجيني.


اما نسبة العكورة فقد تم استخدام المياه النقيه بالاضافه الى مياه مع القليل من الرواسب ومياه مع الكثير من الرواسب. اخيرا فقد تم استخدام المياه النقيه ومياه تحتوي على الملح لقياس المواد المذابة في الماء. عن طريق هذا المشروع يمكن استخدام البيانات عن طريق ربطها ببرامج اتخاذ القرار, تحليل البيانات وغيرها من البرامج.

APPROVAL PAGE

I certify that I have supervised and read this study and that, in my opinion, it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Master of Science Engineering.


.....
Amelia Wong Binti Azman
Supervisor

14/11/22


.....
Huda Adibah Mohd Ramli
Co-Supervisor

I certify that I have read this study and that, in my opinion, it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Master of Science Engineering..

.....
Rosminazuin Ab. Rahim
Internal Examiner

.....
Suriza Ahmad Zabidi
Internal Examiner

This dissertation was submitted to the Department of Electrical and Computer Information and is accepted as a fulfilment of the requirement the degree of Master of Science Engineering.

.....
Rafiqul Islam
Head, Department of Electrical and
Computer Information Engineering


This dissertation was submitted to the Kulliyah of Engineering and is accepted as a fulfilment of the requirement for the degree of Master of Science Engineering.

.....
Sany Izan Ihsan
Dean, Kulliyah of Engineering

DECLARATION

I hereby declare that this dissertation is the result of my investigations, except where otherwise stated. I also declare that it has not been previously or concurrently submitted as a whole for any other degrees at IIUM or other institutions.

Abdulrahman Bahaa Zaidan

Signature.....

Date 2022/12/05

INTERNATIONAL ISLAMIC UNIVERSITY MALAYSIA

**DECLARATION OF COPYRIGHT AND AFFIRMATION OF
FAIR USE OF UNPUBLISHED RESEARCH**

**DESIGN AND DEVELOPMENT OF
IN SITU WATER QUALITY KIT**

I declare that the copyright holders of this dissertation are jointly owned by the Student and IIUM.

Copyright © 2022 Abdulrahman Bahaa Zaidan and International Islamic University Malaysia.
All rights reserved.

No part of this unpublished research may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording or otherwise without prior written permission of the copyright holder except as provided below

1. Any material contained in or derived from this unpublished research may only be used by others in their writing with due acknowledgement.
2. IIUM or its library will have the right to make and transmit copies (print or electronic) for institutional and academic purposes.
3. The IIUM library will have the right to make, store in a retrieval system and supply copies of this unpublished research if requested by other universities and research libraries.

By signing this form, I acknowledged that I have read and understand the IIUM Intellectual Property Right and Commercialisation policy.

Affirmed by Abdulrahman Bahaa Zaidan



2022/12/05

.....
Signature

.....
Date

ACKNOWLEDGEMENTS

First and foremost, I would like to thank Allah for blessing me each and every day. Thanks for the strength, wisdom, and faith to make it through life's everyday challenges.

I would like to thank my parents. There are so many things that I want to thank them for. They are the reason behind all of my success, and I am forever indebted to them. I would especially like to thank Associate Professor Amelia Wong Binti Azman for her outstanding guidance, advice, inspiration, and encouragement throughout my research progress. It is my greatest honor to work under her supervision. Thank you for nurturing me into a hard-working student and for all your patience and guidance. You were more than just supervisors to me. I am deeply indebted to you.

In addition, my gratitude goes to my beloved, lovely wife; for her prayers, understanding, and endurance while away.

Finally, I want to thank Fawaz Mohammed for his countless midnight calls and messages just to do this research; I am so grateful and thankful to you. I am very thankful for all of the above for being the solid support system in my life. Thank you. Allah blesses you

Once again, we glorify Allah for His endless mercy on us, one of which is enabling us to successfully round off the efforts of writing this thesis. Alhamdulillah.

TABLE OF CONTENTS

Abstract	ii
Abstract in Arabic	iii
Approval Page.....	v
Declaration	vi
Copyright Page.....	vii
Acknowledgements	viii
Table of Contents	ix
List of Tables	xii
List of Figures	xiii
CHAPTER ONE: INTRODUCTION	1
1.1 Overview.....	1
1.2 Research Background	1
1.2.1 Field Programmable Gate Array (FPGA).....	2
1.2.2 Heterogeneous System Architecture (HSA)	3
1.3 Problem Statement.....	4
1.4 Research Objectives.....	6
1.5 Research Scope	6
1.6 Dissertation Layout.....	7
1.7 Summary.....	7
CHAPTER TWO: LITERATURE REVIEW	8
2.1 Overview.....	8
2.2 Different Water Quality Approaches.....	8
2.2.1 Conventional Approach	9
2.2.2 Internet of Things (IoT) Approach	10
2.3 Sensors Used in Monitoring Water Quality Parameters.....	11
2.3.1 pH Sensor.....	11
2.3.2 Temperature Sensor	12
2.3.3 Turbidity Sensor	13
2.3.4 Conductivity Sensor.....	14
2.3.5 Dissolved Oxygen Sensor.....	15
2.3.6 Total Dissolved Solids (TDS) Sensor.....	16
2.3.7 Free Chlorine Sensor	16
2.3.8 Water Level Sensor.....	17
2.4 Related Research	18
2.4.1 Arduino as a Processing Platform.....	18
2.4.2 Raspberry Pi as Processing Platform.....	26
2.4.3 TI CC3200 as Processing Platform.....	30
2.4.3.1 FPGA as Processing Platform	33
2.4.4 Other Processing Platforms	34
2.4.5 Summary of Related Works.....	38
2.5 Proposed FPGA-Based SWQS	43
2.6 Understanding The FPGA-SoC Heterogeneous system	44
2.6.1 FPGA Chip	44
2.6.2 System-on-Chip	45

2.6.3 Heterogeneous Platform	45
2.6.4 Features of FPGA-SoC Heterogeneous Platform	46
2.6.5 Memory Management on FPGA-SoC	47
2.6.6 Quartus II Development Software	48
2.6.7 Quartus Intellectual Property Libraries	49
2.6.8 Linux Terminal Interface	51
2.6.8.1 Linux Kernel Compilation.....	51
2.7 Summary.....	55
CHAPTER THREE: METHODOLOGY	56
3.1 Overview.....	56
3.2 Research Methodology	57
3.3 Proposed Design Development and System Requirement	60
3.3.1 Smart Water Quality System (SWQS)	60
3.3.1.1 Interfacing Multiple Sensors	63
3.4 SWQS Hardware Design	64
3.5 SWQS Software Design	65
3.6 Proposed Design Power Cycle	68
3.7 Proposed Design Testing Plan	69
3.8 The Design Cycle of The Proposed FPGA-Based SWQS.....	70
3.8.1 The Hardware Design Flow of SWQS	70
3.8.1.1 Processors	71
3.8.1.2 Memory	73
3.8.1.3 Clocking and Synchronization	73
3.8.1.4 Data Transfer Protocols.....	74
3.8.1.5 System Peripherals	75
3.8.1.6 Bridges.....	78
3.8.1.7 Pin Assignment.....	79
3.8.1.8 Synthesis Report.....	80
3.8.2 The Software Design Flow of SWQS.....	81
3.8.2.1 Firmware Development Flow.....	81
3.8.2.2 The Linux Application Development Flow of SWQS	83
3.9 The Embedded Linux Design Flow of SWQS	86
3.9.1 Bootloader Compilation.....	86
3.9.2 Root File-system Creation	87
3.10 Adding New Core to SWQS Design	87
3.11 Summary.....	88
CHAPTER FOUR: RESULT AND ANALYSIS	89
4.1 Introduction.....	89
4.2 Results	89
4.2.1 The Results of the Quartus Project Compilation	89
4.2.2 The Implementation of Prototype.....	90
4.2.3 Design Verification Method	92
4.2.4 The Results of the SWQS Linux-Based Application	98
4.3 Discussion.....	100
4.3.1 Motivations of the Middleware Layer	100
4.3.2 System Flexibility.....	101
4.3.3 Applications of the Design	102

4.4 SWQS Design Compared to Previous Work	102
4.5 Summary	103
CHAPTER FIVE: CONCLUSION AND FUTURE WORK.....	104
5.1 Conclusion	104
5.2 Future Work.....	105
REFERENCES.....	106

LIST OF TABLES

Table 2.1	Summary Table	38
Table 2.2	Summary of All Sensors Used in Each Study Given in Table 2.1	43
Table 2.3	Files Generated After the Yocto Compilation	54

LIST OF FIGURES

Figure 1.1	Hardware Design Flow of FPGA (Intel, 2020)	3
Figure 2.1	Conventional-Based Water Quality Monitoring Example (Tucsonaz, 2015)	10
Figure 2.2	IoT-Based Water Quality Monitoring Example Using Arduino (Pinterest, 2017)	11
Figure 2.3	Configuration of pH Probe (Aaruththiran, Yujia, & Bagherian, 2019)	12
Figure 2.4	Configuration of Temperature Probe (Dhaker, 2020)	13
Figure 2.5	Turbidity Sensor	14
Figure 2.6	Electrical Conductivity Sensor	15
Figure 2.7	Dissolved Oxygen Optical Sensor (Staff, 2020)	15
Figure 2.8	Total Dissolved Solids Sensor	16
Figure 2.9	Free Chlorine Sensor (Karak et al., 2012).	17
Figure 2.10	Ultrasonic Level Sensor for Liquids	18
Figure 2.11	Overall Water Quality Monitoring System (Ngom et al., 2019)	19
Figure 2.12	Prototype of Water Quality Monitoring System (Li et al., 2018)	20
Figure 2.13	Overall Water Quality Scheme (Chowdury et al., 2019)	21
Figure 2.14	Overall System Architecture (Lezzar et al., 2020)	22
Figure 2.15	Hardware Implementation for Water Quality Monitoring Device in Pipeline (Saravanan et al., 2018)	23
Figure 2.16	Experimental Setup of SQWM System (Mukta, Islam, Barman, Reza, & Khan, 2019)	24
Figure 2.17	Circuit and Block Diagram of IoT System (Pujar et al., 2020)	25
Figure 2.18	System General Blocks and Data Flow (Encinas et al., 2017)	26
Figure 2.19	Block Diagram of Developed (Khatri et al., 2020)	27
Figure 2.20	IoT-based Monitoring System (Niswar et al., 2018)	28
Figure 2.21	System Architecture (Raju & Varma, 2017)	29
Figure 2.22	Overall Block Diagram (Vijayakumar & Ramya, 2015)	30
Figure 2.23	Overall Block Diagram (Geetha & Gouthami, 2016)	32
Figure 2.24	Overall Block Diagram (Billah et al., 2019)	33

Figure 2.25	The Block Diagram of Smart Water Quality Monitoring System (Myint et al., 2017)	34
Figure 2.26	Block Diagram of Proposed System (Birje et al., 2016)	35
Figure 2.27	(a) Module 1: The Measurement and Sensing Module Block Diagram (b) Module 2: The Notification Module Block Diagram (Cloete et al., 2016)	36
Figure 2.28	Architecture of the E-Sensor AQUA System (Danh et al., 2020)	37
Figure 2.29	Cyclone V SoC FPGA from Intel (Intel PSG Website, 2020).	45
Figure 2.30	Example of a Memory Model of the Proposed SWQS Design	48
Figure 3.1	Research Methodology Phases	57
Figure 3.2	Research Methodology of the Proposed SWQS	58
Figure 3.3	Overall methodology of the Proposed SWQS	60
Figure 3.4	Proposed System Blocks and Data Flow	61
Figure 3.5	Multi-Core Heterogeneous System Architecture Design of the Proposed Design	62
Figure 3.6	FPGA Single Core Processing Element of the Proposed Design	63
Figure 3.7	Proposed SWQS Hardware Proposed Data Acquisition Design	64
Figure 3.8	The Summary of the FPGA Hardware Design Flow	65
Figure 3.9	Firmware Applications for Each Core with SWQS Linux-based Application	65
Figure 3.10	Flow Diagram of Firmware 0	66
Figure 3.11	The Sensor's Core Firmware Flow Diagram	67
Figure 3.12	The Proposed Design Abstraction Layers	68
Figure 3.13	The Architecture of the Proposed SWQS	71
Figure 3.14	The Configuration of the ARM Processor	72
Figure 3.15	Nios II Processor Configuration	72
Figure 3.16	The Configuration of On-chip-Memory Controller	73
Figure 3.17	Clock source IP in Platform Designer	73
Figure 3.18	PPL Configuration in Platform Designer	74
Figure 3.19	ADC Input Signals	75
Figure 3.20	System Timer IP Configurations in Platform Designer	75
Figure 3.21	mSGDMA Controller IP Configurations	76
Figure 3.22	Mutex configuration in Platform Designer	76
Figure 3.23	The Configuration of JTAG IP in Platform Designer	77

Figure 3.24	The Configuration of System ID Core in Platform Designer	77
Figure 3.25	Avalon Memory Mapped Controller IP	78
Figure 3.26	Address Span Extender IP	79
Figure 3.27	Top View of Pin Assignment	80
Figure 3.28	Synthesis Report of the Design	81
Figure 3.29	Nios II Software Development Flow	82
Figure 3.30	The Entire FPGA Development Flow	83
Figure 3.31	SWQS Application Memory Layout	84
Figure 3.32	FPGA Components Base Addresses	85
Figure 3.33	U-Boot Development Flow	87
Figure 3.34	Nios II Core Connections	87
Figure 4.1	The Result of the System Compilation	90
Figure 4.2	System Setup Components	90
Figure 4.3	Design Block Diagram	91
Figure 4.4	Core 1 pH Data Collection and Data Transfer Test Block Diagram	93
Figure 4.5	Readings of pH Sensor	93
Figure 4.6	The Block Diagram of Core 2 TDS Data Collection and Data Transfer Test	94
Figure 4.7	Readings of TDS Sensor	94
Figure 4.8	The Block Diagram of Core 3 Turbidity Data Collection and Data Transfer Test	95
Figure 4.9	Readings of Turbidity Sensor	95
Figure 4.10	Line Graph of pH Values of Pure Water, Lemon Juice, and Milk	96
Figure 4.11	Testing the pH Sensor	96
Figure 4.12	Line graph of TDS Values of Pure Water and Water with Salt	97
Figure 4.13	Testing of the TDS Sensor	97
Figure 4.14	Line graph of Turbidity Values of Pure Water, Water with Little Dust, and Water with More Dust	98
Figure 4.15	Testing the Turbidity Sensor	98
Figure 4.16	SWQS Results on Linux-Based Application	99
Figure 4.17	SWQS Proposed Single Design Core	101

CHAPTER ONE

INTRODUCTION

1.1 OVERVIEW

Water quality is an important factor that needs to be considered as it is directly related to people's lives. Therefore, this research focuses on designing a Smart Water Quality System (SWQS) Field Programmable Gate Array (FPGA)-based to eliminate the problems and drawbacks of previous works. This chapter illustrates the work by giving a brief background and defining the research problem. The chapter then presents the research motivation. Finally, it emphasizes the research scope.

Section 1.3 defines the research problem statement. Then, in Sections 1.4 and 1.5, the research objectives and research scope are presented, respectively. In addition, an outline of the main structure of the thesis is briefly reported in Section 1.6. Finally, Section 1.7 summarizes Chapter 1.

1.2 RESEARCH BACKGROUND

One of the major causes of death is consuming contaminated or polluted water. According to the World Health Organization (WHO), in 2017, 2.2 billion people were drinking water without any safety management services, and 144 million collected water from untreated water bodies, such as lakes, streams, and rivers (“2.1 Billion People Lack Safe Drinking Water at Home, More than Twice as Many Lack Safe Sanitation,” 2017). To reduce the death rate due to contaminated and polluted water, measuring water quality, especially for consumption, becomes important. Water quality indicators can mean differently. It shows the suitability of any water body used for different uses, such as drinking, cooking, and cleaning. Water usage has different chemical, biological, and physical acceptance levels. For instance, drinking water has specified water quality parameters, such as pH levels ranging from 6.5 to 8.5. As many parameters need to be measured, thus, several sensors must be employed for the water quality test. To this date, the water safety management services that utilize a system that

support large computational loads need a huge amount of power, not portable, and big devices, making it impossible to be commercialized.

Hence, it is pertinent to establish a new smart system considering the latest technological advancement that can carry out the huge computational load at lower power but with high performance. Therefore, this research proposes a water quality system that utilizes the FPGA platform and the Advanced RISC Machine (ARM) processor.

1.2.1 Field Programmable Gate Array (FPGA)

FPGA is a reconfigurable computing device with several programmable units that could solve any computational issues (Giesemann, Paya-Vaya, Blume, Limmer, & Ritter, 2014). This device is an integrated circuit made of semiconductor material, and the main feature of FPGA is the device's electrical functionality can be reconfigured even by the customer. As a result, these powerful devices can be customized to accelerate key workloads and enable design engineers to adapt to emerging standards or changing requirements.

Figure 1.1 illustrates the common design flow for an FPGA platform, beginning with hardware design specifications. The hardware design specifications consist of the needed hardware design's functionality, memory size, the number of input/output ports, speed, and finally, how the data transfers. The next step is the architecture design, in which the hardware design can be further split into system and sub-system modules, i.e., the micro-architecture level design (Gerstlauer et al., 2009).

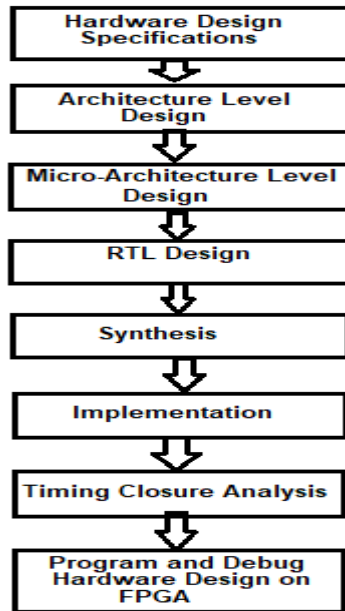


Figure 1.1 Hardware Design Flow of FPGA (Intel, 2020)

Once the architecture and the micro-architecture level of the needed hardware design are completed, the Register Transfer Level (RTL) will begin. At this level, Hardware Description Language (HDL) will be used to translate the system and sub-system blocks into a Hardware netlist (“AN 311: Standard Cell ASIC to FPGA Design Methodology and Guidelines,” 2009). Synthesis and implementation processes will begin when the digital module design is done. These two steps will translate the HDL design into a physical netlist prepared for timing analysis. Timing analysis is the process of ensuring the hardware design is working from a time perspective. In other words, it will check whether the design is the speed requirements of the system or not (Gerstlauer et al., 2009). In the end, the hardware design will be executed on an FPGA board.

1.2.2 Heterogeneous System Architecture (HSA)

A Heterogeneous System Architecture (HSA) is a computer platform that functions with associated software that makes different kinds of processors with different architectures work in shared memory efficiently and cooperatively from a single source program (Kyriazis, 2012). Integrating multiple computing elements at low frequencies leads to high performance with low power consumption; architectural heterogeneity improves platform flexibility (Burgio et al., 2016). This heterogeneous platform, such as FPGA-System-on-Chip (SoC), improves the performance of embedded using

hardware containing more than one type of processor. This approach has shown improved performance, particularly in artificial intelligence (AI), in which computationally demanding models must be trained and executed.

SWQS utilizes different sensors to measure water parameters such as pH, turbidity, and Total Dissolved Solids (TDS) and then processes the data on FPGA-SoC. The proposed design methodology reduces the complexity of the FPGA-SoC heterogeneous platform by adding a middleware layer for software developers to interact with the FPGA system in the form of an application program interface. Therefore, heterogeneous architecture is the best choice for complex systems with multiple input and output ports to enhance the overall system performance.

1.3 PROBLEM STATEMENT

Currently, in Malaysia, water quality monitoring is done by traditional methods, consisting of taking samples from the area under test and then driving them back to a laboratory to analyze them. The analysis usually is for detecting chemicals and microbial that cause the water's pollution. This method is not only time-consuming but requires significant human interaction. As a result, important data may be lost because of the manual collection process. In addition, the water quality analysis is not done within a short time, so determining a real-time water condition is not plausible. This traditional method is only good if the samples are taken and analyzed simultaneously (Geetha & Gouthami, 2016). Moreover, the water might get contaminated, making it very difficult and costly to recover (Billah, Yusof, Kadir, Ali, & Ahmad, 2019). Moreover, the technicians cannot take samples from all locations, which may lead to inaccurate data (Lezzar, Benmerzoug, & Kitouni, 2020). Besides the issues arising from the manual sample collection, the chemical materials used in water quality testing are usually toxic and very expensive (Khatri, Gupta, & Gupta, 2020).

Even though the research on water quality monitoring systems have been applied many times, the current system is still expensive, has short-distance data transmission, and is not easy to use (Geetha & Gouthami, 2016). Most current SWQS is costly (Pasika & Gandla, 2020), and there should be a big effort by researchers to reduce the cost to make the system more affordable for everyone. Performance is also an important issue

that needs to be considered. SWQS must have high performance and accuracy to reduce errors that might cause poor health conditions and death for the people who consumed the water if the measurements are incorrect.

While FPGA has high processing power, developing an FPGA can be complex and requires more effort than configuring the same design on the Central Processing Unit (CPU) (Besta, Stanojevic, Licht, Ben-Nun, & Hoefler, 2019). In addition, it provides less specialized components (i.e., floating point) operations. It is for this reason that FPGA remains to be a prototype platform for embedded systems. That said, the use of heterogeneous platforms mesh with FPGA has recently gained popularity for design applications that need performance and programmability offered via a processor and flexibility and configurability accomplished using the FPGA fabric (Zhong, Niar, Prakash, & Mitra, 2016). The SoC heterogeneous platforms improve the performance of embedded systems using a hardware design that contains more than one processor.

In addition, when comparing FPGA with other processors such as Arduino and Raspberry Pi, in terms of configurability and implementation, FPGA is reconfigurable based on the user's requirements. However, Arduino and Raspberry Pi are configured and implemented during manufacturing. Additionally, FPGA can process the data in parallel to overcome the latency issue when many inputs are used. On the other hand, there is no way to perform pipelines using processors such as Arduino and Raspberry Pi. In addition, the processing rate of the SWQ data using an FPGA processor is high as its frequency reaches 1 GHz. However, the frequency is slightly lower in other processors, for instance, 16 MHz and 400 MHz for Arduino and Raspberry Pi, respectively. Last but not least, the pins of FPGA are 40 pins that are not hardcoded as their interface can be modified based on the sensor's data exchange protocol, unlike the pins of Arduino and Raspberry Pi, which are hardcoded during the manufacturing process. Moreover, FPGA-SoC is a heterogeneous platform that can work with shared memory for more cooperativity and efficiency. In addition, a heterogeneous platform such as FPGA-SoC improves the performance of embedded using more than one processor.

For these reasons, the main objective of this project is to design SWQS using an FPGA-SoC platform to monitor different water parameters, namely; pH, TDS, and turbidity parameters, rather than relying on the conventional way of measuring water quality parameters.

1.4 RESEARCH OBJECTIVES

The main objectives of the project are as below:

- i. To design a reconfigurable hardware-based Smart Water Quality System (SWQS) via using a Field Programmable Gate Array-System-on-Chip (FPGA-SoC) heterogeneous platform.
- ii. To implement a real-time prototype for the proposed Smart Water Quality System (SWQS).
- iii. To evaluate the proposed Smart Water Quality System (SWQS) based on pH, Total Dissolved Solids (TDS), and turbidity parameters.

1.5 RESEARCH SCOPE

The scope of this research mainly concentrates only on the hardware design of SWQS by utilizing the heterogeneous platform of FPGA-SoC to process signals obtained from water quality sensors. The system design integrates FPGA with the SoC to create a customizable heterogeneous platform that segments the system functionality into tasks. The proposed design in this study will utilize two development kits, the DE10 Nano FPGA-SoC development kit from Intel and the Raspberry Pi development board. The utilized boards will not impact the proposed system design since the design flow is the same for any FPGA development board. The system has two SoC sub-systems: an external one (Raspberry Pi) and an internal one (ARM SoC). The external sub-system will provide the system with all the required augments to ease prototype implementation, like LCD, mouse, and keyboard. The internal SoC will be part of the SWQS as the main system processor.

In addition, testing the proposed system will only be based on three water quality parameters to verify and validate the functionality and reliability of the proposed FPGA-SoC platforms. The utilized sensors in this design will be the pH, TDS, and turbidity sensors. These sensors were used as a proof-of-concept to validate the system's functionality. However, other sensors related to SWQS can be adapted to any future system based on the user's requirements.

1.6 DISSERTATION LAYOUT

This dissertation is composed of five chapters; a brief introduction and overview of the research are provided in Chapter 1. In Chapter 2, an in-depth investigation was conducted about the previous studies in the SWQS development field. Chapter 3 elaborates on the proposed system, and the design steps needed to develop an SWQS hardware design based on a heterogeneous platform are presented as well as presenting the flow of software to program the system. Furthermore, the proposed system test results and the collected data, were discussed in Chapter 4. Finally, in Chapter 5, the summary of the research findings, contribution, claims, and comparative analysis was reported.

1.7 SUMMARY

This chapter presented a detailed overview of the research topic, known as SWQS. First, the problem statement of this study was illustrated. Then, the research objectives are presented in this chapter. Furthermore, the scope was explained. Finally, the thesis layout and the relation between each chapter were discussed.

CHAPTER TWO

LITERATURE REVIEW

2.1 OVERVIEW

This chapter describes the academic literature correlated with Smart Water Quality System (SWQS) hardware implementation. The main objective of this chapter is to find out and elaborate on the latest research achievements in the field of SWQS design and development. In addition, this chapter highlighted the drawbacks and problems encountered by researchers in their designs, as well as obstacles in providing suitable SWQS solutions.

In Section 2.2, a brief overview of water quality approaches is presented. Then, Section 2.3 explains some sensors that measure water quality parameters such as pH, temperature, turbidity, electrical conductivity (EC), and dissolved oxygen (DO). In Section 2.4, previous studies have been presented based on the controller or the processor. They have been used to collect data on water quality parameters and summarize the related studies' motivations and drawbacks. Section 2.5, on the other hand, shows the proposed SWQS FPGA-based followed by Section 2.6. It presents general information about FPGA in terms of architecture, such as memory, speed, interfaces, etc., and software tools, such as Quartus, Platform Designer, etc. Finally, Section 2.7 summarizes this chapter.

2.2 DIFFERENT WATER QUALITY APPROACHES

There are two ways of measuring water quality. The first is the traditional method involving samples from the river, lake, or any water source, while the second uses sensors to measure water quality. In the following subsections, both conventional and Internet of Things (IoT) methods will be discussed in detail.

2.2.1 Conventional Approach

The traditional way of measuring water quality parameters, such as the water pH, turbidity, DO, and EC, starts with several samples for testing. Note that sampling selects a small portion of the water to be handled and transported to the laboratory (Ngom, Diallo, Gueye, & Marilleau, 2019). After transporting the samples to the laboratory, specific materials or solutions must be added to measure a specific parameter. For example, to measure the level of phosphorus, one of the crucial water parameters, samples must be transferred to the lab as soon as possible to minimize any external effects that might change the measurement of the total phosphorus. Potassium persulfate should be mixed with the water sample before heating it for 30 minutes. After the heating process, the mixture must be cooled to room temperature. Before measuring the total phosphorus, sodium hydroxide is added and mixed with the sample gently. The last step is to measure the total phosphorus using a spectrophotometer device 7 minutes after mixing. Sometimes, the process can take more than five working days (Li, Jaafar, & Ramli, 2018).

Besides the elaborated identification process, the sampling process is not easy as the samples must be taken from the specified location and involve a highly complex process. Additionally, water samples should be transferred to the laboratory and tested as soon as possible to avoid water pollution. Moreover, this method is time-consuming and costly, requiring equipment cleaning, measuring procedures, and recording. Finally, due to human interaction, many errors might occur during the process, and that will affect the accuracy of the reading.

To conclude, this method is inefficient, and more research should be conducted to develop alternative methods to avoid the challenges mentioned above. Figure 2.1 shows an example of measuring water quality using the traditional method.

TESTING FOR WATER QUALITY

Tucson Water performs 14,000+ water quality tests a year



Figure 2.1 Conventional-Based Water Quality Monitoring Example (Tucsonaz, 2015)

2.2.2 Internet of Things (IoT) Approach

With the advancement in technology, many alternative methods have been designed and developed to measure water quality parameters incorporating the IoT and machine learning to solve the problems of water quality assessment (Geetha & Gouthami, 2016). The IoT involves using a smart sensor connected to a processor to process the data for monitoring. It can also be connected to a communication tool such as Wi-Fi, LoRa, and Bluetooth to send the data remotely to the main station for real-time monitoring. This way, data can be sent from over a few meters to thousands of meters away, depending on the communication tool's capacity.

This method costs less money when compared to the conventional method, as the sensors and controller are comparatively cheap. More importantly, it provides real-time measurement and monitoring to help the user detects any changes immediately.

That said, SWQS needs more exploration since the sensors are expensive with good quality or cheap with bad quality. Furthermore, the sensors need regular maintenance to avoid damage, as the sensors can be poorly affected by the materials in the liquid. In addition, some communication tools can be expensive, and some can only send data at a lower range, making them unsuitable for big water sources. Figure 2.2 shows an example of measuring water quality using the IoT method.

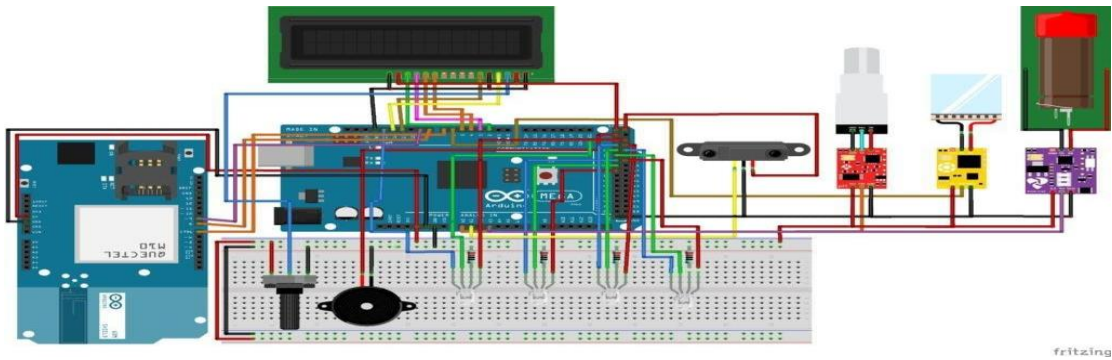


Figure 2.2 IoT-Based Water Quality Monitoring Example Using Arduino (Pinterest, 2017)

2.3 SENSORS USED IN MONITORING WATER QUALITY PARAMETERS

Sensors convert the physical parameter into equivalent measurable electrical quantity, which is given as input to controllers through an optional wireless communication device and parameters.

2.3.1 pH Sensor

The pH represents the number of hydrogen ions in the water. It can be calculated using the negative base of 10 logarithms of the hydrogen ions per liter (Sensorland, 2019). For most water sources, the pH value should be from 6 to 9; if it is more or less, most sea creatures, such as mussels and clams, would be affected negatively (Kraxner, 2015).

pH probe, as shown in Figure 2.3, contains inner and outer tubes. The outer tube contains a Potassium Chloride (KCl) solution, which considers the controller, while the inner tube has a buffer solution with pH 7. The bottom of the inner tube is usually made of a glass overlay that allows the hydrogen ions to move between the probe and test solution. The inner and outer tubes have a silver wire covered with silver chloride connected to the sensor amplifier circuit. Once the probe is placed in the water, the glass membrane will allow the hydrogen ions to go through it and replace the wire's ions which will allow current to flow, causing EC. Figure 2.3 shows the configuration of the pH probe. The voltage values will vary according to pH value; for example, 0 to 0.4 V is equivalent to pH 7 to pH 0, and so on. The pH value will then be obtained when the microprocessor processes the value. If the water or the solution pH is less than 7, this

means a high concentration of hydrogen ions, implying it is an acidic solution. Once the pH value reaches more than 7, the solution is a base. As the pH values might be minimal and unreadable, an amplifier circuit must be added to obtain clear pH values (“PH Meter,” 2022).

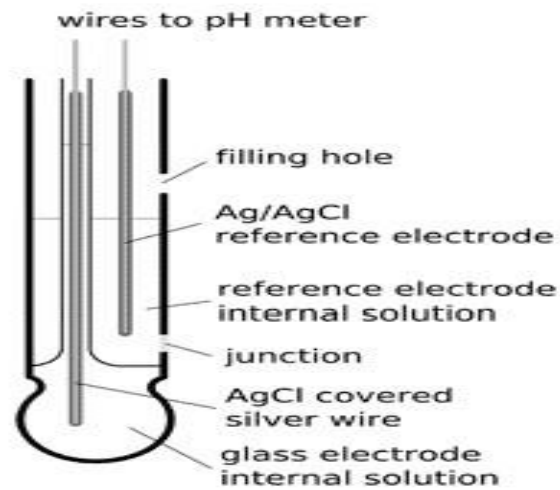


Figure 2.3 Configuration of pH Probe (Aaruththiran, Yujia, & Bagherian, 2019)

2.3.2 Temperature Sensor

The temperature sensor is made of a probe and built-in Integrated Circuit (IC) containing a register and alarm that will show a warning when the temperature is high or not in the range. A pull-up resistor is used so the microcontroller will reduce the resistance when the sensor sends the data. This sensor can function without additional power as it has a capacitor that stores energy from high signals. However, an external power source is needed to prevent any outbreak of the sensor. The temperature sensor is essential as it affects the conductivity and turbidity as the ions move in the water when the temperature is high (Thermometer, 2019). Figure 2.4 shows the configuration of the temperature probe.

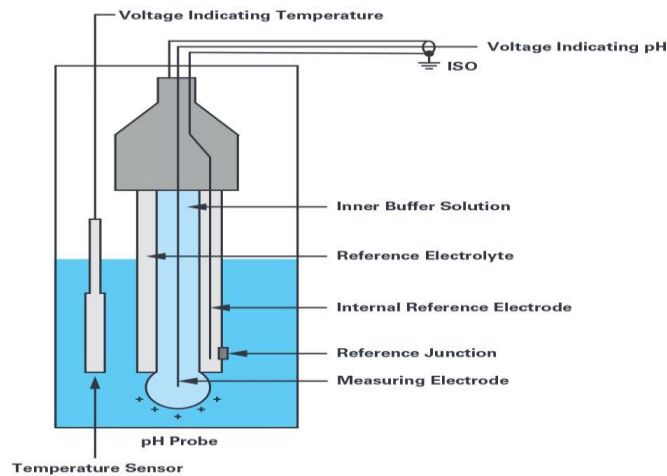


Figure 2.4 Configuration of Temperature Probe (Dhaker, 2020)

2.3.3 Turbidity Sensor

Turbidity is the measuring of water clarity and cloudiness. It indicates whether the water has suspended particles or not. It is an important factor that should be measured in most water treatment and supply stations as it might cause harm to aquatic life, not to mention human health.

The concept of a turbidity sensor is a light-transmitting and scattering rate that relies on the total amount of particles in the water. If the light is sent to the water and scattered, the water has many particles that cause the light to scatter. On the other hand, if the phototransistor receives the light, then the water is clear. A phototransistor is connected to a resistor; when the voltage is high, the water is clear of particles (“Gravity: Analog Turbidity Sensor For Arduino,” n.d.). Figure 2.5 shows a turbidity sensor.



Figure 2.5 Turbidity Sensor

2.3.4 Conductivity Sensor

The ability of water to allow current through it is known as EC. Current relies on the number of electrons that conduct electricity. The parameters that can affect the conductivity are the solution temperature and ions concentration, as well as the EC parameter, which can affect other parameters such as DO (Acmasindia, 2019). In the conductivity sensor, two electrodes are placed against each other. Once a current is applied to the outer pair, the inner pair potential can be measured. As a result, the current will switch charge electrodes, finally leading the ions to pull the oppositely charged electrodes. As a result, the electrode will carry many charges proportional to the water conductivity (“EC/TDS/PPM Meter On Limited Budget,” 2008). An EC sensor is used in various applications such as cooling tower water treatment, boiler water treatment and reverse osmosis monitoring. The sensor selection is based on the required application to ensure lifetime and accuracy. It is measured in “mS” (milliSiemens) or “ μ S” (microSiemens) per centimeter (“Gravity: Analog Turbidity Sensor For Arduino,” n.d.). Depreciating or appreciating water conductivity may indicate that the water is polluted, as some materials might increase conductivity, such as chloride, nitrite, and phosphate ions (O’Donnell, 2017). Not to mention, the conductivity can be proportionally increased when the temperature is high. However, the conductivity sensor suffers from some practical drawbacks. For example, the user cannot determine what ions are dissolved in the water, and the sensor should be cleaned properly before using it again in another solution. Figure 2.6 shows an EC sensor.



Figure 2.6 Electrical Conductivity Sensor

2.3.5 Dissolved Oxygen Sensor

DO is an instrumental water parameter that is measured in mg/l. Electrolysis and optics are the methods that are used to measure DO. However, the optic method is better in terms of accuracy and time. The DO sensor consists of a photodetector, two blue and red LEDs, and a luminescent dye between the LEDs, as shown below in Figure 2.7. Once the blue light reaches the dye, the electrons will get energy and let the light emit until it becomes stable. On the other hand, the photodetector will receive the light and destroy it if the solution contains oxygen because there is an interaction between the dye and the oxygen.

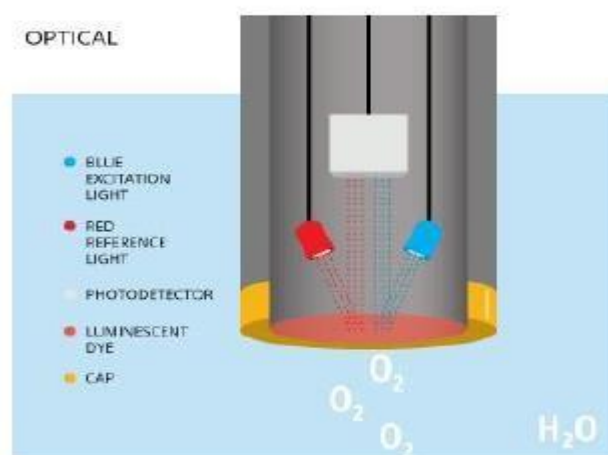


Figure 2.7 Dissolved Oxygen Optical Sensor (Staff, 2020)

2.3.6 Total Dissolved Solids (TDS) Sensor

It is a smart sensor that measures the total dissolved materials in a liquid such as water. The Total Dissolved Solids (TDS) sensor measures the solution conductivity as solids that are ionized in liquid-like minerals and salt raise the conductivity of the liquids. Therefore, the TDS sensor mainly measures conductivity; the TDS value could be estimated from these readings. The unit that measured the values is Parts Per Million (PPM). In addition, the sensor's cost starts at \$10 only, but with limited features. For instance, a basic one can measure only the TDS of a liquid. However, expensive one can measure the temperature, salinity, and even more.

It is essential to measure the TDS value, as it indicates that the water has many dissolved solids when the value is high. As time pass, these materials might cause damage and decrease the devices' and water pipelines' lifetime. Therefore, it is important to find a way to solve this issue, like using a water filter (Aquasana, 2022). Figure 2.8 presents an example of a TDS sensor.



Figure 2.8 Total Dissolved Solids Sensor

2.3.7 Free Chlorine Sensor

Free chlorine is one of the most important water quality parameters that need to be measured to ensure water quality. Free chlorine is known as Residual Chlorine (RC). It shows the indication for the water potability level. It measures the amount of RC, which

exists in the liquid as dissolved gas, such as Chlorine (Cl_2) and hypochlorite ion (ClO^-), which might measure hypochlorous acid (HOCl) as well. Free chlorine sensor could be utilized to measure the total amount of three materials Cl_2 , OCl^- , and HOCl . Note that Mg/L is the unit that is used to measure the free Cl_2 parameter.

Cl_2 is commonly used to disinfect and clean the contaminated water source. Free Cl_2 in the water can be tested using a different kit, such as digital colorimeters or color-wheel test kits. The sensor read indicates that the water containing free Cl_2 is free of contamination. Figure 2.9 shows an example of a free Cl_2 sensor (Karak, Bhagat, & Bhattacharyya, 2012).



Figure 2.9 Free Chlorine Sensor (Karak et al., 2012).

2.3.8 Water Level Sensor

The water level is an essential parameter that needs to be measured in real-time so fast action can be taken before something happens. A water quality sensor is a device that is utilized to measure the low and high levels of water in a calm situation. The water level sensor is a contact sensor that is used to convert the water level, which is an analog read, into an electrical signal that a microcontroller can process. Different kinds of sensors could be used to measure the water level, such as optical water level sensor, magnetic flap level, hydrostatic level transmitter, ultrasonic sensor, etc.

The water level can be measured by placing the sensor into the liquid at the surface, and then the pressure of the liquid will be converted into the height of the liquid using the following equation:

$$P = \rho \cdot g \cdot H + P_0, \quad (2.1)$$

where P is the pressure of water measured at the surface, ρ is a fixed value which is the density of the water, g is the gravity, P_0 is the pressure of the atmosphere at the surface of the liquid, and finally, H is the depth where the sensor was placed. Figure 2.10 shows the ultrasonic sensor, one of the sensors that can be used to measure the water level (“Ultrasonic Level Sensor,” 2022).



Figure 2.10 Ultrasonic Level Sensor for Liquids

2.4 RELATED RESEARCH

2.4.1 Arduino as a Processing Platform

Arduino is an electronic device that utilizes an open-source platform based on easy-to-use hardware and software. Arduino microcontroller could be used to process data from different sensors and devices by sending a set of programming codes to the board. Arduino Software (IDE) is an easy-to-use software that the programmer can use to control the Arduino board. The board has sixteen digital pins and six analog pins, all hard-coded. In addition, Arduino has a built-in clock frequency that reaches up to 16 MHz. Arduino board has been used in many water quality systems with different sensors. The below graphs conclude some of the studies that have used Arduino to design water quality systems (“What Is Arduino?,” 2018).

In (Ngom et al., 2019), the author presents a water quality monitoring system that uses the LoRa transmission system to send the data to be visible on a website. The system used an Arduino Mega 2560 microcontroller to get and process the sensors' data. Water quality was monitored using four sensors: pH sensor, oxidation/reduction potential (ORP) sensor, EC sensor, and water temperature sensor. Moreover, the Arduino microcontroller was used because it was easy to handle due to the hardware and software flexibility. In addition, the LoRa transmission system transmits the signals using industrial, scientific, and medical (ISM) bands. Therefore, there is no need to pay for the local telecommunication operator. The prototype is a low-power consumption system as all sensors may operate using 5 V, and it is recharged using a solar panel. However, this system did not measure other important water parameters such as water level, turbidity, etc. In addition, the LoRa transmission system is the low range, transmitting data within 2 km to 3 km at a high price because it needs a gateway for transmission. The complexity and cost are the main problems in water quality monitoring systems. Figure 2.11 shows the overall water quality monitoring system.

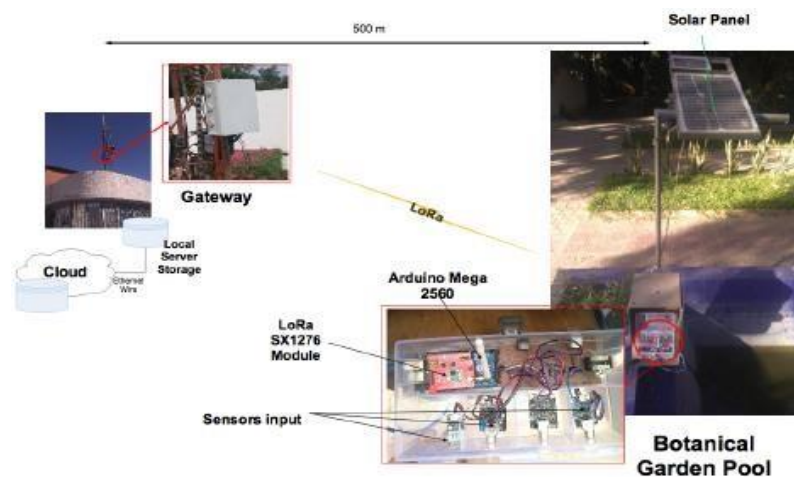


Figure 2.11 Overall Water Quality Monitoring System (Ngom et al., 2019)

Moreover, (Li et al., 2018) have designed a wireless water quality system to measure water quality parameters, send it through Wireless Sensor Network (WSN) technology, and finally display the obtained data on a website platform. Arduino Uno microprocessor is used to process the obtained data from the smart sensors and direct the data into the transmission unit to transmit the data. The pH sensor, temperature sensor, and TDS sensor are connected to the processor. TDS is a water quality parameter

used to show the inorganic salts and small amounts of organic matter present in solution in water. The obtained data is transferred to the IoT platform through a Wi-Fi shield. Then, the data will be stored and displayed on the screen using ThingSpeaker software, providing immediate visualization of data simultaneously to ensure that the transmission is done successfully without any loss. A Wi-Fi shield is used as it can transmit a high amount of data, and the transmission rate of a Wi-Fi shield is high compared to other communication mediums. The pH sensors used in this project can function in temperatures between 0 to 60°C without damage within the same range of temperature needed. In addition, this sensor is used to measure the pH value without delay. Furthermore, the TDS sensor has very high accuracy and is waterproof to be used in water. On the other hand, the above system requires calibration before collecting the data. Therefore, data transmission might delay or even lost with a poor Wi-Fi connection. Figure 2.12 presents the prototype of a water quality monitoring system.

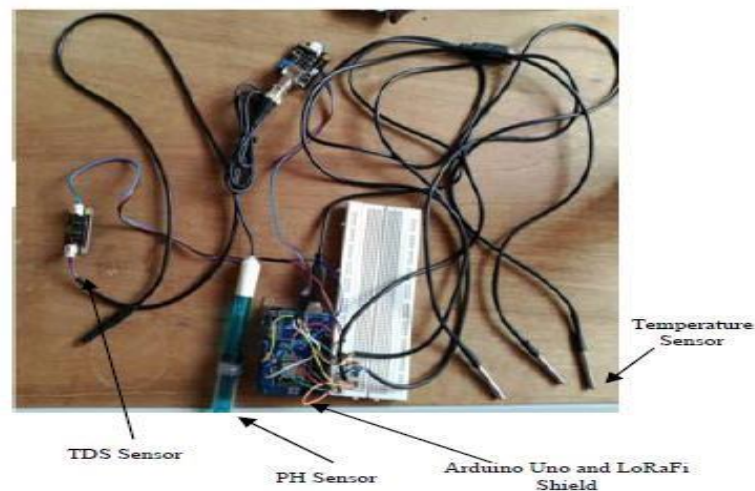


Figure 2.12 Prototype of Water Quality Monitoring System (Li et al., 2018)

Additionally, (Chowdury et al., 2019) have developed SWQS to measure the quality of the water remotely. Conductivity, pH, temperature, and turbidity are the parameters that were measured in this project, as seen in Figure 2.13. Arduino Mega 2560 was used to be the processing unit for the data obtained from the sensors because it has several ports, which make it suitable for many sensors to be connected and then display the results on LCD to monitor the data in real-time. The obtained data is just a number; the user might not understand it, which is why a classification method was

required to classify the data of each sensor into good or bad along with the real-time numbers. Thus, big data analytics was integrated with IoT because of its high speed, reliability, and scalability. IoT application was utilized to help the user get the visualized data on a mobile, laptop, and Personal Computer (PC), easing the visualization. Users can get daily/monthly/ even yearly reports as the system has a data management layer to provide the client with the reports. The designer had utilized an Arduino board limited to the interfaces, such as Inter-Integrated Circuit (I2C) and Serial Peripheral Interface (SPI), even Arduino Mega has many pins, but still, the interfaces cannot be changed as it is hard-coded and cannot be reconfigured.

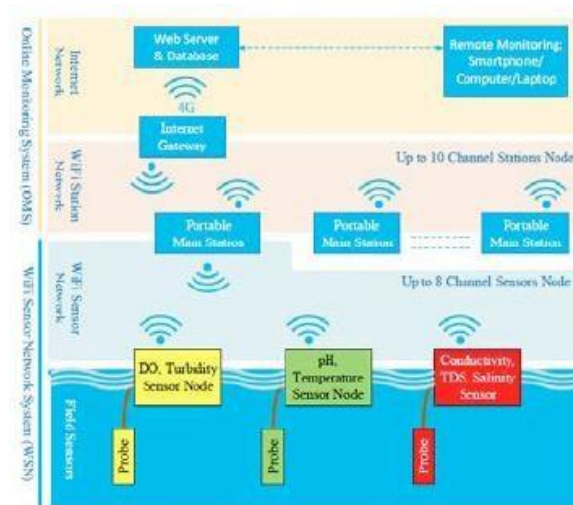


Figure 2.13 Overall Water Quality Scheme (Chowdury et al., 2019)

Moreover, (Lezzar et al., 2020) have designed and developed an IoT system that measures specific water quality parameters to achieve high accuracy. The system has five sensors to measure parameters that indicate if the water is contaminated with pH, temperature, turbidity, ORP, and Cl₂. ORP is an essential environmental water parameter that reflects the clearness of the liquid and could remove the pollutants in ponds. Meanwhile, the Cl₂ sensor measures the Cl₂ concentration in the water, which needs to be kept at a low level to keep drinking water healthy. The Arduino ATmega1281 microcontroller is the main processing unit to which all sensors are connected. The system solves the problem of the power supplies as it is attached to solar cells to recharge the batteries. All obtained data was sent to the user and stakeholders using the Message Queuing Telemetry Transport (MQTT) protocol via the SIM800c

module to achieve quick data transfer. The system can monitor the water quality in a fixed tank or mobile water source. Moreover, it can localize where the water is contaminated exactly. Thus, a decision can be made accordingly. The designed system consists of a cloud infrastructure to get the parameters remotely. It is a lifespan system that was designed to avoid short-term maintenance. SIM800c module is a GSM/GPRS solution that supports different frequency bands instead of Wi-Fi because Wi-Fi is unavailable everywhere. Figure 2.14 presents the overall system architecture.

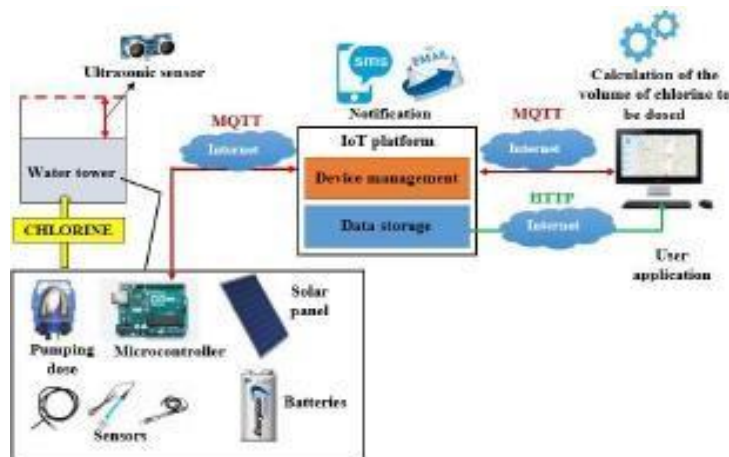


Figure 2.14 Overall System Architecture (Lezzar et al., 2020)

Additionally, (Saravanan, Anusuya, Kumar, & Son, 2018) have proposed a Supervisory Control and Data Acquisition (SCADA) system that cooperates with IoT for real-time water quality monitoring in India. Several sensors measure physical and chemical water parameters such as water temperature, flow, pressure, pH, and color. A color sensor is utilized to discover whether the water is contaminated. When the water is mixed with dust, it will be shown in the RGB value within the range of 0 - 255. GPRS module was connected to Arduino ATmega 368 microcontroller to connect the latter with the internet. The water contamination status was sent to the server through a personal computer or mobile device to be displayed on the website. The data obtained from the sensors were viewed on the web platform, and there was an LCD for viewing data on-site.

Moreover, after the data was obtained from each sensor, it was compared with the threshold values to inform authorized users via SMS of any abnormal sensing to

take action needed as fast as possible. SCADA system was used to get the sensor reading from different stations for real-time monitoring. In addition, the primary advantage of the SCADA system is the report generation available to the operator in each station. The system is designed to make the operation easier, reduce size, weight, and cost, and improve sensitivity. Using the General Packet Radio Services (GPRS) module has many advantages, such as high-speed communication between a mobile device and the main network. Arduino microprocessor is used instead of a Programmable Logic Controller (PLC) controller to accelerate the SCADA system speed. Furthermore, the most exciting systems were standalone devices that were not connected to any IoT platform, while this system utilized IoT to avoid taking the results on-site. However, the proposed SCADA system cannot be implemented in areas not covered by Wi-Fi; hence, the GPRS module constantly needs Wi-Fi. Figure 2.15 shows hardware implementation for water quality monitoring devices in the pipeline.

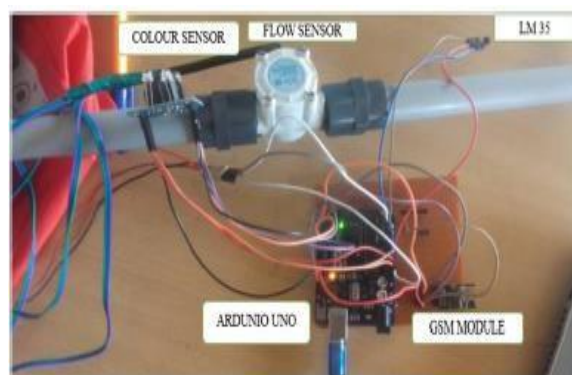


Figure 2.15 Hardware Implementation for Water Quality Monitoring Device in Pipeline (Saravanan et al., 2018)

An IoT system was designed by (Mukta, Islam, Barman, Reza, & Khan, 2019) to measure four different parameters: pH, conductivity, temperature, and turbidity. Arduino Uno is used as a microcontroller to process the data obtained from the sensors. The Arduino microcontroller is connected to a desktop directly, so the obtained data can be displayed on it. An application was developed in the .NET platform to check the obtained data with the World Health Organization (WHO). The sixty-water sample was taken from three sources: drinkable, unclear, and natural. Fast forest binary classifiers are machine learning algorithms for training and testing the module, which has been used to classify whether the test water sample is drinkable or not. In the end, it was

concluded that the fast forest algorithm is the best in terms of accuracy, and the F1 score, which measures the model's accuracy, was used for the overall system performance. In addition, the system is not real-time based, meaning samples need to be collected, which is time-consuming. Apart from that, some water features might be affected when transferring it. Additionally, Figure 2.16 illustrates the experimental setup of the SQWM system.

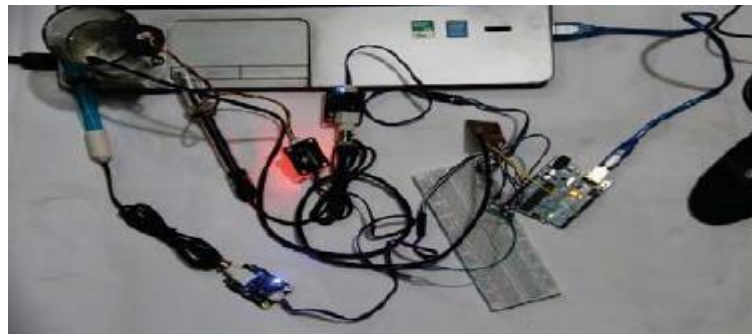


Figure 2.16 Experimental Setup of SQWM System (Mukta, Islam, Barman, Reza, & Khan, 2019)

Furthermore, (Pujar, Kenchannavar, Kulkarni, & Kulkarni, 2020) have developed a real-time water quality system to monitor six water parameters like pH, EC, nitrate, Biochemical Oxygen Demand (BOD), Total Dissolved Oxygen (TDO), and temperature. The sensors are connected to Arduino Mega 2560 controller to process the sensors' data, as shown in Figure 2.17. A total of thirty-six (36) samples were randomly collected from six different stations. Note that the samples were taken in different seasons throughout the year to check whether the weather impacts water quality or not. After data obtaining, it was sent through the ESP8266 Wi-Fi shield to the main station for monitoring. One-way and two-way Analysis of Variance (ANOVA) analysis tools were used to evaluate the system, and it has been found that one-way ANOVA is the most suitable tool for training data, such as an IoT system. Furthermore, this research found that different seasons have different results. In other words, temperature, DO, conductivity, BOD, and nitrate parameters were impacted in the winter.

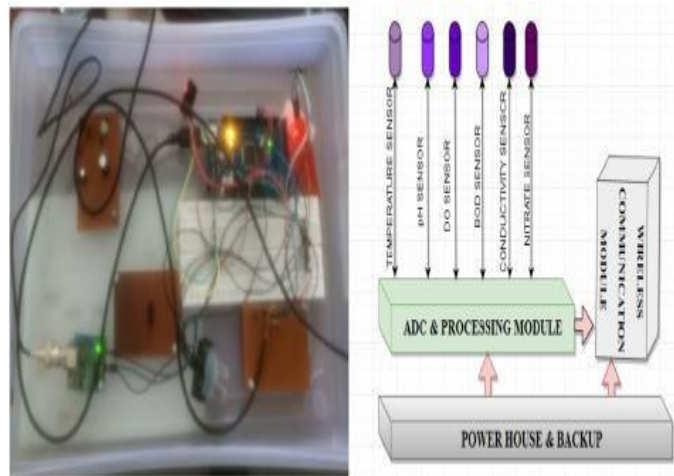


Figure 2.17 Circuit and Block Diagram of IoT System (Pujar et al., 2020)

(Encinas, Ruiz, Cortez, & Espinoza, 2017) proposed a real-time water quality system to monitor three water quality parameters, namely pH, temperature, and DO. Arduino microcontroller was implemented for processing the data into Zigbee wireless communication tool for transmitting the data obtained from the sensor remotely to the main station. An application was developed using the C# programming language, in which its database was designed using the MySQL platform. While the readings were obtained from the sensor, it was directly sent to the database for local storing and sent to a web server to visualize the data on the application. The overall system costs less compared to other designs, and it is portable and has low power consumption.

Additionally, the C# programming language was used because it allows giving a request to the sensors to send their readings through the microcontroller along with a multiplexer. The Zigbee tool will transmit the data obtained from the sensors to the computer for display on the application. A multiplexer is a device that is used to raise the efficiency of the communication system by collecting the data from each sensor and transmitting via a single line only. However, the system was designed to display data without alerting the end user if the water condition was bad. Developing an artificial intelligence (AI) module to alert when the data is not within the standards set in the database has been recommended. Figure 2.18 presents the system's general blocks and data flow.

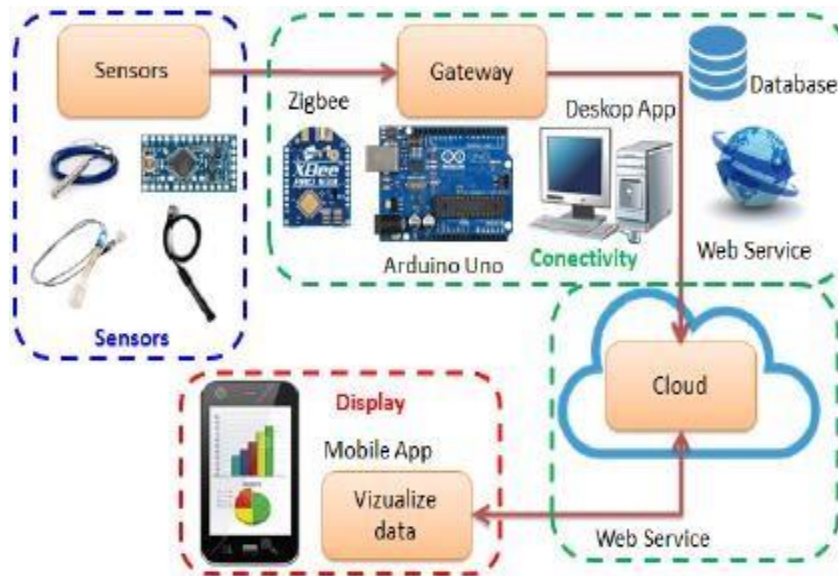


Figure 2.18 System General Blocks and Data Flow (Encinas et al., 2017)

2.4.2 Raspberry Pi as Processing Platform

The Raspberry Pi is another low-cost and small-size controller that could function as a computer along with other peripheral devices, such as a mouse and keyboard. It helps developer to design any device using Python or Scratch programming languages. Note that Raspberry Pi has the capability of being connected to other devices, such as sensors. Since the Raspberry Pi controller has been used in many smart devices to process the sensors' data, it has an inbuilt Wi-Fi module for remote access. Thus, there is no need for external equipment. In addition, the Raspberry Pi controller can function on 12 V only; thus, it can be implemented easily. Many water quality systems have utilized the Raspberry Pi controller to collect the data from water quality sensors and send it through different transmission devices for monitoring (“What Is a Raspberry Pi?,” 2015). The below studies show the main advantages and disadvantages of using Raspberry Pi in water quality designs.

(Khatri et al., 2020) invented a real-time water quality system in India that measures water quality parameters using different sensors, such as pH sensor, EC sensor, ORP sensor, DO sensor, and temperature sensor, as mentioned in Figure 2.19. ORP is a water quality measurement that indicates if the water is oxidizing or reducing. ORP parameter should be measured when its value is low, indicating that the water has less DO. It then leads to an increase in the toxicity of certain materials, which increases

water contamination. The obtained data from the sensor was processed using a Raspberry Pi 3 controller, which collects data, analyzes it, and makes decisions since the controller is programmed using python. Finally, the system's performance was validated by comparing the results with another work using Absolute Percentage Relative Error (APRE). Raspberry Pi has an inbuilt Wi-Fi module for remote access. Thus, there is no need for external equipment. In addition, the Raspberry Pi controller can function on 12 V only; therefore, it can be implemented easily. The system has very high accuracy in measuring the parameters compared to other work. However, Linux operating systems must be used when implementing a system using Raspberry Pi, which is not popular among users. In the future, it has been recommended to implement a water quality monitoring system using fuzzy logic in the IoT environment and distribution networks.

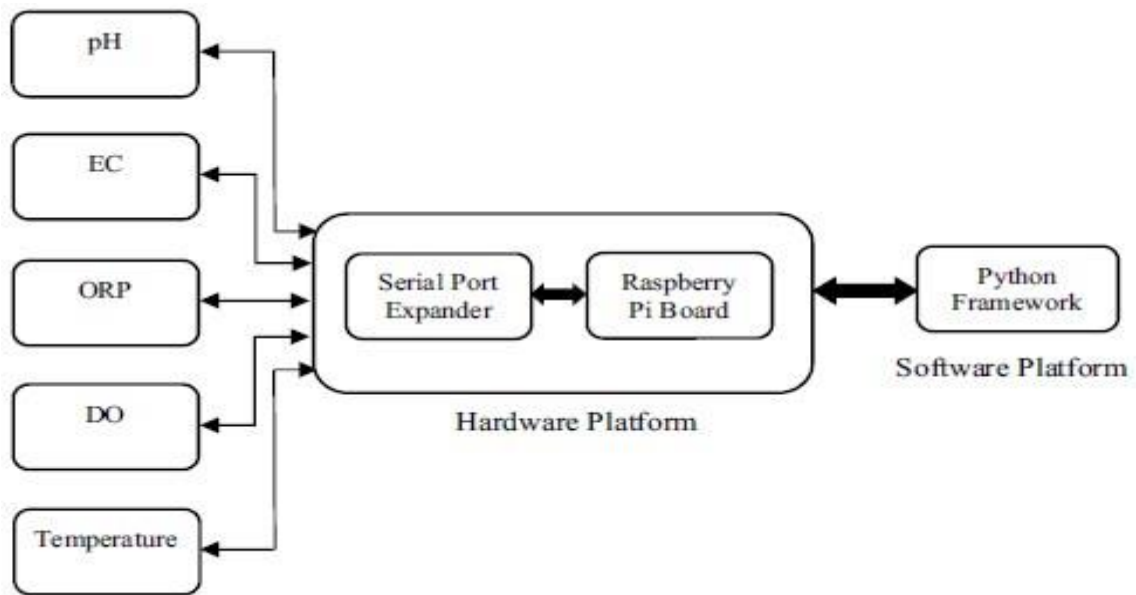


Figure 2.19 Block Diagram of Developed (Khatri et al., 2020)

(Niswar et al., 2018) proposed a water quality monitoring system using an IoT platform to measure water quality for crab farming in Indonesia. The system has three sensors to measure pH level, temperature, and salinity. The sensors are connected to Raspberry Pi and Arduino processors for processing the data into the mobile phone using the MQTT protocol. LoRa wireless communication system is the middle way to transfer data long distances. The user has designed a web-based application for the

mobile phone to remotely monitor water quality using the node-red platform. This open-source programming tool can be used to communicate with IoT devices. The system was designed to alert the crab farmer if the farm water quality is not acceptable for all water parameters. The system is essential for crab farmers to monitor the water quality parameters as water quality can affect their survival. However, the system can measure only pH, salinity, and temperature and neglects other parameters such as DO, which might have a bad impact if it is not within the acceptable level. In addition, the LoRa system and the sensors are connected to the Arduino microprocessor and Raspberry Pi, making it complicated. Furthermore, the data was transferred using 3G and 4G networks, which are not available everywhere, which might delay data transfer or cause data loss. The author recommended adding this system to the water circulation to reduce human intervention for improving water quality. In addition, it has been recommended to reduce the energy consumption for all devices used to design the system. Figure 2.20 shows an IoT-based Monitoring System.

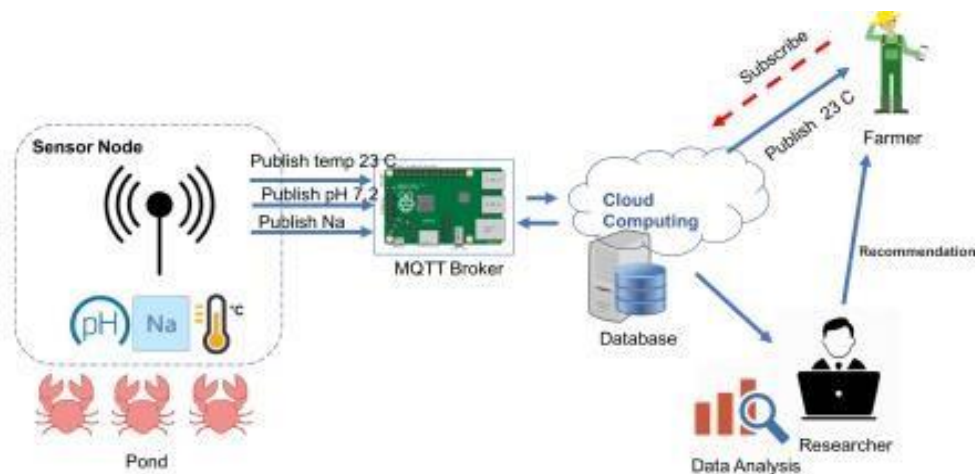


Figure 2.20 IoT-based Monitoring System (Niswar et al., 2018)

A real-time water quality monitoring system has been proposed for aquaculture farmers to alert them if the water body is polluted (Raju & Varma, 2017). The system consisted of several sensors to measure different water quality parameters. The water quality sensor: DO, temperature, carbonates, nitrate, pH, ammonia, and salt, are connected to a Raspberry Pi 3 controller that contains an inbuilt Wi-Fi module. The system was powered using a solar panel to reduce power consumption. The system

always stores data and alerts the farmers if the water conditions are not in the allowable range. In addition, a mobile application was designed to allow the user to monitor the data remotely in real-time so the farmer can view the historically obtained data. Using solar panels will overcome the problem of electricity breakdown as the farmers face a lot of power cuts. When the farmer receives a message alerting him of abnormal water conditions, the message also states how to solve the faced issue. Nevertheless, the overall system's initial cost is high compared to other systems. The cost and energy consumption can be reduced when the system is developed to be automated using an internet network. The system architecture is shown in Figure 2.21.

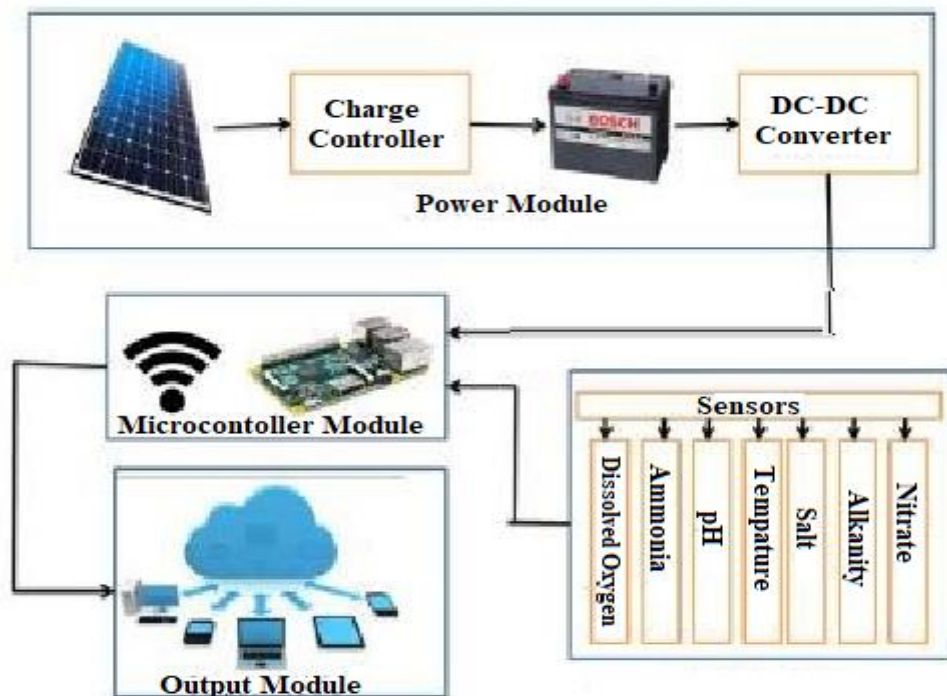


Figure 2.21 System Architecture (Raju & Varma, 2017)

(Vijayakumar & Ramya, 2015) designed a low-cost water quality monitoring system that used smart sensors to measure five water quality parameters; pH, conductivity, turbidity, DO, and temperature. These sensors are connected to a Raspberry Pi B+ controller to process the sensors' data to the USR-WIFI232-X-V4.4 module that transfers the obtained data to the cloud using a gateway. A mobile application was provided to view the data obtained from the sensors. Raspberry Pi controller can be connected to several sensors and interfaces simultaneously, making it

suitable for systems with many inputs and outputs. The controller and the used wireless communication module were the best choice for such a system as it is portable, low cost, and capable of processing, analyzing, sending, and finally viewing data on a mobile phone. Thus, it was more efficient compared to other systems. It has been suggested to measure biological parameters as the system measured only physical parameters and implement the design in many areas to collect more data about the condition of the water body. The overall block diagram is presented in Figure 2.22.

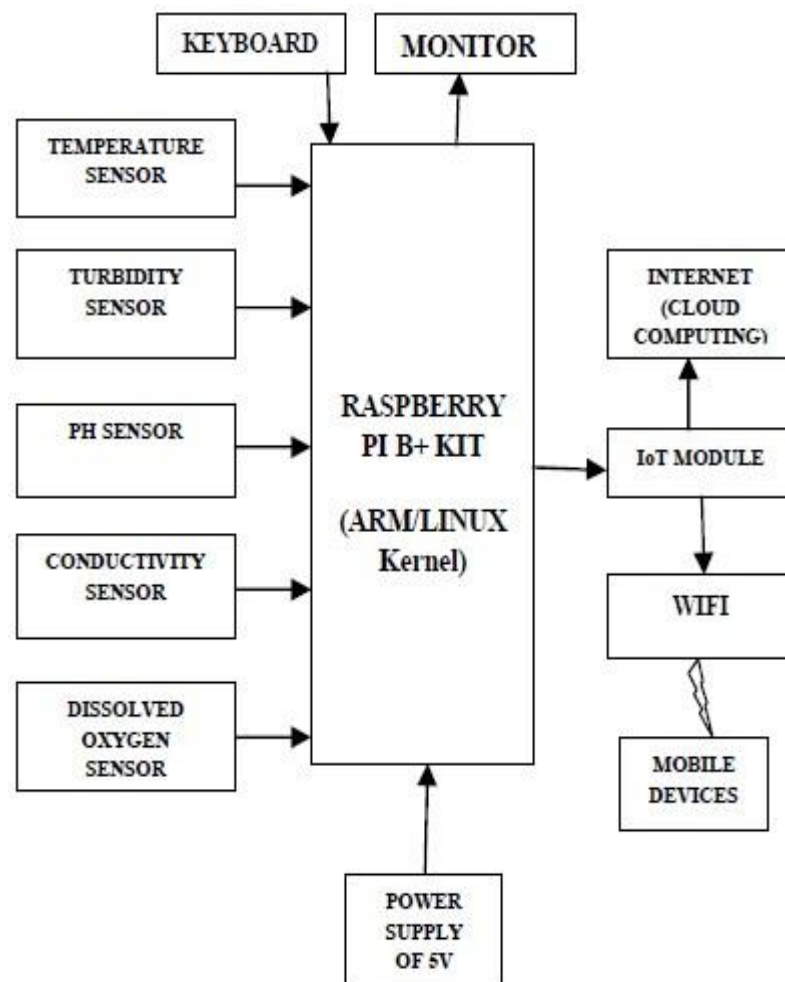


Figure 2.22 Overall Block Diagram (Vijayakumar & Ramya, 2015)

2.4.3 TI CC3200 as Processing Platform

The Texas Instrument CC3200 microcontroller is a single chip with an internal Wi-Fi chip invented for IoT applications. It allows the developer to design a completed application using only a single chip. It contains software, tools, and sample applications

and is easy to program. It has different peripherals, such as I2C, Serial Peripheral Interface (SPI), Universal Asynchronous Receiver-Transmitter (UART), and Analog-to-Digital Converter (ADC) channels (“CC3200 Is the Industry’s First Single-Chip Microcontroller Unit with Built-in Wi-Fi,” 2015). TI CC3200 has been used to design some water quality to process the data obtained from the sensors.

Additionally, (Geetha & Gouthami, 2016) have designed a cheap system for water quality monitoring. To check water quality, the design uses five sensors to measure pH, turbidity, temperature, water level, and EC. The data is collected using a TI CC3200 controller, a single chip with Wi-Fi built into it for wireless communication purposes, as presented in Figure 2.23. The controller, as mentioned above, is programmed to store the data in the cloud using the Ubidots platform to analyze the data after storing it. Testing WHO’s data is added to the module to compare it to the obtained data from sensors. In addition, once readings from sensors reach abnormal values, the user will get an alert that a problem needs to be solved. TI CC3200 controller is easy to use, and its speed is high compared to other processors with external Wi-Fi chips (Texas instrument CC3200 Simple Link, 2017). The controller works in four modes; Hibernate, Sleep, Deep Sleep, and Active. Thus, the consumed power will be reduced because the controller is not always functioning. However, using a Wi-Fi module is not a good choice as it consumes high power. Still, its range of communication is high compared to other communication protocols, not to mention the need for external hardware chips that are not needed anymore when using a Wi-Fi module. It has been recommended to improve the system by implementing machine learning algorithms for detecting abnormal water quality parameter values.

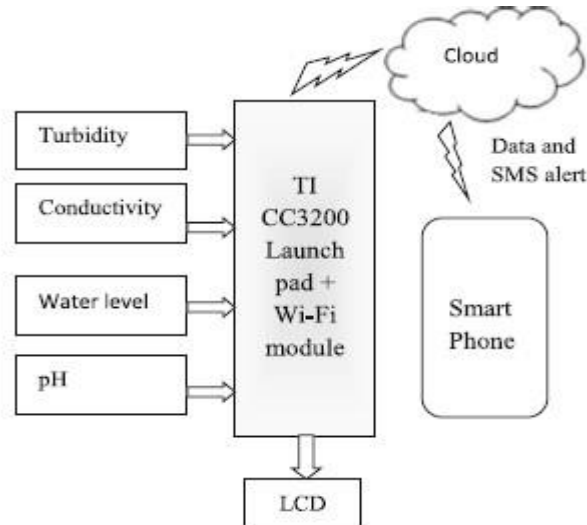


Figure 2.23 Overall Block Diagram (Geetha & Gouthami, 2016)

Furthermore, a new system has been developed to monitor the water quality parameters in real-time to reduce human intervention (Billah et al., 2019). This system was designed to allow the farmers to monitor their waterways to take the actions needed when the water parameters are not good. Thus, three sensors were used to measure temperature, pH, and turbidity. TI CC3200 microcontroller is the main chip used to process the data from the sensors and send it through the Wi-Fi network. Finally, the data was displayed on LCD using graphs, charts, etc. Then, data is transmitted using MQTT to the end-user for monitoring purposes. CC3200 microcontroller has built-in Wi-Fi. Hence, there is no need for outer Wi-Fi equipment. Furthermore, the MQTT protocol is used to make the communication between the microprocessor and end-user easier as it is not complicated. However, the MQTT protocol was low in data rate transmission, and that caused the system to be slow. On the other hand, some challenges have been faced during real implementation, such as the turbidity parameters being unstable most of the time as it is susceptible to the water flowing; this sensor needs some time to get a more stable reading. Besides, the turbidity sensor can only measure the quality, not the quantity. Therefore, the user cannot get the turbidity reading like other sensors. Figure 2.24 presents the overall block diagram.

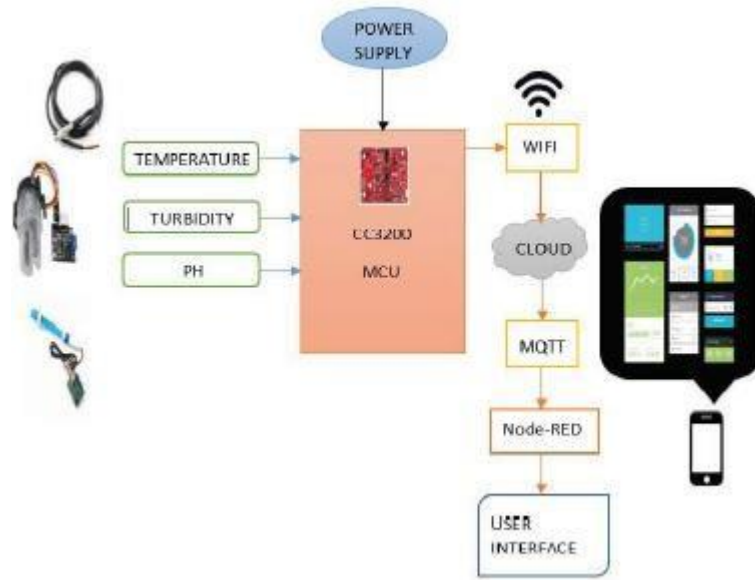


Figure 2.24 Overall Block Diagram (Billah et al., 2019)

2.4.3.1 FPGA as Processing Platform

FPGA is a reconfigurable device that contains many programmable units (Giesemann et al., 2014). This device is an integrated circuit made of semiconductor material. The main advantage of FPGA is any user can reconfigure the device's electrical functionality as it is not hard-coded. These powerful devices can be customized to accelerate key workloads and enable design engineers to adapt to emerging standards or changing requirements. In Section 2.5, we will elaborate more on the architecture of an FPGA.

In (Myint, Gopal, & Aung, 2017), water quality parameters have been monitored using IoT technology using five smart sensors to measure water quality parameters; pH, water level, turbidity, temperature, and carbon dioxide (CO₂). They were measured on the water's surface, as illustrated in Figure 2.25. A Very High-Speed Integrated Circuit Hardware Description Language (VHDL) and C language using the Quartus II tool were used to program the FPGA controller. It is the core system used to collect the data obtained from the sensor and process it to a personal computer. The Zigbee-based wireless communication system was applied to transfer the data remotely from the water location into the main station, where the user can see the water parameters in real-time. The carbon dioxide sensor was SEN0219, which is the best choice for detecting the CO₂ level due to its high stability and sensitivity. The power

consumption is very low, and finally, it is waterproof and does not cause any poisoning. In addition, the Zigbee communication tool is easy to use, install and upgrade. Selecting Nios II was to get the best performance of the processing unit. Moreover, the designed system decreased water quality measurement costs and time consumption. In addition, the work did not maximize the potential of an FPGA by utilizing only one softcore (Nios II) processor. This increases the time to collect the data as it will collect in series instead of parallel. However, the author suggested increasing the number of nodes to cover more areas and measure the water quality for a wide area.

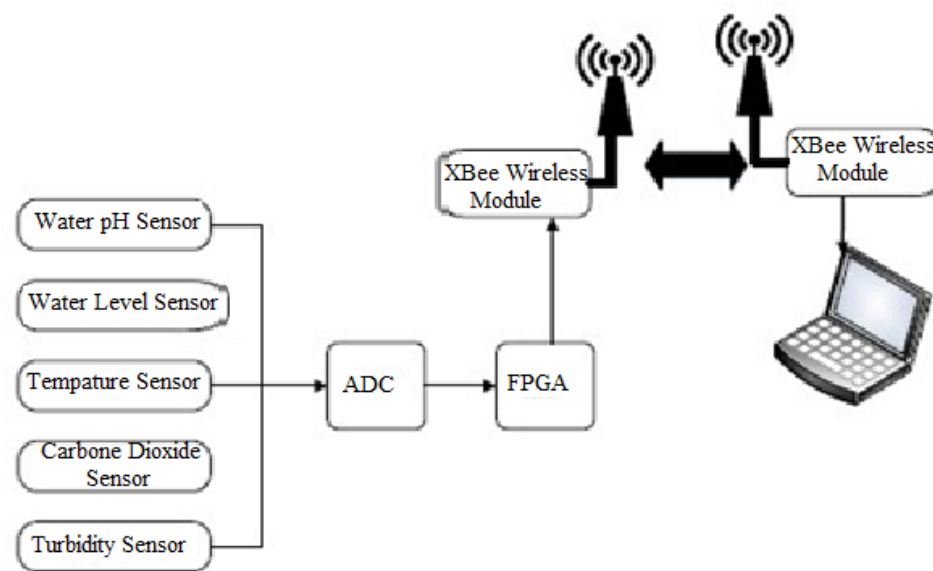


Figure 2.25 The Block Diagram of Smart Water Quality Monitoring System (Myint et al., 2017)

2.4.4 Other Processing Platforms

Besides the above-mentioned platforms, such as Arduino, Raspberry Pi, TI CC3200, and FPGA, there are some works published during the last few years utilizing other microcontrollers. The below studies will mention some of these studies.

(Birje, Bedkyale, Alwe, & Adiwarekar, 2016) developed a new system to monitor two parameters that show if the water is safe for the life of aquatic or not. Note that pH sensors, pH meter, and turbidity sensors were used to measure the mentioned parameters in the water. The sensors were connected to Analog-to-Digital Converter

(ADC) to convert the analog readings of the sensors, as the Peripheral Interface Controller (PIC) microcontroller cannot process analog signals. Finally, the PIC microcontroller was connected directly to the LCD to display the data obtained from the sensors. The system is cost-effective, portable, and easy to set up by the user. The pH meter is a commercially available voltmeter, but it is not suitable because it has very high resistance, so it cannot be used to measure the voltage of the pH electrode. Thus, it is necessary to design a pH meter to overcome the problem mentioned above. PIC microcontroller has the advantage of using software control for self-reprogramming along with a power-saving mode, making it suitable for such a system. However, the system is not real-time monitor the water quality parameters. Plus, it has many hardware modules, such as resistors, capacitors, amplifiers, LEDs, photodiodes, and voltage regulator IC, making it impossible to redesign it again. LCD is not a good choice for monitoring water quality parameters due to its limited size and display. Thus, it is essential to monitor the water quality parameters using a mobile application or a website. The author suggested using the Global System for Mobile communication (GSM) module to view the data remotely on smartphones. Figure 2.26 shows the block diagram of the system.

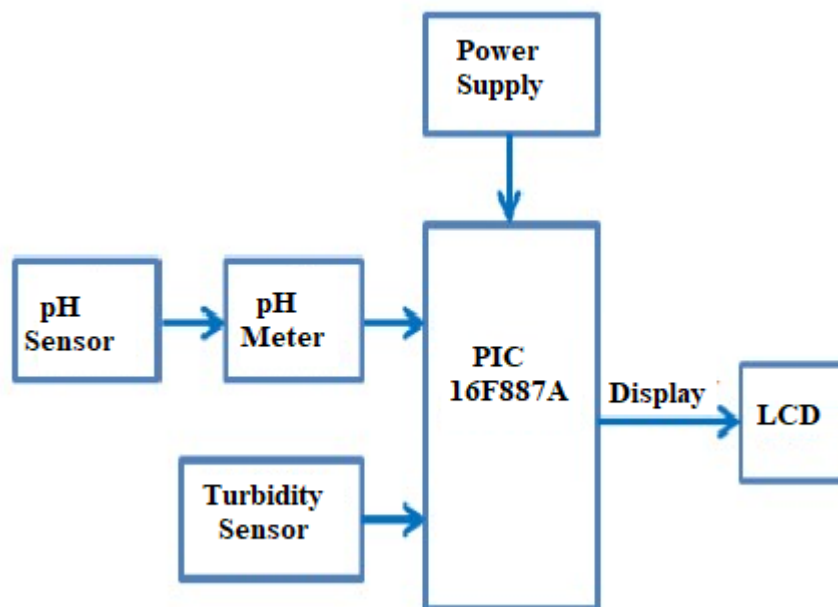


Figure 2.26 Block Diagram of Proposed System (Birje et al., 2016)

(Cloete, Malekian, & Nair, 2016), proposed a low-cost sensor to measure water quality parameters; temperature, pH, ORP, flow, and conductivity. As the sensors were designed locally, adding a signal conditioning circuit was necessary to interface the sensor to the controller. Zigbee wireless communication module was connected to the microcontroller, allowing the obtained data to be sent remotely. Water quality parameters were then displayed on LCD. A buzzer option was added to the system, and the buzzer went off whenever the measurements were out of the allowable range. The sensors were designed locally as most of the water quality parameters are commercially available and cost-effective. The turbine used to design the flow sensor was cheap and could do digital readings. The two-electrode way was used to design the conductivity sensor and was easy to maintain, and its cost was cheap. The Zigbee module uses less power and does not need additional infrastructure, but the sending and receiving range is very small, from 10 m to 70 m. However, some sensors, such as ORP, required additional signal conditioning circuits, which increased the system's complexity. In addition, historical data was unavailable because it is a real-time system. Thus, it is important to add such a function to estimate the water quality over the year. It has been recommended to design a turbidity sensor because it is an essential parameter that needs to be measured to get a low-cost design rather than using expensive sensors.

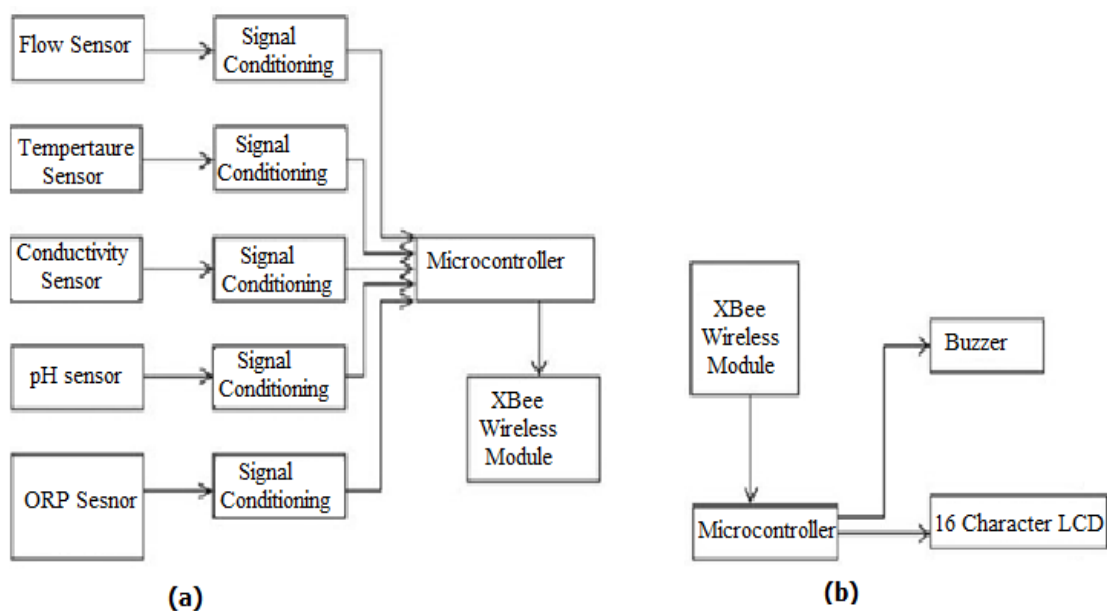


Figure 2.27 (a) Module 1: The Measurement and Sensing Module Block Diagram
 (b) Module 2: The Notification Module Block Diagram
 (Cloete et al., 2016)

(Danh, Dung, Danh, & Ngon, 2020) established a new system known as the Aquaculture system that monitors water quality in the Mekong River. The system used four sensors to measure temperature, pH, DO, ORP, and salinity. When the readings are detected directly, they will be sent to ThingSpeak, an IoT platform, to save the data on the server. The master control unit received updated data measured by the sensors every minute. The system application is available on the App Store on iOS devices and the Play Store for Android users to view the data in real-time. The system has the option of sensor automatic cleaning to remove dust and algae because such dirt may cause unstable sensor measurement; thus, automatic cleaning will increase the system's effectiveness. The master control unit has a built-in wireless communication system to transfer data to the user. An SMS message will alert the user if the data is above or below acceptable levels. However, the system is complicated with many electronic devices for controlling, measuring, data transferring, etc., making it heavy for the user to carry and set up from one place to another. In addition, even with the automatic cleaning system, some of the sensors have been damaged due to dust. Therefore, it is important to use sensors of good quality and waterproof to prevent any damage. Figure 2.28 shows the architecture of the E-Sensor AQUA system.



Figure 2.28 Architecture of the E-Sensor AQUA System (Danh et al., 2020)

2.4.5 Summary of Related Works

Table 2.1 shows the conclusion of all related studies by listing drawbacks, motivation, and future work. Meanwhile, Table 2.2 shows the sensors used in each study to measure physical and chemical water quality parameters.

Table 2.1 Summary Table

No.	Author and year	Motivation	Drawbacks	Future work	Processor
1	(Vijayakumar & Ramya, 2015)	1) Low cost	1) Wi-Fi module is not inbuilt	1) Measuring biological water quality parameters	Raspberry Pi
		2) It is portable	2) It is complex	2) Implementing the design in many areas	
		3) Displaying data on the mobile app			
2	(Khatri et al., 2020)	1) No need for an external Wi-Fi module as Raspberry Pi has an internal one	1) Raspberry Pi with Linux operating system is not user-friendly.	1) Implementing a new system using fuzzy logic in an IoT environment	Raspberry Pi
		2) Easy to function as Raspberry Pi needs only 12 V			
		3) High accuracy			
		4) Cost-effective			
3	(Ngom et al., 2019)	1) No payment for LoRa transmission band.	1) Most water quality parameters were not measured.	Unavailable	Arduino
		2) Low power consumption.	2) Low transmission rate.		

		3) Solar panel alternated power source.	3) System is costly.		
4	(Li et al., 2018)	1) pH sensor can function at 0 to 60°C.	1) System requires calibration before starting.	Unavailable	Raspberry Pi
		2) TDS sensor is waterproof and has high accuracy.	2) Data might lose or delay due to weak Wi-Fi.		
5	(Chowdury et al., 2019)	1) User can get a report about the water condition at any time.	1) Data is only a number.	Unavailable	Arduino
6	(Lezzar et al., 2020)	1) It is a lifespan system that does not require short-term maintenance	1) SIM800c requires GSM/GPRS network to transfer the data	Unavailable	Arduino
7	(Saravanan et al., 2018)	1) Less cost, weight, and size	1) SCADA system cannot be implemented in areas that have no Wi-Fi cover	Unavailable	Arduino
		2) GSM provides high-speed communication			
		3) Using Arduino processor accelerated SCADA system			
		4) Utilizing IoT to control the system remotely.			
8	(Mukta, Islam, Barman, Reza, & Khan, 2019)	1) F1 score showed that the system's overall	1) System measures only physical parameters.	Unavailable	Arduino

		performance was high	2) Not a real-time system		
9	(Pujar et al., 2020)	1) System measure water parameters in different seasons	1) Wide range of data collection	Unavailable	Arduino
10	(Encinas et al., 2017)	1) The system is portable 2) Less costly and power consumption.	1) No user alert in the system.	1) Developing an AI module for providing alerts to the user when water is in bad condition.	Arduino
11	((Niswar et al., 2018)	1) Monitoring water quality remotely 2) Alert the user of any shortcomings	1) Complexity of the system 2) Needs good coverage of 3G and 4G networks	1) Adding the system to the water circulation to reduce human intervention 2) Reducing power consumption	Raspberry Pi
12	(Raju & Varma, 2017)	1) System works all day and stores data 2) Solar panel is used to overcome power problem 3) User receives a message on how to solve any issue	1) Overall cost is high	1) Reducing power consumption by automating it using the internet.	Raspberry Pi
13	(Geetha & Gouthami, 2016)	1) TI CC3200 processor is easy to use and fast	1) Adding Wi-Fi increases	1) Implementing machine learning	TI CC3200

		2) It has four functioning modes	power consumption		
		3) Consume less power			
		4) High range of communication			
14	(Billah et al., 2019)	1) Has a built-in Wi-Fi module	1) Turbidity sensor readings were mostly unstable	Unavailable	TI CC3200
		2) MQTT protocol is user friendly	2) Turbidity sensor does not show numbers.		
15	(Myint et al., 2017)	1) Carbon dioxide has high stability and sensitivity and uses low power	1) VHDL is not easy to use	1) Increasing the number of nodes to cover more areas	FPGA
		2) Nios II provides high performance of processing unit			
		3) Low cost			
16	(Birje et al., 2016)	1) The system is cost-effective, portable as well as easy to set up by the user	1) System is not real-time	1) Using GSM to view data remotely	PIC
		2) PIC microcontroller has software control for self-reprogramming and power-saving mode	2) Many hardware modules make it difficult to be redesigned		
			3) LCD is small		

17	(Cloete et al., 2016)	1) Less cost	1) Sending data remotely with a range from 10 m to 70 m	1) Design a low-cost turbidity sensor	Unavailable
		2) Zigbee module uses less power	2) ORP sensors require additional signal conditioning circuits		
		3) No need for external infrastructure	3) It is only displaying data with storing it		
18	(Danh et al., 2020)	1) Sensor can be automatically cleaned	1) System is complicated as it has many controlling, measuring, data transferring devices	Unavailable	Unavailable
		2) User is updated through SMS if there is any shortcoming	2) Not portable 3) Even with automatic cleaning, some sensors had been damaged due to the dust		

Table 2.2 Summary of All Sensors Used in Each Study Given in Table 2.1

Parameter	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	No. of times the sensor is used
pH	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	18
Water level	X	X	X	X	X	X	X	X	X	X	X	X	✓	X	✓	X	X	X	2
Turbidity	✓	X	X	✓	✓	✓	X	✓	X	X	X	X	✓	✓	✓	✓	X	X	9
Conductivity	✓	✓	✓	X	✓	✓	X	✓	✓	X	X	X	✓	X	X	X	✓	X	9
Temperature	✓	✓	✓	✓	✓	X	✓	✓	✓	✓	✓	X	✓	✓	✓	X	✓	X	16
Flow	X	X	X	X	X	X	✓	X	X	X	X	X	X	X	X	X	✓	X	2
Pressure	X	X	X	X	X	X	✓	X	X	X	X	X	X	X	X	X	X	X	1
Chlorine	X	X	X	X	X	✓	X	X	X	X	X	X	X	X	X	X	X	X	1
Color	X	X	X	X	X	X	✓	X	X	X	X	X	X	X	X	X	X	X	1
Nitrate	X	X	X	X	X	X	X	X	✓	X	X	✓	X	X	X	X	X	X	2
Salinity	X	X	X	X	X	X	X	X	X	X	✓	✓	X	X	X	X	X	✓	3
Ammonia	X	X	X	X	X	X	X	X	X	X	X	✓	X	X	X	X	X	X	1
CO	X	X	X	X	X	X	X	X	X	X	X	✓	X	X	X	X	X	X	1
CO2	X	X	X	X	X	X	X	X	X	X	X	X	X	X	✓	X	X	X	1
DO	✓	✓	✓	X	X	X	X	X	X	✓	X	✓	X	X	X	X	X	✓	6
BOD	X	X	X	X	X	X	X	X	✓	X	X	X	X	X	X	X	X	X	1
TDS	X	X	X	X	X	X	X	X	✓	X	X	X	X	X	X	X	X	X	1
ORP	X	✓	✓	X	X	✓	X	X	X	X	X	X	X	X	X	X	✓	✓	5
Sensors used in the Study	5	5	5	3	4	5	5	4	6	3	3	5	5	3	5	2	5	4	

2.5 PROPOSED FPGA-BASED SWQS

Based on the findings from the literature review, as summarized in Section 2.4, it is concluded that there are several processors have been used to design SWQS, such as Arduino, Raspberry Pi, FPGA, TI CC3200, and others. However, Arduino, Raspberry Pi, TI CC3200, and other processors are limited by the number of pins that are hard-coded. The FPGA platform, on the other hand, could reconfigure the interface of each pin, as well as it has a large number of pins. This means it allows the FPGA to be used for the real end product and not just as a prototyping platform. In addition, when it comes to speed, FPGA surpassed other processors with a frequency reach of up to 1 GHz. Speed is an important factor that needs to be considered, as it can affect the system performance when many sensors are connected to the processor. Additionally, any sensor with any interface could be easily connected to an FPGA.

The selection of sensors used in this work as a proof-of-concept was based on the importance of the water quality parameters and the availability of the sensors. From

Table 2.2, it can be seen that the pH sensor and temperature are most commonly used. Unfortunately, the temperature sensor is not widely available. The TDS sensor could measure the total dissolved materials in a liquid, and the conductivity could be easily extracted from the TDS sensor; it will be two parameters in one sensor. Finally, the turbidity sensor, which measures the clarity of the water, is also selected as one of the parameters to be measured, given that it is one of the most commonly used sensors after pH and temperature.

Furthermore, the number of sensors used in previous studies varies between two (2) and five (5). Therefore in this proposed work, three (3) parameter sensors will be put to the test-taking, i.e., the mid-point number of parameters. The next section will look at the architecture of an FPGA to understand how FPGA can be used in heterogeneous systems.

2.6 UNDERSTANDING THE FPGA-SOC HETEROGENEOUS SYSTEM

FPGA-SoC is a heterogeneous platform that can improve the performance of an embedded system using more than one processor. This section will explain the architecture of an FPGA, followed by the software development tools used in this research.

2.6.1 FPGA Chip

FPGA is a semiconductor IC where the design engineers may reconfigure most of the electrical functionality inside the device, whether during the Printed Circuit Board (PCB) assembly process or after the FPGA platform has been shipped out to customers (What Is FPGA, 2020). FPGAs benefit designers of many types of electronic equipment, including smart energy grids, aircraft navigation, medical ultrasounds, and data center search engines (What Is FPGA, 2020). Figure 2.29 demonstrates the Cyclone V SoC FPGA solution from intel.



Figure 2.29 Cyclone V SoC FPGA from Intel (Intel PSG Website, 2020).

2.6.2 System-on-Chip

On the other hand, System-on-Chip (SoC) is a hardware platform containing many different microprocessor subsystems, memories, and Input/Output interfaces (J Greaves, 2011). The design of a modern SoC is a complex task involving a range of skills and a deep understanding of a hierarchy of perspectives on design, from processor architecture down to signal integrity (Brackenbury, Plana, & Pepper, 2010). Other than that, the SoC design methodology is a new model for electrical and computer engineering education in digital logic and microelectronics (William D. & Dennis A., 2000). FPGA-SoC is one of the powerful SoCs used in the proposed SWQS. Section 2.6.4 presented in detail all the features of the FPGA-SoC heterogeneous platform.

2.6.3 Heterogeneous Platform

As described in Section 1.2.2, a heterogeneous platform is a new computer platform infrastructure that presents a next-generation hardware platform and associated software that allows processors of different types to work efficiently and cooperatively

in shared memory from a single source program (Hwu, 2016). Architectural heterogeneity improves platform flexibility by exploiting more than one processor.

2.6.4 Features of FPGA-SoC Heterogeneous Platform

The DE10-Nano Development Kit is one type of FPGA-SoC that provides a robust hardware design platform utilized by Intel FPGA-SoC, which combines the latest dual-core Cortex-A9 embedded cores with industry-leading programmable logic for ultimate design flexibility. Intel's SoC integrates an ARM-based Hard Processor System (HPS) consisting of processor, memory interfaces, and peripherals tied with the FPGA fabric using a high-bandwidth interconnect backbone. The DE10-Nano development board is equipped with Ethernet networking, high-speed, analog to digital capabilities, and DDR3 memory (Terasic, 2017a)

Unlike application-specific integrated circuits (ASICs), which can only be implemented once through manufacturing, FPGAs can be reconfigured following the customer's needs. This feature allows the user to make improvements and change the architecture of the FPGA, allows fixing bugs, or uses FPGAs to rapidly prototype hardware designs, which can later be manufactured as ASICs. In addition, FPGA can be reconfigured quickly to finish different tasks (Ziener, 2018). However, the flexibility and reconfigurability increase cost and design complexity and provide less specialized components such as floating-point operations. Hence, the main purpose of using FPGA is to utilize its huge parallelism, i.e., its ability to perform pipelines to overcome latency issues (Purkayastha, Shiddhibhavi, & Tabkhi, 2018). ASICs approach can offer very high performance and huge power efficiency; however, it lacks flexibility and the ecosystem offered by Programmable Logic Devices (PLD) (Burgio et al., 2016). FPGAs have many advantages over other processors, such as reconfigurability, and by default, it provides software flexibility. In addition, more complex water quality systems are demanded as they do not require image processing. They also require communicating with other devices and offer a usable user interface.

While the FPGA-SoC heterogeneous platform enhances the performance of any design, it is not as popular as it costs higher when compared to processors like Arduino. In addition, the design complexity increases when moving to the hardware level.

Nevertheless, when it comes to speed, the FPGA has a higher processing speed in which the frequency can reach up to 1 GHz, while a processor like Arduino is at 24 MHz. FPGA can be used not only for designing a small prototype but also to design a real end product. It is interconnected, and reconfigurable features allow flexibility and additional functions to be included without changing the main processor.

2.6.5 Memory Management on FPGA-SoC

There are three layers of the memory models in the FPGA-SoC. Figure 2.30 shows an example of a memory model of the proposed SWQS, and it can be described below:

- i. **Level 1 Memory (L1)** is the local memory for each core. Each core's firmware will be stored in built-in memory for each core. When the ARM processor configures the FPGA, each level 1 memory and its firmware will be loaded, allowing the processor core to run.
- ii. **Level 2 Memory (L2):** this level of the shared memory among all cores. This memory will be utilized to store the obtained data from each sensor by cores. The ARM processor will format this data with a timestamp ready for collection.
- iii. **Level 3 Memory (L3):** this is a Double Data Rate type 3 memory (DDR3 memory) of an ARM processor that will be utilized to run the Linux operating system. It is the global memory for the user applications' data processing.

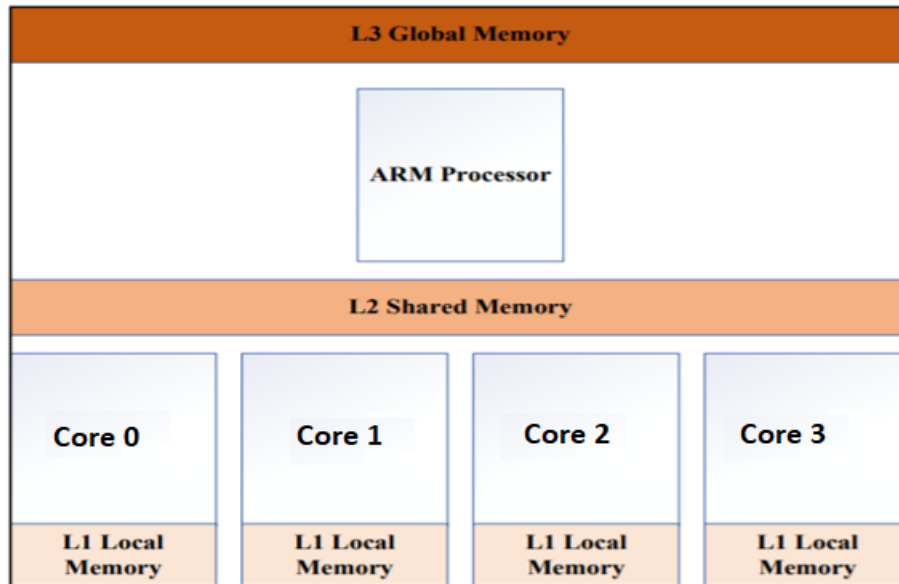


Figure 2.30 Example of a Memory Model of the Proposed SWQS Design

2.6.6 Quartus II Development Software

Quartus II is a tool provided by intel to implement the hardware design on Intel FPGA products. It offers a full range of features for each phase of the hardware design flow of intel FPGA to enhance the design cycle and achieve the highest performance on FPGA (*Intel® Quartus® Prime Standard Edition Handbook Volume 1: Design and Synthesis*, 2018). These features can be summarized below:

- **Project setup:** Quartus Prime helps the designers to create new projects, add/create design files, specify the target FPGA device, and design constraints files. This allows the designer to create multiple design versions that run on multiple FPGA devices to achieve the highest performance possible.
- **Design planning tools:** plan for initial I/O pin layout, power consumption, and area utilization in the Early Power Estimator, the Power Analyzer Tool.
- **Integrated Synthesis:** it provides efficient synthesis support for VHDL (1987, 1993, 2008), Verilog Hardware Description Language (HDL) (1995, 2001), and SystemVerilog (2005) design entry languages.

- **System and IP Integration:** this defines and generates a complete system in much less time than using traditional, manual integration methods with Platform Designer.
- **Platform Designer:** Platform Designer is the latest generation system that integrates hardware and software designs. Platform Designer will save time and big effort in the design process of FPGA via generating logic interconnection automatically to connect Intellectual Property (IP) functions and other subsystems. The Platform Designer utilizes a robust hierarchical framework that offers a swift response to connect large systems. In addition, it gives support to Blackbox entities. This will enable the Platform Designer to provide a quick response time in starting systems and creating new connections by generating and operating on IP functions that changed. Furthermore, the tool of Platform Designer supports different design entryways, like blocked-based design entry, block boxes, Register Transfer Level (RTL), and schematic entry (Intel, 2021a).

2.6.7 Quartus Intellectual Property Libraries

FPGA design contained components from softcore invented by intel, known as the IP library. These components are built in multiple processing cores architecture. Each core consists of a processor along with some necessary components to run the core. Each core contains the following items that have the advantage of accessibility for each processing core individually:

- i. **Nios II softcore processor** is an Intel processor that is softcore configurable 32-bit. It can be implemented and programmed to the FPGA. This processor comes with two types: f/core (for fast) and e/core (for economics). The proposed design will only utilize the f/core due to licensing. Quartus Prime software will be used to design the water quality system and compile the design on FPGA.
- ii. **On-chip memory:** it is a Random-Access memory (RAM) that is utilized to store Nios II processor firmware. Furthermore, the details of the firmware

and its benefits will be elaborated in Chapter 4, Section 4.2.2.1. It is L1; hence, it can only be accessed through one core. This type of memory is situated inside an FPGA chip.

- iii. **Avalon System Interconnect:** it is a bus system that was developed by Intel Programmable Solutions Group. This bus system connects all system components to the memory-mapped interface type.
- iv. **System Timer:** it is a softcore internal timer. It was used to synchronize the collection and transfer of the data among system cores.
- v. **System ID:** this component is utilized to provide each core Identifier number. The ID number will help identify which core should be running.
- vi. **I2C Controller:** I2C is a data transfer protocol that uses two serial wires. This controller can be used to control I2C sensors.
- vii. **SPI Controller:** Serial Peripheral Interface (SPI) is a data transfer protocol to control SPI sensors.
- viii. **ADC Controller:** Analog-to-digital controller (ADC) is used to control the flow of data of ADC-type sensors.

Some shared peripherals subsystem is a combination of IPs design that several processing cores can access as shared resources. These IPs give system-level services for the processing cores, such as mailbox, mutual exclusion (mutex), shared memory, Advance eXtensible Interface (AXI)-Avalon translation (for ARM SoC interface and handshaking), and a clock that can be synchronized using Phase Lock Loop (PLL). Elaboration on shared peripherals is shown below:

- i. **Mutex** is a mutual exclusion peripheral that will be used to control the cores' operation. In addition, it helps in arranging the accessibility of cores to each shared peripheral. In the proposed design, mutex IP is required as it is a multi-core system. Therefore mutex is required for controlling the access to shared memory and peripheral between cores. For example, when the mutex is idle, the collected data from the pH sensor will be written to the shared memory. However, if the mutex is not idle because the shared

memory access is given to the TDS sensor or turbidity sensor, then the firmware will be waiting for the mutex to be free.

- ii. **PLL:** for synchronizing the operation clock of the cores of the processors, PLL is utilized.
- iii. **Mailbox:** it is a softcore IP that is utilized to send messages among cores.
- iv. **Shared Memory:** it is the L2 of the system. When mutex approves the access, shared memory can be used by multiple processing cores.

2.6.8 Linux Terminal Interface

An embedded Linux application is a special version of the Linux operating system running on the embedded computer system, like FPGA. This requires a processor to be part of this FPGA, such as a hard-core processor. Note that a hard-core processor is a hard IP integrated into FPGA. Generally, a hard-core processor runs at 600 MHz to 1 GHz, such as an ARM processor. The entire system will be controlled through this application as the application reads segmented and analyzed data from all sensors stored in the shared memory. PC, laptop, or any screen can be connected to FPGA directly and display the data from shared memory. However, this does not make the system portable. Therefore, Raspberry Pi was used to be connected ARM terminal for accessing it, reading the water quality data, and displaying it. Raspberry Pi was chosen as Linux operating system. Further, from the Linux Terminal in the Raspberry Pi operating system, the ARM terminal can be accessed. Linux application has been used as it can be implemented on Raspberry Pi processor as well as ARM terminal could be accessed using Linux Terminal on Raspberry Pi.

2.6.8.1 Linux Kernel Compilation

While there are various selections of Linux compilation software, the Ubuntu 20.02 machine is used to compile the Intel Angstrom distribution. Some packages must be installed on the host PC to permit the compilation of Angstrom distribution. The

packages listed down are not fixed lists since the installation depends on the requirement of each machine.

```
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get install phablet-tools sed wget cvs subversion git-core coreutils unzip texi2html
texinfo libstdc++-dev docbook-utils gawk python-pysqlite2 diffstat help2man make gcc build-
essential g++ desktop-file-utils chrpath libgl1-mesa-dev libglu1-mesa-dev mercurial autoconf
automake groff libtool xterm lib32z1
```

In addition, it is necessary to make a setting for the /bin/sh point of bash instead of a dash by running a command of the terminal.

```
sudo dpkg-reconfigure dash
```

The root filesystem and Linux kernel can be compiled using recipes from Angstrom. The Yocto Project (YP) is building the embedded Linux when the recipes are cloned. YP is an open-source collaboration project that helps developers create custom Linux-based systems regardless of the hardware architecture. The project provides a flexible set of tools and a space where embedded developers worldwide can share technologies, software stacks, configurations, and best practices. These can be used to create tailored Linux images for embedded, and IoT devices or anywhere a customized Linux Operating System is needed. YP can be run using the below commands:

```

$ cd ~
$ mkdir angstrom-build
$ cd angstrom-build
$ wget http://releases.rocketboards.org/release/2019.10/src/altera.xml
$ wget http://commondatastorage.googleapis.com/git-repo-downloads/repo
$ chmod 777 repo
$ export PATH=$PATH:~/angstrom-build
$ repo init -u git://github.com/Angstrom-distribution/angstrom-manifest -b angstrom-v2019.06-warrior
$ mkdir -p .repo/local_manifests
$ mv altera.xml .repo/local_manifests/
$ repo sync
$ MACHINE= cyclone5
$ sed -i '/meta-altera/a \\ \ ${TOPDIR}\layers\meta-altera-refdes \\' conf/bblayers.conf # This is to
add the meta-altera-refdes layer to conf/bblayers.conf
$ sed -i '/meta-atmel/d' conf/bblayers.conf # this has conflicting packages
$ sed -i '/meta-freescale/d' conf/bblayers.conf # this has conflicting packages
$ echo "DISTRO_FEATURES_remove = \" wayland \"" >> conf/local.conf # broke builds prior to
this not needed for GSRD
$ echo "DISTRO_FEATURES_remove = \" alsa \"" >> conf/local.conf # breaking builds prior to
this not needed for GSRD
$ rm layers/meta-altera-refdes/recipes-devtools/socfpga-test/socfpga-test_1.0.bb # developmental
unit tests stub for older kernels

```

```

$ export KERNEL_PROVIDER=linux-altera-ltsi
$ export KERNEL_TAG=refs/tags/ACDS19.3_REL_GSRD_PR
$ export KBRANCH=socfpga-4.14.130-ltsi
$ export BB_ENV_EXTRAWHITE="$BB_ENV_EXTRAWHITE KBRANCH KERNEL_TAG
UBOOT_TAG KERNEL_PROVIDER"
$ bitbake gsr-console-image

```

Images directory will then contain all the generated images after the compilation of the YP. Table 2.3 presents the important files list of the YP.

Table 2.3 Files Generated After the Yocto Compilation

File	Description
u-boot-*	U-Boot ELF file
u-boot-*.bin	U-Boot binary file
u-boot-*.img	U-Boot image
vmlinux	Kernel file
zImage	Compressed kernel image
gsrd-console-image-*.cpio	Root Filesystem in cpio archive format
gsrd-console-image-*.ext3	Root Filesystem as ext3 image
gsrd-console-image-*.tar.gz	Root Filesystem in tar gzip archive format
gsrd-console-image-*.tar.xz	Root Filesystem in tar archive format
gsrd-console-image-*.jffs2	Root Filesystem in jffs2 format
console-image-*.cpio	Standard Angstrom Root Filesystem in cpio archive format
console-image-*.ext3	Standard Angstrom Root Filesystem as ext3 image
console-image-*.tar.gz	Standard Angstrom Root Filesystem in tar gzip archive format
console-image-*.tar.xz	Standard Angstrom Root Filesystem in tar archive format
console-image-*.jffs2	GSRD Root Filesystem in jffs2 format

Linux kernel and U-boot are then compiled from the git trees in <https://github.com/altera-opensource>. Finally, the git source can be cloned, and the Linux kernel and U-boot can be compiled using the below instructions.

```

$ cd ~
$ wget https://releases.linaro.org/archive/14.04/components/toolchain/binaries/gcc-linaro-arm-
linux-gnueabi-4.8-2014.04_linux.tar.bz2
$ tar xjf gcc-linaro-arm-linux-gnueabi-4.8-2014.04_linux.tar.bz2
$ export CROSS_COMPILE=~/.gcc-linaro-arm-linux-gnueabi-4.8-2014.04_linux/bin/arm-linux-
gnueabi-
$ cd ~
$ git clone http://github.com/altera-opensource/u-boot-socfpga
$ cd u-boot-socfpga
$ git checkout -b test_branch
$ export CROSS_COMPILE=~/.gcc-linaro-arm-linux-gnueabi-4.8-2014.04_linux/bin/arm-linux-
gnueabi-
$ make mrproper
$ make socfpga_eyclone5
$ make
$ cd ~
$ git clone https://github.com/altera-opensource/linux-socfpga
$ cd linux-socfpga
$ git checkout -b test_branch TAG # Refer to the tag in the above
$ export CROSS_COMPILE=~/.gcc-linaro-arm-linux-gnueabi-4.8-2014.04_linux/bin/arm-linux-
gnueabi-
$ export ARCH=arm
$ make socfpga_defconfig
$ make zImage

```

Therefore, after doing all the above steps, the Linux image is ready to be burned on the SD card. Plus, the SWQS Linux application is ready on the ARM processor.

2.7 SUMMARY

In this chapter, the literature review of water quality monitoring systems has been explored and investigated. This chapter summarized some studies developed to measure physical and chemical water quality parameters by identifying gaps in the current studies and showing the weakness, challenges, and critical analyses. The challenges and motivations of each design were conducted in this chapter, followed by the description of the proposed FPGA-based SWQS. Later, the chapter provides a general overview of the FPGA platform along.

CHAPTER THREE

METHODOLOGY

3.1 OVERVIEW

The primary objective of designing and developing in situ Smart Water Quality System (SWQS) Field Programmable Gate Array (FPGA)-based kit is to enhance and improve the safety of the water. This can be realized using the integration of sensors and processors to detect the materials that cause water contamination. The liquid quality parameters will be measured using different sensors to reach high quality.

This chapter describes the methodology proposed to develop the SWQS engine and derive the data acquisition procedure from this system. Furthermore, this chapter elaborates on the experimental details and how sensors will be set up along with FPGA to measure the water quality. Section 3.2 introduces the research methodology phases based on four main phases and then specifies the methodology for designing SWQS. In addition, Sections 3.3 and 3.4 illustrate the hardware and software design of the proposed SWQS, respectively. Section 3.5 discusses the proposed design power cycle, while Section 3.6 proposes the testing plan of the design. This chapter elaborated on the technical guideline for an SWQS system utilizing multiple processing cores covering both software and hardware design. Section 3.7 covers the FPGA design cycle that contains the hardware design flow by including all components required to design a processing core and all the processing elements. In addition, the flow of the software design for both core firmware development, as well as the Linux applications development will also be explained.

The embedded Linux design cycle and all the system components to design a Linux image will be demonstrated in Section 3.8. On the other hand, Section 3.9 elaborates on the steps of adding the processing cores to the proposed hardware design based on the designer's preferences.

3.2 RESEARCH METHODOLOGY

The research methodology of this study was developed based on four phases; these phases are summarised in Figure 3.1. Each phase has its description and what outcome is derived.

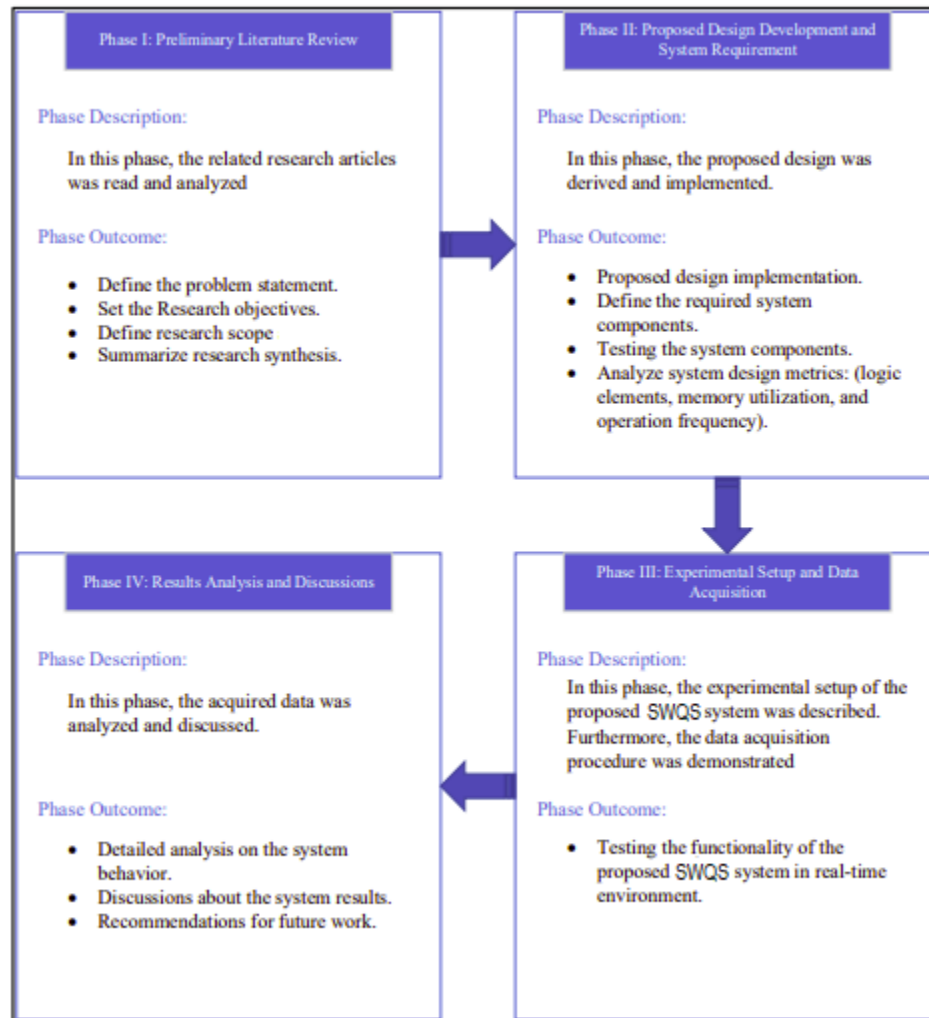


Figure 3.1 Research Methodology Phases

Phase I was accomplished in Chapters 1 and 2. The problem statement, objectives, and scope were discussed in Chapter 1. Furthermore, the research literature review, motivation, challenges, and design aspects were discussed in Chapter 2. Phase II and Phase III were covered in Chapter 3. All the design considerations, methodology, implementations, and testing scenarios were discussed. Finally, Phase IV was covered in Chapter 5. All the results of the system design, data collection, system reliability, data validity, and testing scenarios were discussed.

The investigation that has been done on the water quality parameters was done by reading books and articles. From that, the literature has been elaborated with related works. Next, motivations, challenges, and recommendations have been derived and concluded. The proposed design was chosen based on the issues and problems of the literature; however, there are many of them. Therefore, the scope of the research was designed to focus on the hardware design and make SWQS that eliminates the issue of performance and reconfigurability. In addition, based on the scope of the research, three objectives have been set to reach SWQS with high performance, high efficiency, less power use, and a system that can be reconfigured. Figure 3.2 shows the research methodology of the proposed SWQS.

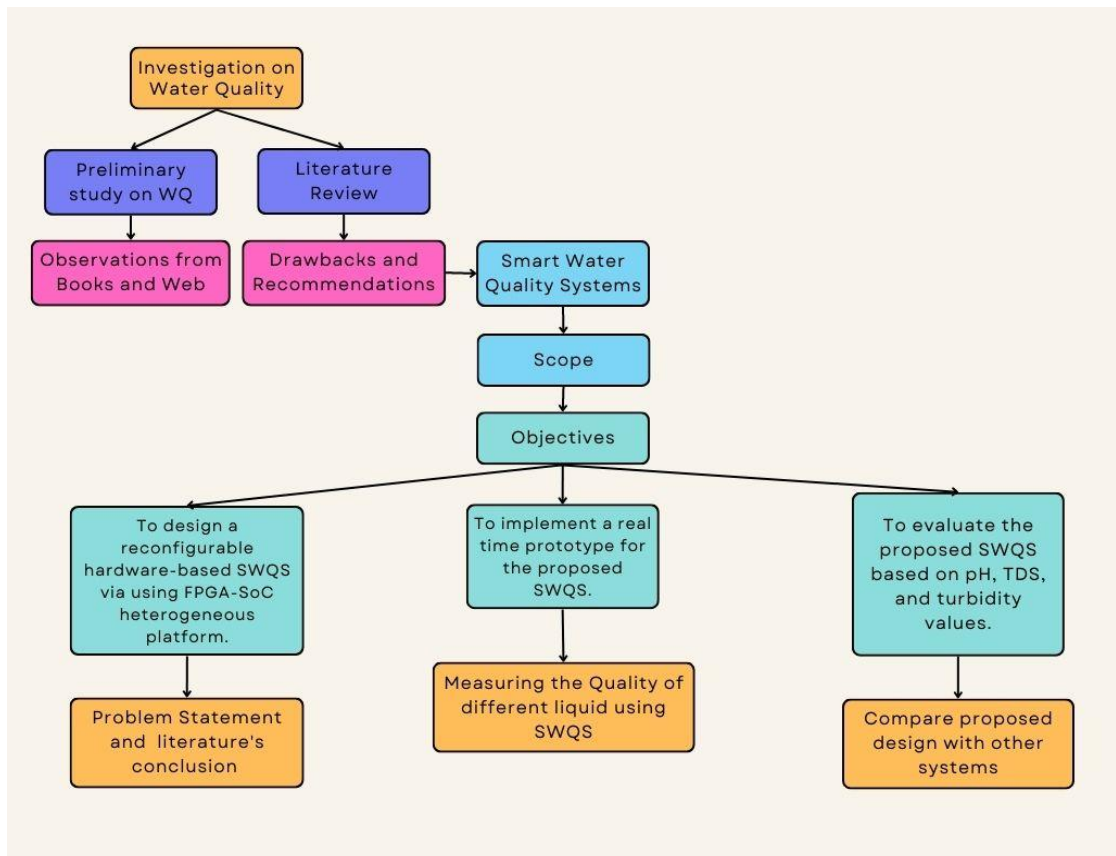


Figure 3.2 Research Methodology of the Proposed SWQS

The Hardware design flow on FPGA starts with building the embedded system architecture using Platform Designer in Quartus Prime. Platform Designer is the next-generation system integration tool of Quartus Prime. Once the design is done in Platform Designer, a design compilation is needed to transfer the Verilog system files to the programmable file used to configure the FPGA. The compilation must be successful in passing this step. Subsequently, assign the pins in another tool called Pin Planner from Quartus Prime. After that, the Quartus Prime project can be compiled to generate the software object file (sof), which can be used to program the FPGA.

Once the hardware flow is done, the developer can start the software flow. The software flow generates the executable link and format file (.elf), which is the processor firmware. Once the .elf file is ready, it can be stored in the processor's core memory.

Linux application development flow starts after the completion of the hardware flow and firmware development flow. In this research, the SWQS application was developed using C++ programming language. The development starts with the memory management step, defining the memory layout. After that, the address of the Advance eXtensible Interface (AXI) bridge should be defined. Consequently, this address will be added to the hardware registers span of Intel FPGA. Therefore, the result address will be used to access the FPGA-related design registers by adding the offset of each register. After that, a virtual memory must be created in any Linux-based application to avoid segmentation faults due to the memory protection unit of Linux, which restricts any direct access to an address in the system.

Next is the generation of the Bootloader and the Linux image. The tool used to generate the Bootloader is Embedded Software and Tools for Intel System-on-Chip (SoC) FPGA software (SoC EDS). After finishing the Quartus Prime project, the Handoff folder was created. This folder contains information about the hard processor system components and connections. This folder is used with the SoC EDS to generate the pre-loader and U-boot. Once the Bootloader folder is created, a compilation is required to generate the binary files of the Bootloader.

Raspberry Pi was used to run Linux operating system on it, and then the Linux application in the ARM processor was accessed using the Linux Terminal on the Raspberry pi. Figure 3.3 summarizes the entire methodology of this work toward designing and developing an FPGA-based SWQS.

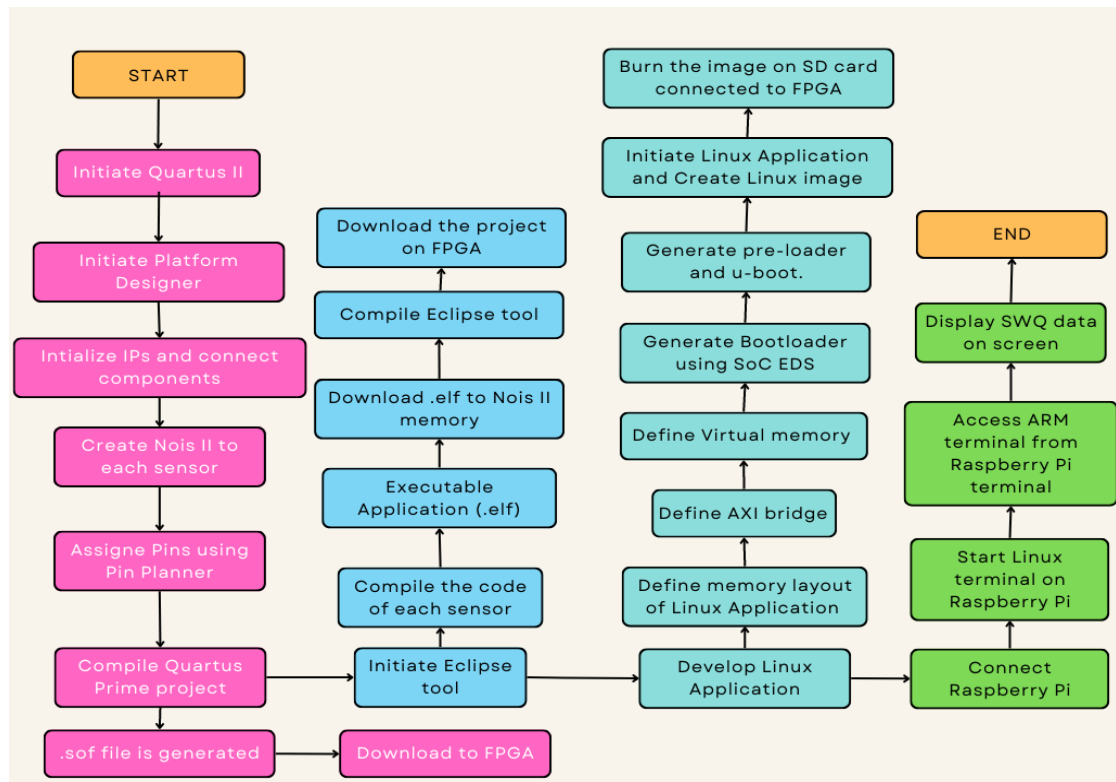


Figure 3.3 Overall methodology of the Proposed SWQS

3.3 PROPOSED DESIGN DEVELOPMENT AND SYSTEM REQUIREMENT

3.3.1 Smart Water Quality System (SWQS)

The proposed design was built on a DE10 Nano FPGA-SoC development kit, which was connected to three water quality sensors named pH, Total Dissolved Solids (TDS), and turbidity. The selection of sensors may be added depending on the requirements of agencies. Section 3.10 of this dissertation will explain how the changes can be done at the software level.

Figure 3.4 presents the general overview of the proposed system and its data flow. In the beginning, the DE10 Nano FPGA-SoC development kit was connected to three water quality sensors named pH, TDS, and turbidity. Note that the main FPGA chip is divided into two parts. The first is the main chip, which focuses on processing the data from sensors and making it ready to display. On the other hand, the second part is the ARM processor, which has the Linux application. The ARM processor was utilized to run the Linux application. This application controls the entire system, reads the segmented, and analyzes data from sensors stored in the shared memory. The next step was to read the data from the ARM processor; therefore, a PC or laptop could be used to display the data. However, the PC and laptops or any screen are not portable, and for that reason, they were replaced by Raspberry Pi. The Raspberry Pi processor can be connected to any screen easily same as other processors like Arduino. On the other hand, Raspberry Pi was used as Linux operating system can be run on it. The Linux application in the ARM processor could be accessed using the Linux Terminal on the Raspberry pi. The entire system is portable and can be fed power via a 20000 mAh power bank.

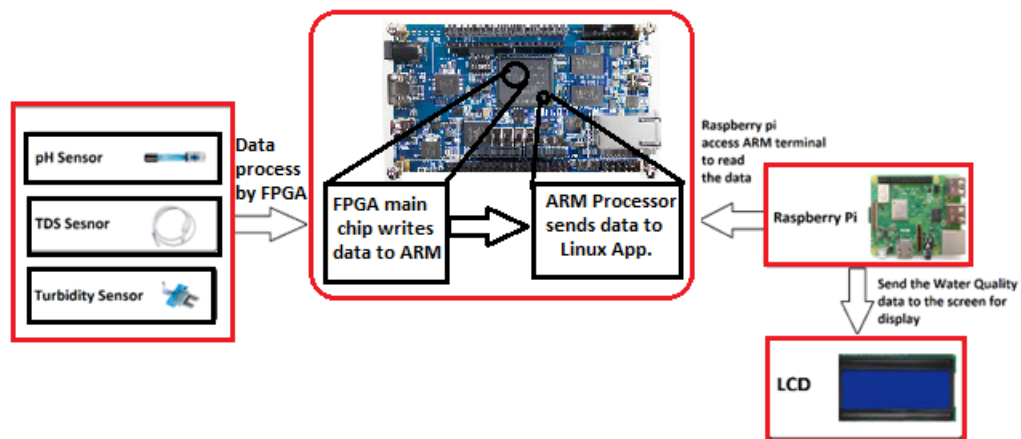


Figure 3.4 Proposed System Blocks and Data Flow

The complex requirements of developing SWQS designs and applications have been discussed in the literature review in terms of computation power, algorithms complexity, resource flexibility, and cost. To overcome this trade-off among design requirements and considerations, a heterogeneous platform is the critical design

approach in SWQS applications. The generic proposed multi-core heterogeneous platform can be seen in Figure 3.5.

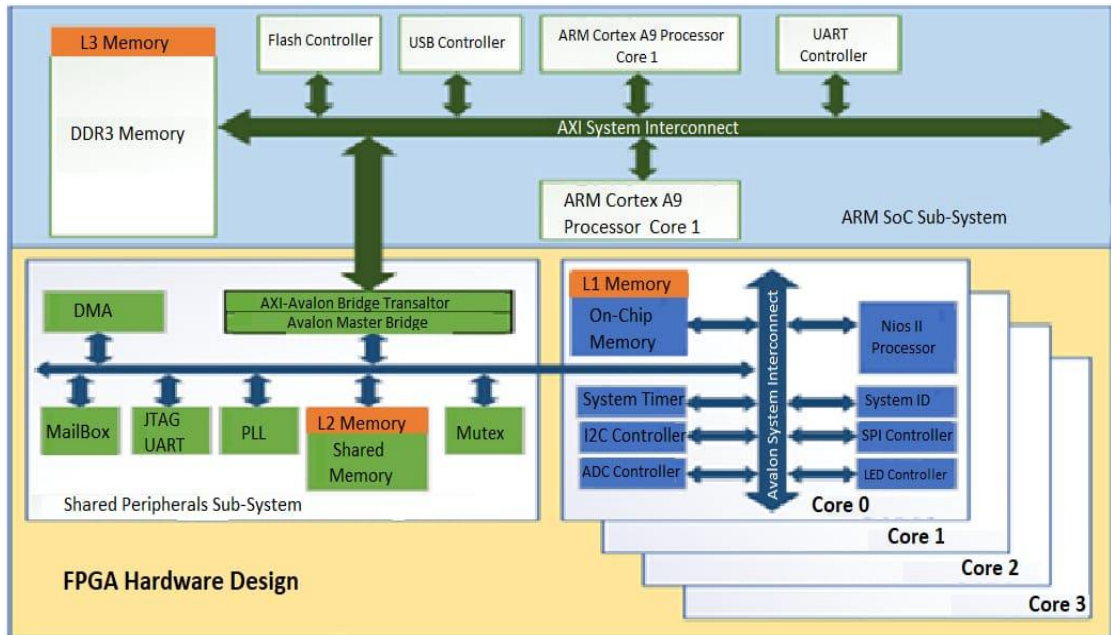


Figure 3.5 Multi-Core Heterogeneous System Architecture Design of the Proposed Design

The hardware design was developed based on digital design methodology on FPGA to parallelize the SWQS functionality based on each developed core. In comparison, the software stack was developed on an ARM SoC sub-system to improve the computation power of the ARM processor. Figure 3.6 presents the FPGA single-core processing element of the proposed design.

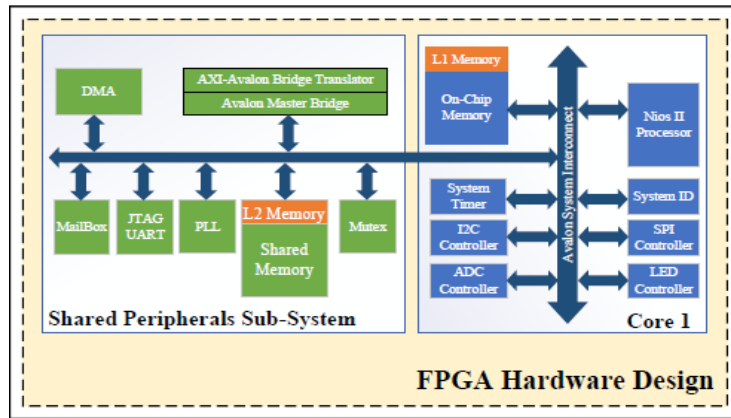


Figure 3.6 FPGA Single Core Processing Element of the Proposed Design

3.3.1.1 Interfacing Multiple Sensors

The hardware limitations in the proposed design were minimized through FPGA. The designer can add the preferred sensor with any data transfer protocol, like Inter-Integrated Circuit (I2C), Universal Asynchronous Receiver-Transmitter (UART), General Purpose In/Out (GPIO), and Serial Peripheral Interface (SPI). All the data transfer protocols can be easily implemented on FPGA.

From Figure 3.7, the processing cores will be utilized for specific SWQS functionality. The proposed SWQS hardware data acquisition system was developed based on the following functionalities: pH detector, TDS detector, and turbidity detector. Each of these functions was assigned to a processing core, core 1 is the pH detector, core 2 is the TDS detector, core 3 is the turbidity calculator, and core 0 is the system synchronizer. Each core is running independently, which might cause data synchronization issues. Hence, core 0 was utilized to synchronize collected data before sending it to the ARM SoC sub-system. In addition, core 0 is responsible for packetizing the collected data from the other cores and initializing the DMA, which is responsible for transferring the collected data from the FPGA domain to the ARM sub-system domain. Finally, the Linux Terminal on Raspberry Pi was utilized to access the ARM terminal and send the data to the screen.

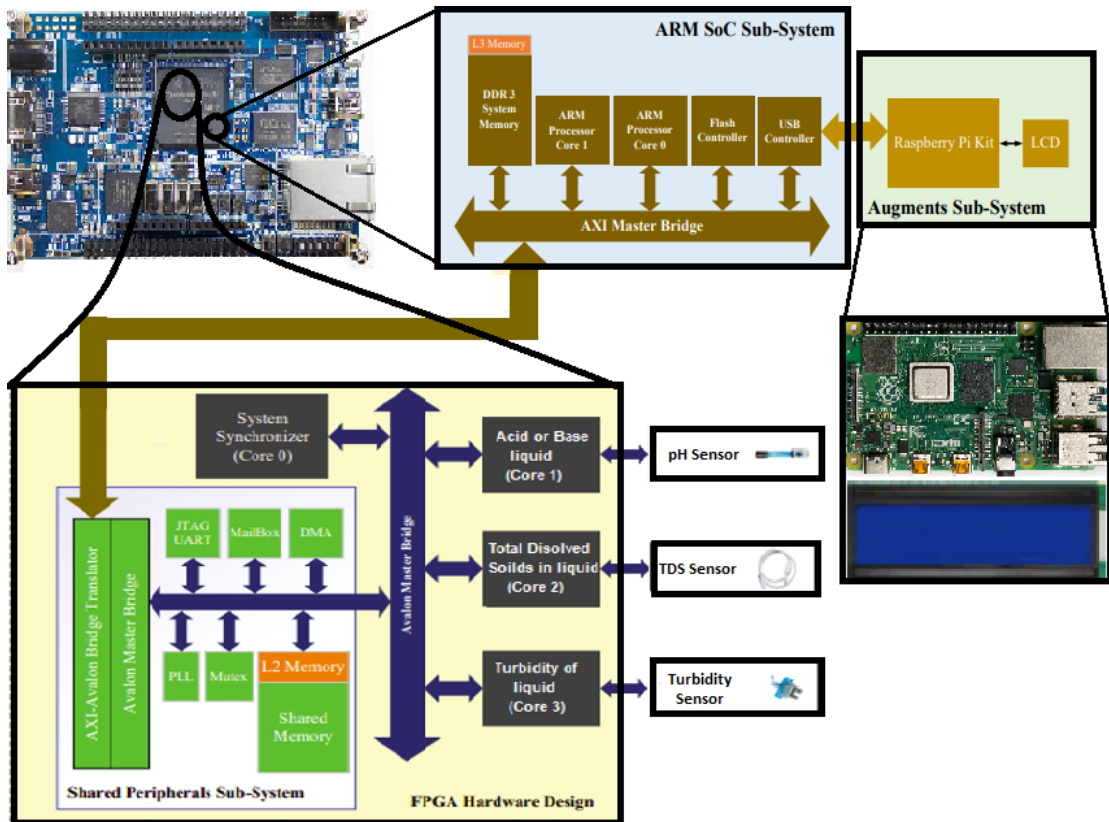


Figure 3.7 Proposed SWQS Hardware Proposed Data Acquisition Design

3.4 SWQS HARDWARE DESIGN

The proposed SWQS hardware design has two parts: hardware and software. The FPGA hardware design flow begins with using Platform Designer to build the embedded platform architecture in Quartus Prime. Section 3.8.1 demonstrates the configuration steps of the SWQS.

After the design in the Platform Designer is done, a compilation of a design is required for FPGA configuration by transferring the files of the Verilog system to a programmable file on an FPGA device. The step of compilation should be successfully done to pass this stage. The next step is to assign the FPGA interface pins once the Qsys file generated from Platform Designer is done. Other than that, the pins can be assigned using the Pin Planner tool in Quartus Prime software. The location of each pin is obtained from the user manual of the development board. After that, the Quartus Prime project should comply to get the sof used for FPGA programming. Figure 3.8 exhibits the FPGA hardware design flow of the proposed SWQS.

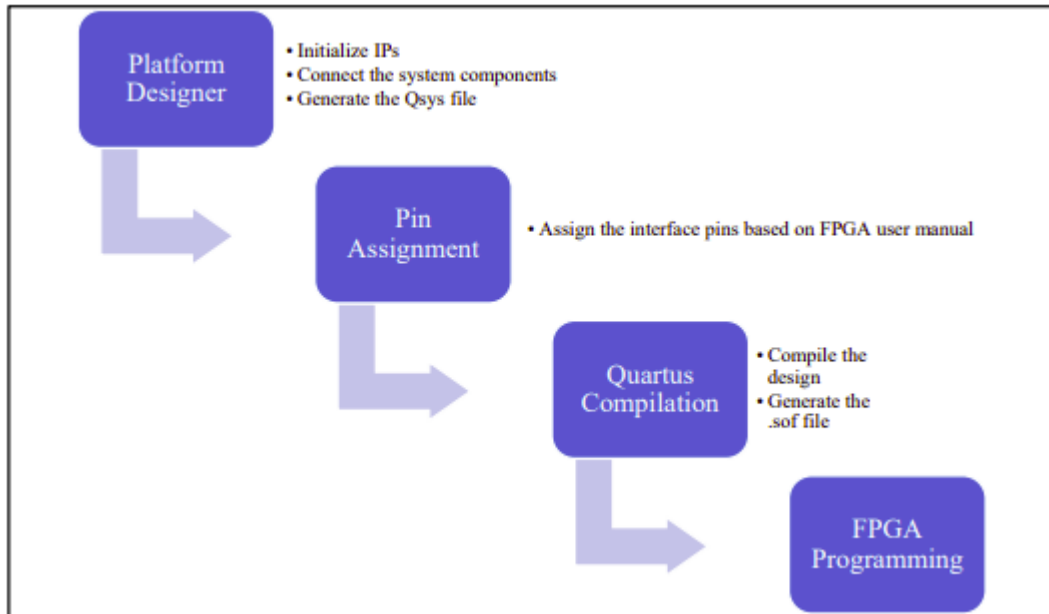


Figure 3.8 The Summary of the FPGA Hardware Design Flow

3.5 SWQS SOFTWARE DESIGN

The software part of the proposed design will have four firmware applications when separated cores are used for each firmware. As shown in Figure 3.9, system synchronization of data collection and the transmission between FPGA and ARM will be done using firmware 0. Note that firmware 1, firmware 2, and firmware 3 will be responsible for initializing, calibrating, controlling, and collecting data from the pH sensor, EC sensor, and temperature sensor, respectively.

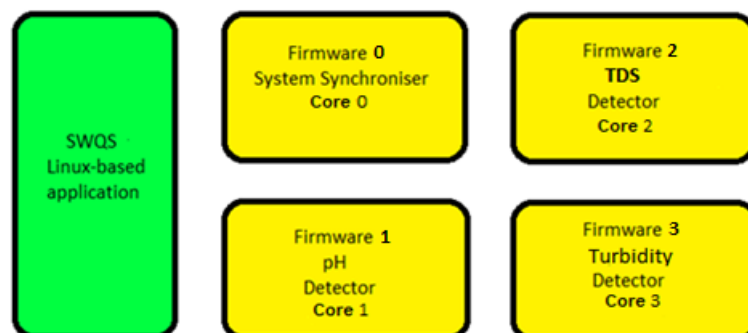


Figure 3.9 Firmware Applications for Each Core with SWQS Linux-based Application

System synchronization of data collection and the transmission between FPGA and ARM will be done using firmware 0. First, firmware 0 starts the system's peripherals, such as Direct Memory Access (DMA), mutual exclusion (mutex), SPI, and GPIO. Consequently, it configures the DMA and checks the flag of the shared memory. When new data is collected from the cores of the cores, the collected data is then going to be packetized and will be written to the source memory location of DMA. After that, DMA will be triggered to transfer the data to the system's Double Data Rate type 3 (DDR3 memory). The firmware 0 flow diagram is displayed in Figure 3.10.

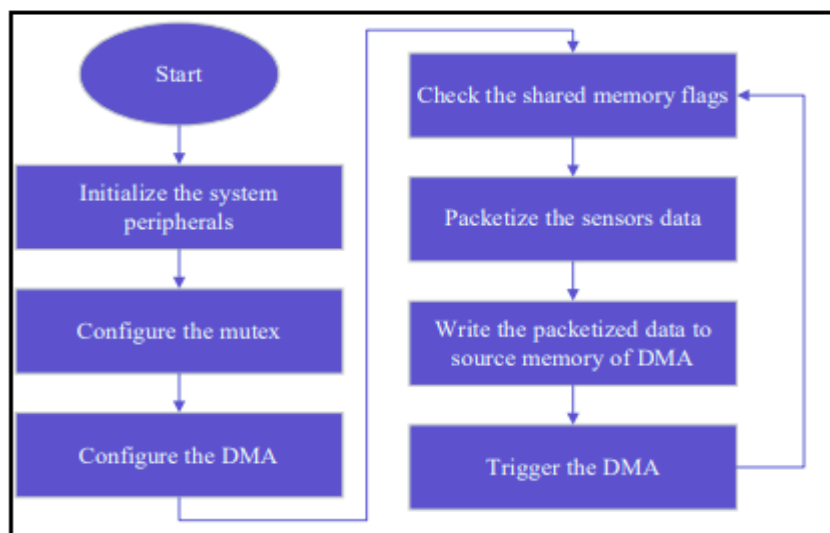


Figure 3.10 Flow Diagram of Firmware 0

Firmware 1, firmware 2, and firmware 3 will be responsible for initializing, calibrating, controlling, and collecting data from the pH sensor, TDS sensor, and turbidity sensor, respectively. Calibrating the sensor is important to make sure that the sensors are functioning with high accuracy and to prevent uncertain measurements of the sensors. Apart from that, each sensor's firmware will start the process of initializing the controller of the sensor, which is the process of defining and identifying the channels of each sensor. It is done at the first usage. In addition, calibrating the sensor is required to ensure high-accuracy measurements and to prevent uncertain sensor readings. Next, the controller will check the sensor ID whether is valid or not to confirm the sensor connectivity. At the moment, the sensor connectivity is confirmed. The controller will read data from each sensor and store the collected data in the sensor's local memory.

The firmware will then check the status of the mutex to have access to the shared memory. When the mutex is idle, the collected data from the sensor will be written to the shared memory. However, if mutex is not idle because the shared memory access is given to other sensors, then the firmware will be waiting for the mutex to be free. Figure 3.11 illustrates the sensor's core firmware block diagram.

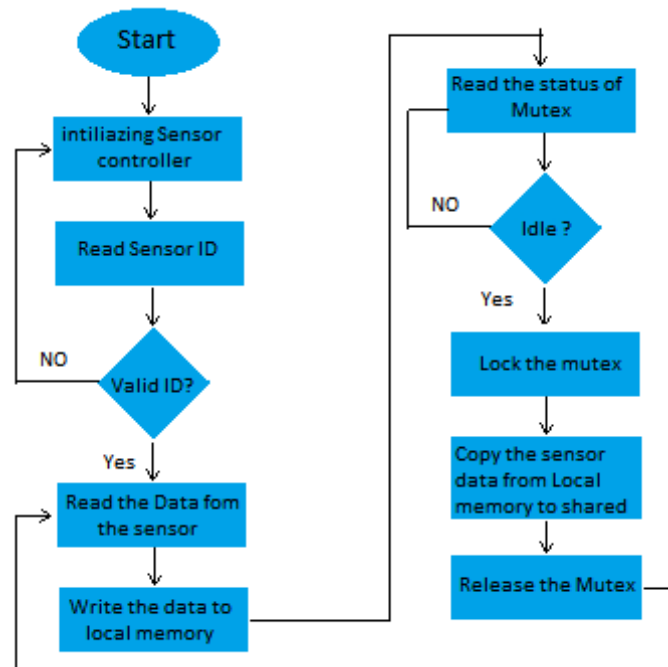


Figure 3.11 The Sensor's Core Firmware Flow Diagram

On the Linux side, the SWQS application will be developed using Linux C++ libraries. The developed application will be used to control the cores that are running on FPGA. Each core can be individually reset, print the system's status, and finally print out the collected data stored in DRR3 memory. This application will ease the interaction between the user and the hardware layer without the need for interaction with the hardware directly or frequent updating. In addition, the most important feature is that any runtime environment or application running on a Linux operating system can be easily integrated with this application.

From a software standpoint, our proposed design will add layers between hardware and application layers. As can be seen in Figure 3.12, the layer is known as the middleware layer. Accessing and modifying the hardware will be easier with the

existence of this layer. Many Application Programming Interface APIs are provided by the middleware layer that can be used to access the hardware. Moreover, these APIs will ease the development because these APIs can cover multiple hardware accesses, and any software platform can be migrated to these APIs based on the requirements.

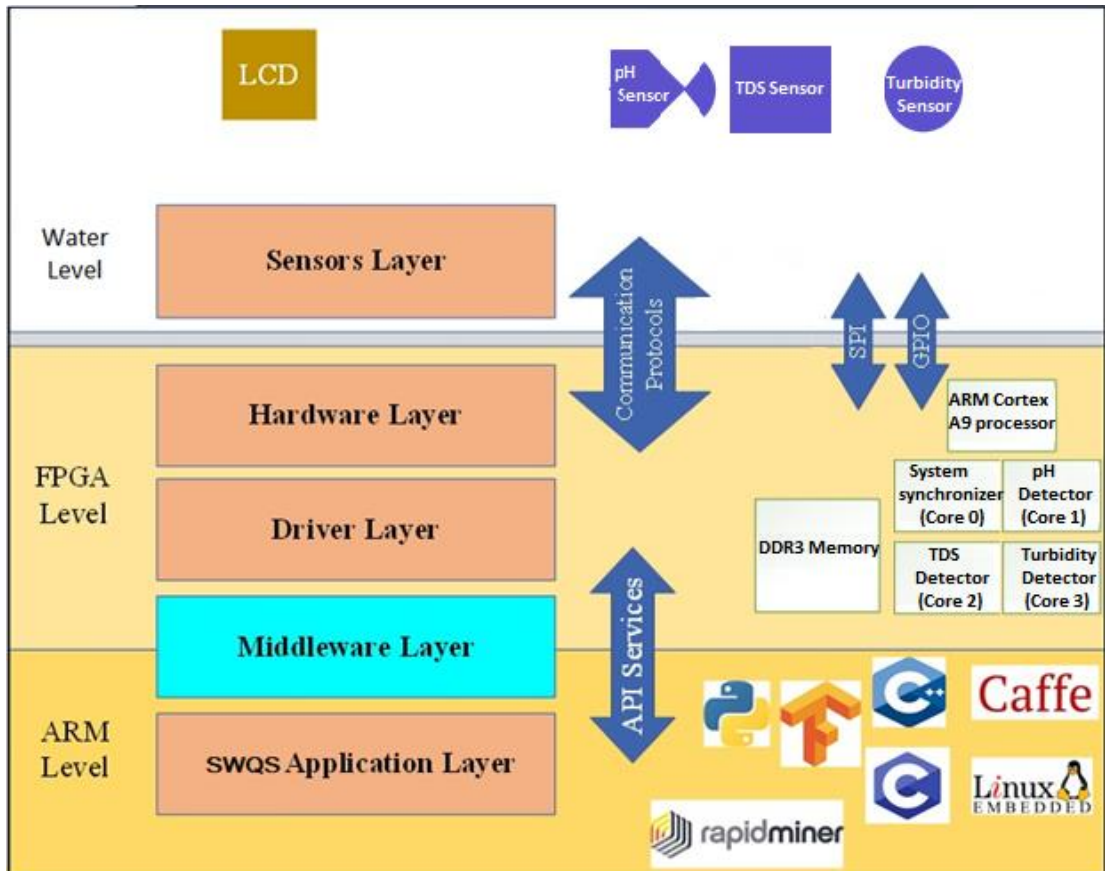


Figure 3.12 The Proposed Design Abstraction Layers

3.6 PROPOSED DESIGN POWER CYCLE

There are two operations that the proposed SWQS design has, one operation for the FPGA side, while the other operation is for the SoC side. First, the system's image will be stored on a global memory (L3) SD card mounted to the system board. The SD card contains the required files for FPGA configuration and running the embedded Linux system. These files are Linux Kernel Image, Root File System (Rootfs), FPGA configuration file, First Stage Bootloader (pre-loader), Second Stage Bootloader (U-boot), Linux Device Tree, and embedded Linux C application.

Once the system is powered on, the pre-loader will be loaded from the SD card to the ARM on the memory chip to start the chip pins, DDR3 calibration, and the clocks. Subsequently, the U-boot is loaded to DDR3 to start the Mux pins of the Hard Processor System (HPS), configure FPGA, load the Rootfs and device tree to DDR3, and finally give the needed control to the kernel image for booting the system. At the time the configuration of FPGA is done, the four cores are going to be loaded and ready to start running. ARM will begin a test to verify the availability of the cores, L1, L2, and L3 initialization, and the response of the sensors.

3.7 PROPOSED DESIGN TESTING PLAN

This research has proposed a new system design of SWQS based on a heterogenous platform known as the FPGA-SoC platform. The verification of the design will be done based on component-level testing. The design cores will be tested separately to achieve the needed function. Additionally, the peripheral's sub-system functionality will be tested individually. The proposed verification is going to be as the following:

- i. Core 0 data packetize test.
- ii. Core 0 DMA run test.
- iii. Core 1 pH test.
- iv. Core 1 data transfer test.
- v. Core 2 TDS test.
- vi. Core 2 data transfer test.
- vii. Core 3 turbidity test.
- viii. Core 3 data transfer test.
- ix. Mutex functionality test.

The system implementation and data collection were the second tests that will be required to be done. The proposed system was meant to ensure the utilization of new sensors is an easy task and needs less effort to add more components to the system. The data will be collected by the cores from pH, TDS, and turbidity sensors. In addition, the raw data from each sensor will be converted by the core into an understandable user format. In that format, a timestamp will be added to each data sample to gain the

system's synchronisation. Thus, L2 local memory will store the data in the form of 128-bits divided into 32-bit to the core ID, 32-bit to the timestamp, and 64-bit to the data sample. Once the data is ready at the core level, the synchronizer core, core 1, will send the collected data to the ARM processor via DMA.

Consequently, testing the overall system by collecting the data from each sensor in a real-time environment to ensure it is functioning well. The proposed design will be tested on different liquids for each sensor to prove that the sensors read the correct data. The first test will be done on pure water, lemon juice, and milk to test the validity of the pH sensor. At the same time, the TDS sensor will be tested on pure water and water with salt to verify the sensor functionality. Last but not least, the turbidity sensor will be tested on three liquids: pure water, water with little dust, and water with more dust. The liquid choice was made based on the sensor functionality. For example, to test the pH sensor, pure water is required to be tested as its pH value is 7, while milk will be used to see whether the pH sensor readings will be more than 7 or not. Correspondingly, lemon juice is used as an acidic liquid, and the pH value of acidic liquids is less than 7, as it is known. The main goal of the proposed design is a proof of concept that the system is usable and can be used in real life.

3.8 THE DESIGN CYCLE OF THE PROPOSED FPGA-BASED SWQS

The design cycle consists of the hardware flow in terms of the configuration of the Platform Designer. Other than that, it consists of software flow, which is the programming of the sensors using the Eclipse tool, and the flow of the Linux application development. In the subsequent subsections, the FPGA-based hardware and software design flow will be elaborated.

3.8.1 The Hardware Design Flow of SWQS

The sensors are connected to FPGA to process analog data from the sensors and convert it into digital for processing purposes. Each sensor is connected to a separate Nios II core to increase the processing speed, as the cores will function in parallel. The proposed SWQS design can be found in Figure 3.13. The proposed architecture contains

the processors, memory, clocking and synchronization, data transfer protocol system peripherals, and bridges.

Use	C...	Name	Description	Export	Clock	Base	End	IRQ
<input checked="" type="checkbox"/>		clk_50mhz	Clock Source		exported			
<input checked="" type="checkbox"/>		pll	PLL Intel FPGA IP		clk_50mhz			
<input checked="" type="checkbox"/>		jtag_uart	JTAG UART Intel FPGA IP		pll_outclk0	0x0002_0000	0x0002_0007	
<input checked="" type="checkbox"/>		led_pio	PIO (Parallel I/O) Intel FPGA IP		pll_outclk0	0x0001_0040	0x0001_004f	
<input checked="" type="checkbox"/>		pb_pio	PIO (Parallel I/O) Intel FPGA IP		pll_outclk0	0x0001_00c0	0x0001_00cf	
<input checked="" type="checkbox"/>		hps	Arria V/Cyclone V Hard Processor System		multiple	multiple	multiple	
<input checked="" type="checkbox"/>		alvmm_bridge	Avalon-MM Pipeline Bridge		pll_outclk0	0x0000_0000	0x3fff_ffff	
<input checked="" type="checkbox"/>		default_slave	PIO (Parallel I/O) Intel FPGA IP		pll_outclk0	0x0000_1040	0x0000_104f	
<input checked="" type="checkbox"/>		address_span	Address Span Extender		pll_outclk0	multiple	multiple	
<input checked="" type="checkbox"/>		core1_nios2	Nios II Processor		pll_outclk0	0x4000_0800	0x4000_0fff	
<input checked="" type="checkbox"/>		ocram_core1	On-Chip Memory (RAM or ROM) Intel ...		pll_outclk0	0x5000_0000	0x5001_ffff	
<input checked="" type="checkbox"/>		core2_nios2	Nios II Processor		pll_outclk0	0x4004_0800	0x4004_0fff	
<input checked="" type="checkbox"/>		ocram_core2	On-Chip Memory (RAM or ROM) Intel ...		pll_outclk0	0x4002_0000	0x4003_ffff	
<input checked="" type="checkbox"/>		core3_nios2	Nios II Processor		pll_outclk0	0x4004_0800	0x4004_0fff	
<input checked="" type="checkbox"/>		ocram_core3	On-Chip Memory (RAM or ROM) Intel ...		pll_outclk0	0x4002_0000	0x4003_ffff	
<input checked="" type="checkbox"/>		shared_core_memory	On-Chip Memory (RAM or ROM) Intel ...		pll_outclk0	0x0000_0000	0x0000_0fff	
<input checked="" type="checkbox"/>		mutex	Avalon Mutex Intel FPGA IP		pll_outclk0	0x0000_1058	0x0000_105f	
<input checked="" type="checkbox"/>		timer	Interval Timer Intel FPGA IP		pll_outclk0	0x0000_1000	0x0000_103f	
<input checked="" type="checkbox"/>		sysid	System ID Peripheral Intel FPGA IP		pll_outclk0	0x0001_0000	0x0001_0007	
<input checked="" type="checkbox"/>		adc_controller	adc_1tc2308		multiple	0x0000_1050	0x0000_1057	

Figure 3.13 The Architecture of the Proposed SWQS

3.8.1.1 Processors

One ARM hard-core processor system has been utilized for the proposed SWQS design, along with three Nios II soft-core processors. The processor has used the AXI protocol to transfer the data between the FPGA and the ARM processor. There are four bridges to transfer the data: FPGA-to-HPS (F2H), which has 128-bit data from FPGA to ARM processor. However, the second bridge is HPS-to-FPGA (H2F), with 128-bit data transferred from the ARM processor to FPGA. In addition, the third bridge has 64-bit H2F control signals. Finally, Avalon Bridge is utilized for SDRAM transactions that transfer data from m FPGA to DDR3 memory of ARM HPS. Figure 3.14 demonstrates the configuration of the ARM processor.

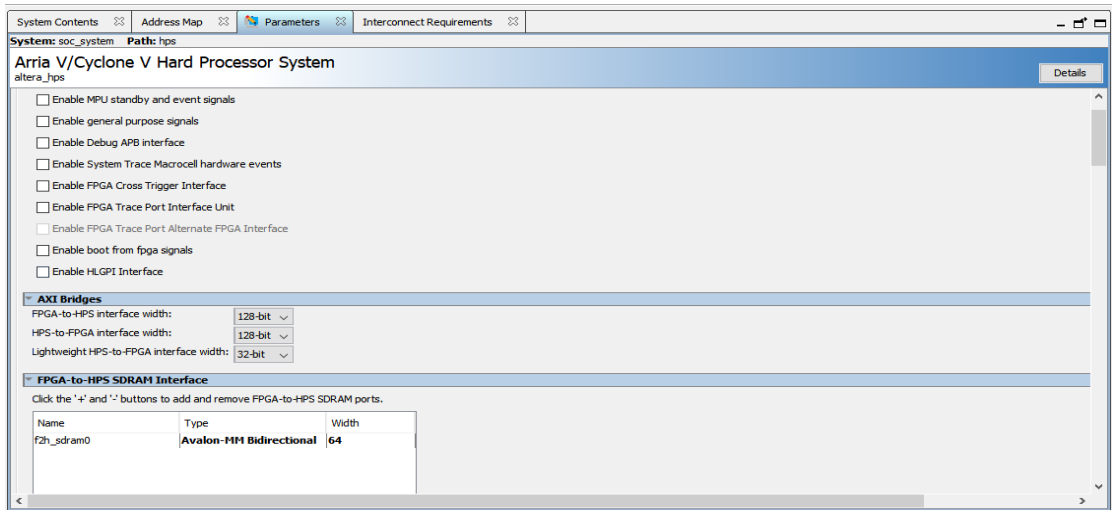


Figure 3.14 The Configuration of the ARM Processor

The Nios II processor is a soft-core processor that can be designed and programmed on FPGA. Figure 3.15 presents the single Nios II processor configuration. The settings of the Nios II processor have been kept as default. However, two factors need to be modified: reset vector and exception vector. When resetting the processor, the firmware in memory is known as the reset vector. However, the memory location required when executing an interrupt is known as the exception vector (Intel Corporation, 2015). Nios II core has a specified On-Chip Memory for reset and exception vectors.

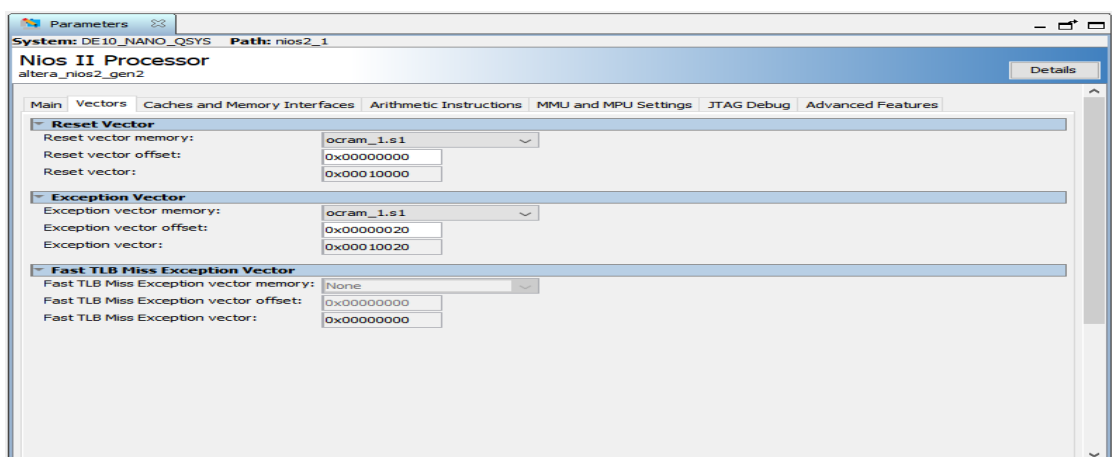


Figure 3.15 Nios II Processor Configuration

3.8.1.2 Memory

The utilized memory for the Nios II firmware is built internally in the FPGA. The memory is split into three, one part for each core. Figure 3.16 displays the configuration of the On-Chip Memory controller. Random-Access Memory (RAM) is used with a size of 64 KB for each core. Note that the data bus is 32-bit for each memory when it is needed to be accessed by an ARM processor.

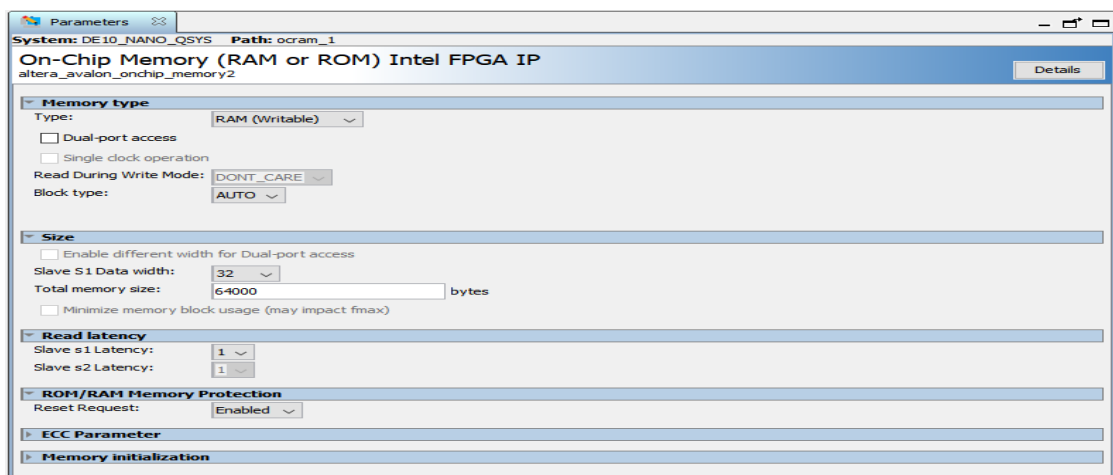


Figure 3.16 The Configuration of On-chip-Memory Controller

3.8.1.3 Clocking and Synchronization

A crystal oscillator has been used to feed the system clock on the development board. 50 MHz is the frequency of the oscillator. An Intellectual Property (IP) known as Clock Source keeps the 50 MHz clock that the Platform Designer needs, as shown in Figure 3.17.

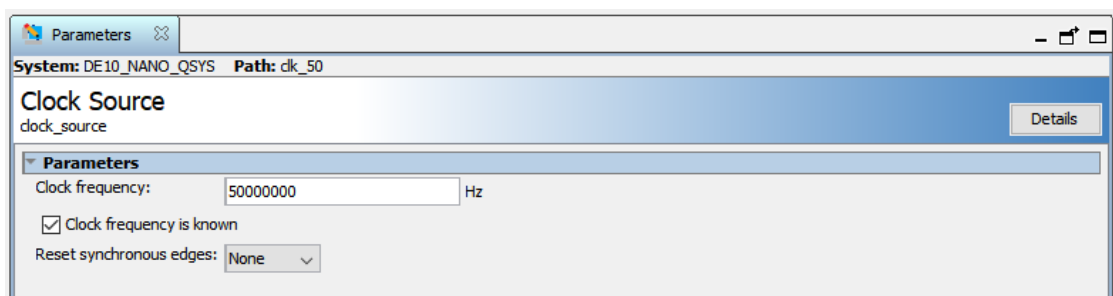


Figure 3.17 Clock source IP in Platform Designer

Phase Lock Loop (PLL) is an IP from Platform Designer that is utilized to amplify the clock. The clock value could be divided or multiplied based on the designer's requirement. The proposed system used a 50 MHz of clock frequency. The configuration of the PLL is presented in Figure 3.18.

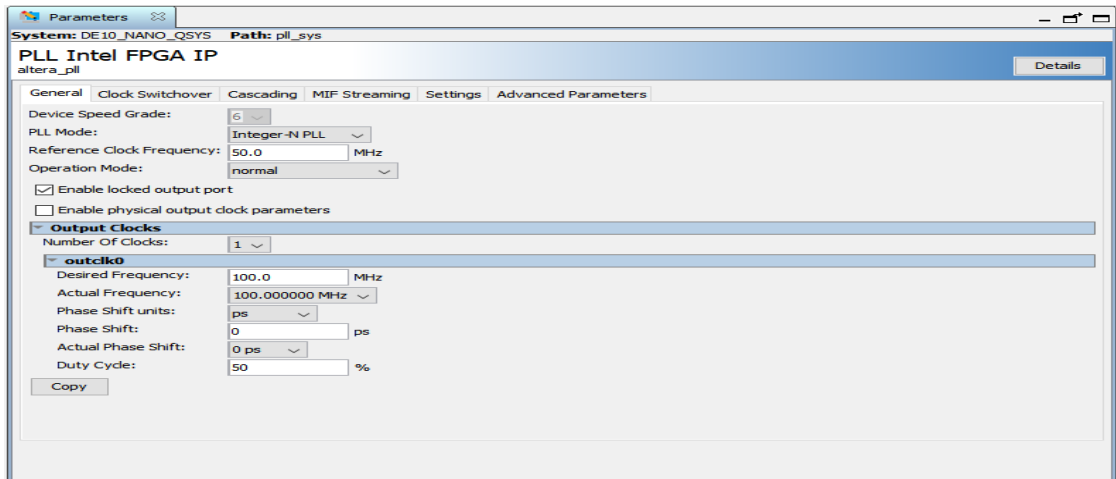


Figure 3.18 PPL Configuration in Platform Designer

3.8.1.4 Data Transfer Protocols

The proposed SWQS design utilized an I2C data transfer protocol communicated with the Analog-to-Digital Converter (ADC) controller as the three used sensors provide analog signals that needed to be converted to digital signals using an ADC (LTC2308). There is an 8-pin analog input that is connected to ADC. LTC2308 is 8-channel, 12-bit ADC that provides data with low noise. These eight input signals are connected to a 2x5 header, as shown in Figure 3.19. The FPGA chip will read the associated register in the converter using the serial interface and then translate the data into a voltage value displayed on Nios II's console.

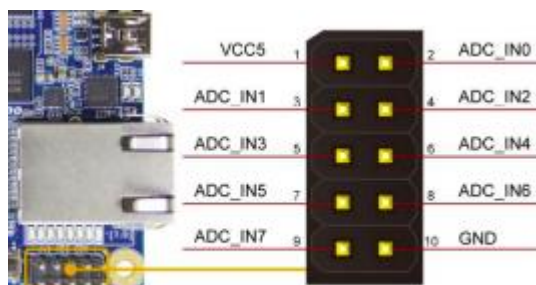


Figure 3.19 ADC Input Signals

3.8.1.5 System Peripherals

Three controllers have been used for each SWQs core. The first controller is the system timer, which is set to 1ms with 64-bits of the size of the counter, as exhibited in Figure 3.20. In addition, the readable snapshot feature is enabled to read the data time samples when the data is collected from the sensors.

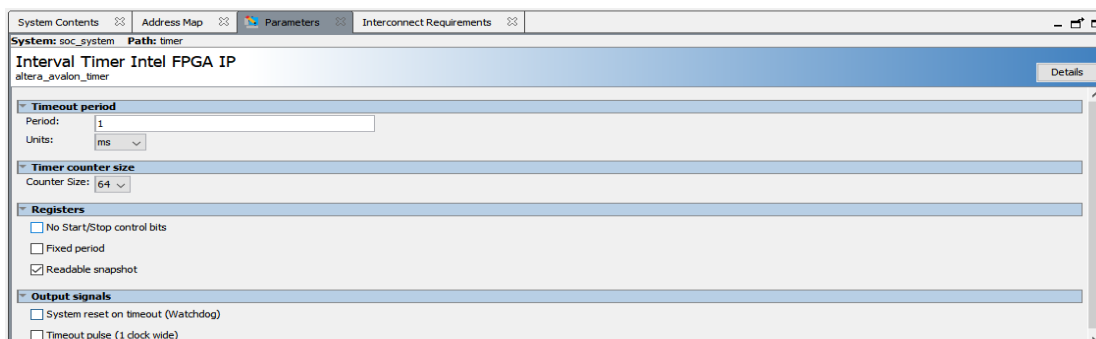


Figure 3.20 System Timer IP Configurations in Platform Designer

The second controller is the Peripheral Input/output (PIO). In contrast, the third controller is the Modular Scatter-Gather Direct Memory Access (mSGDMA) controller IP that is utilized for data transferring from FPGA to DDR3 of the ARM processor through the bridge of Avalon SDRAM. Reducing the data overhead on the Nios II processor is the main purpose of using the mSGDMA controller. Nios II core 0 handles the software configuration of this IP, as shown in Figure 3.21.

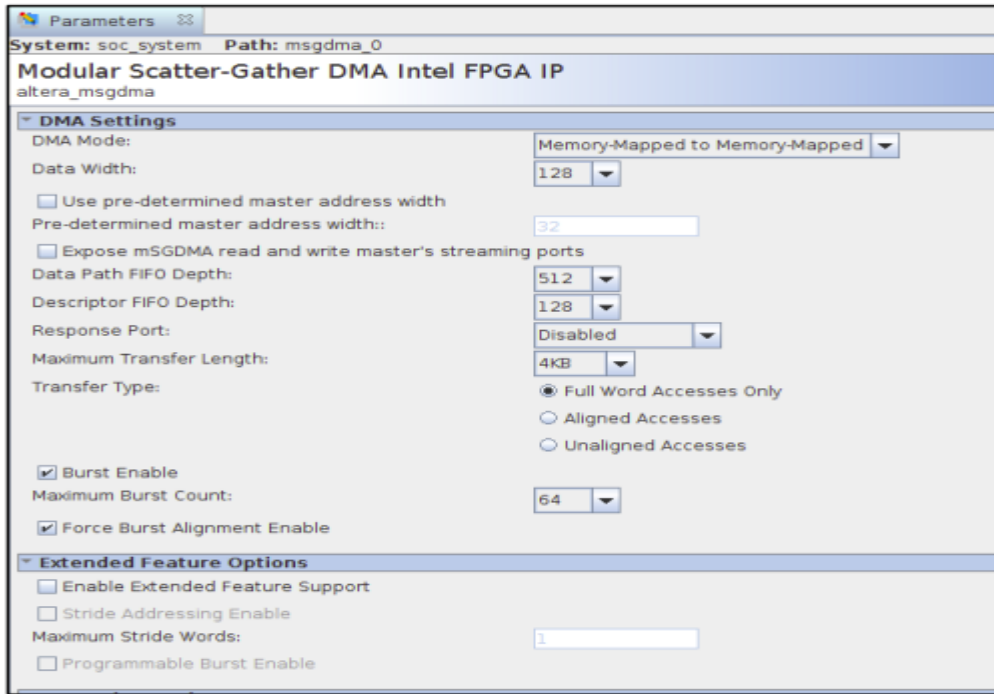


Figure 3.21 mSGDMA Controller IP Configurations

Mutex is a peripheral system used in SWQS design. This IP is needed to control the access of shared peripherals and memory among the cores in a multi-core system. Only one master at a time can access the shared peripheral. Therefore, this IP needs software configurations. However, there is no need for hardware configuration. Figure 3.22 illustrates the configuration of mutex in Platform Designer.

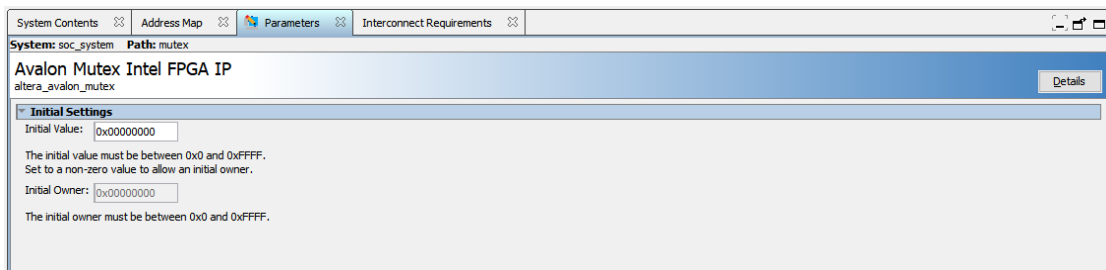


Figure 3.22 Mutex configuration in Platform Designer

Joint Test Action Group (JTAG) IP is used to access each core individually when debugging is needed. The UART IP can be used only in debugging process, and it will be idle if there is no debug command sent by the Nios II controller. There is no software or hardware configuration for this IP. Figure 3.23 presents the JTAG configuration.

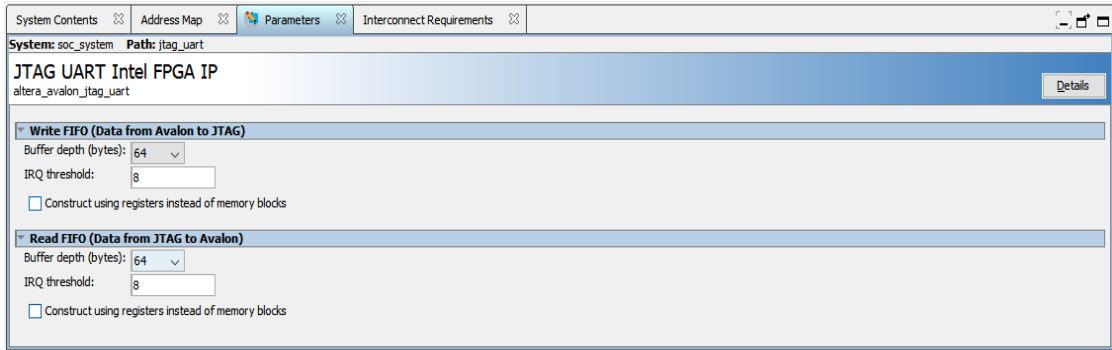


Figure 3.23 The Configuration of JTAG IP in Platform Designer

In addition, System ID Peripheral has been added, a read-only device that gives the systems in Platform Designer a specific and unique ID. Nios II processor utilizes the system ID IP to check whether the executable program was compiled targeting the image of the actual hardware configured in the target FPGA. However, if the system ID does not match the expected executable ID, it will not be executed correctly. Furthermore, there are two 32-bit registers in the core interface. Figure 3.24 demonstrates the configuration of the system ID core.

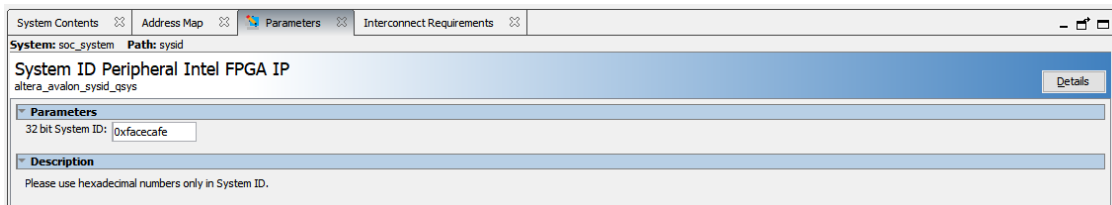


Figure 3.24 The Configuration of System ID Core in Platform Designer

Last but not least, an ADC controller has been utilized to give the ability to interface between the Nios II core and ADC. This is because all used sensors like pH, TDS, and turbidity provide analog data that should have a converter to read by FPGA. Therefore, this IP core is used instead of using a converter chip. It can control all the needed digital signals from and to the ADC controller. Additionally, it provides a memory-mapped register interface to read the ADC values.

3.8.1.6 Bridges

There are two kinds of FPGA internal bridges that are used in the proposed design, which are the Avalon Memory-Mapped Pipeline Bridge, and the second one is the Address Span Extender (ASE) bridge. The main purpose of using Avalon Bridge is to handle more than one slave for one or more multiple masters with the feature of the pipeline. Therefore, this bridge will improve the overall system performance and reduce the bus's overhead. The Avalon Bridge configuration can be seen in Figure 3.25. Since the Nios II processor core is a 32-bit processor, the Avalon Bridge data bus is set to 32-bit as well. The second bridge is the ASE bridge, which extends the memory address view of the Nios II processor and makes it 64-bit memory. Other than that, this bridge will provide the ability to write/read data from/ to DDR3 system memory. Figure 3.26 shows the ASE configuration.

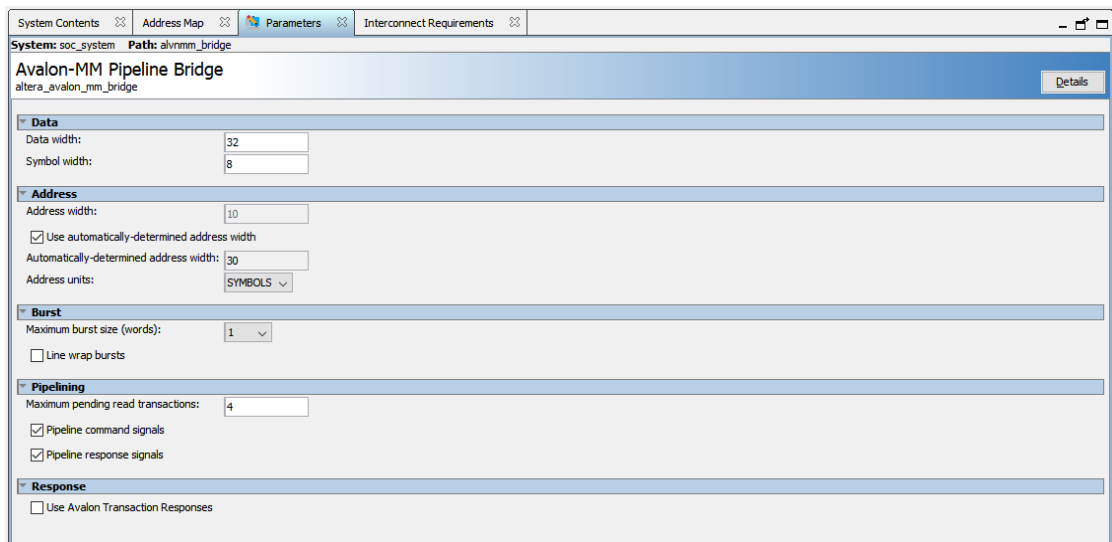


Figure 3.25 Avalon Memory Mapped Controller IP

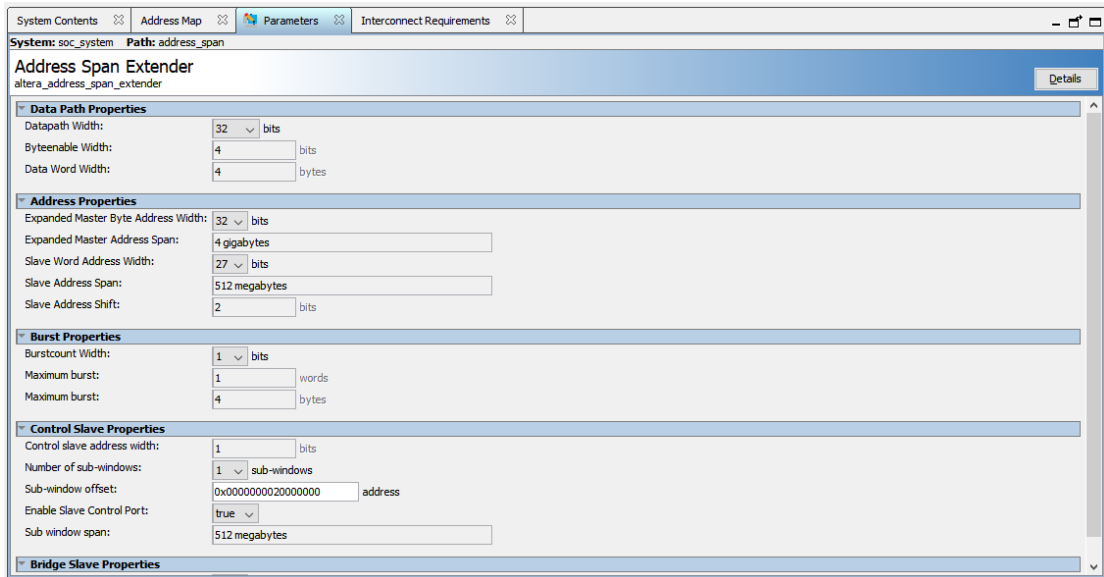


Figure 3.26 Address Span Extender IP

3.8.1.7 Pin Assignment

Pin assignment can be done using the Pin Planner tool to connect the pins inside the FPGA chip with the pins outside. Pins must be defined; otherwise, Quartus software will assign the pins randomly. Pin assignment can be seen in Figure 3.27.

Top View - Wire Bond Cyclone V - 5CSEBA6U2317

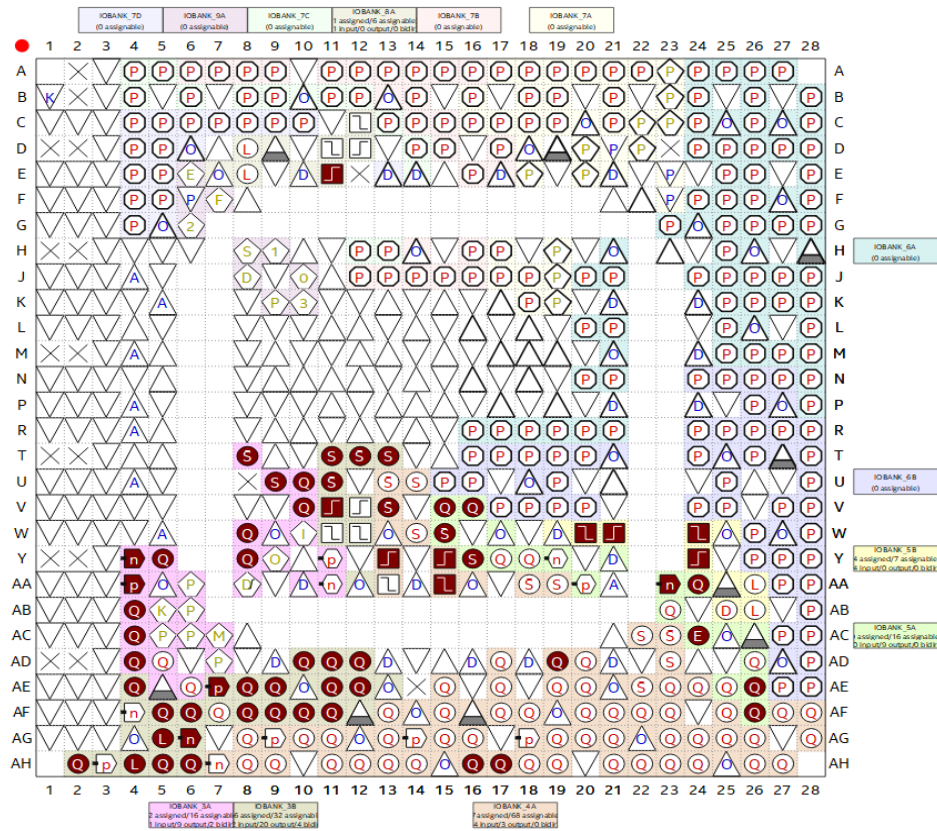


Figure 3.27 Top View of Pin Assignment

3.8.1.8 Synthesis Report

The synthesis report displays detailed synthesis results for each partition in the current project revision. It shows summary information about syntheses, such as the status, date, software version, entity name, device family, timing model status, and various types of logic utilization. From Figure 3.27, it can be observed that the full design has utilized 6313. In contrast, around 57% of the available on-chip internal memory was utilized as local memory for each processor core. More processor cores can be added to this design based on design requirements. Figure 3.28 presents the synthesis report of the proposed design.

Analysis & Synthesis Summary	
<input type="text" value="Filter"/>	
Analysis & Synthesis Status	Successful - Wed Mar 23 02:43:50 2022
Quartus Prime Version	18.1.0 Build 625 09/12/2018 SJ Lite Edition
Revision Name	DE10_NANO_SoC_GHRD
Top-level Entity Name	top_level
Family	Cyclone V
Logic utilization (in ALMs)	N/A
Total registers	6313
Total pins	179
Total virtual pins	0
Total block memory bits	4,331,776
Total DSP Blocks	0
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	1
Total DLLs	1

Figure 3.28 Synthesis Report of the Design

3.8.2 The Software Design Flow of SWQS

3.8.2.1 Firmware Development Flow

The developer can begin with the software flow once the hardware flow is done. The software mainly focuses on the sensors' programming and creating the Linux application where the SWQS data is read. In addition, the software is used to communicate the ARM terminal with the Raspberry pi terminal for external access. The software flow starts with the generating of the .elf file, which is the firmware of the processor. Note that firmware is the embedded code of each water quality sensor that has been run on the cores designed using the Platform Designer tool. In addition, creating firmware for each sensor was utilized to make each core process the data from its firmware instead of making all cores deal with one firmware. For instance, firmware 1 is responsible for pH sensor code; core 2 only could have access to pH firmware instead of making one firmware for pH, TDS, and turbidity core. Therefore, the data will be read in parallel instead of in series. The processor core memory stores the .elf file once it is generated. Figure 3.29 presents the complete software flow of the design.

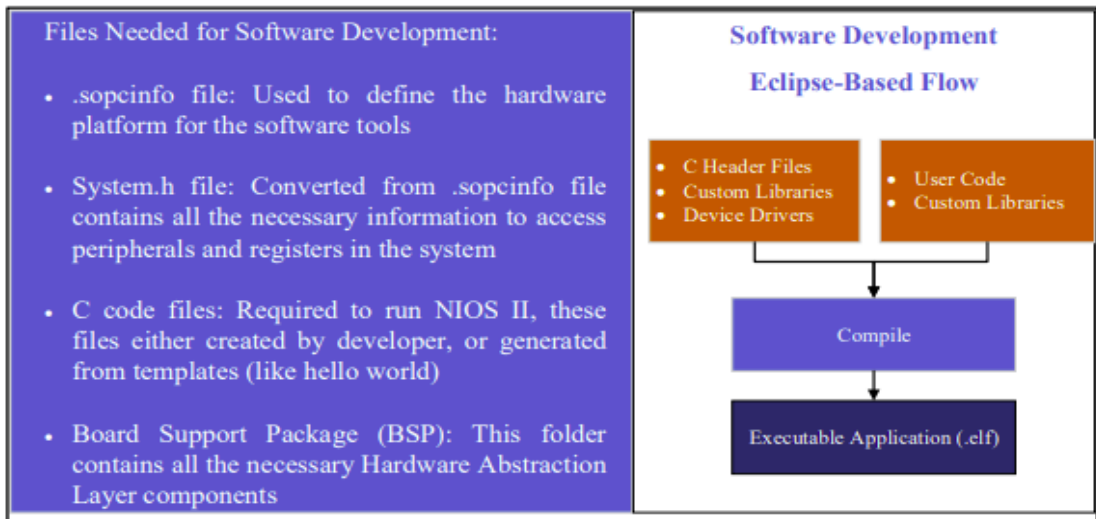


Figure 3.29 Nios II Software Development Flow

Once the Platform Designer is generated, a .sopcinfo file will be generated, which is utilized and is needed by the Nios II eclipse tool to generate the Board Support Package (BSP). Note that BSP is a file package containing Hardware Abstraction Layer (HAL), system calls, device drivers, and system header files. The developer could write the C code based on the design and compile it when the Nios II is generated. After the compilation process, a .elf file will be created that can be uploaded to the Nios II memory. Both hardware and software flows are needed to complete the FPGA design. Figure 3.30 demonstrates the entire FPGA process flow.



Figure 3.30 The Entire FPGA Development Flow

3.8.2.2 The Linux Application Development Flow of SWQS

When the hardware and firmware development flow is completed, the flow of Linux application development begins. Any software development language, such as C, C++, Python, Java, etc., can be used to design the target application. For example, the SWQS application is developed using the C++ programming language in this research.

The memory management stage starts developing, where the memory address is defined. Figure 3.31 presents the SWQS application memory layout.

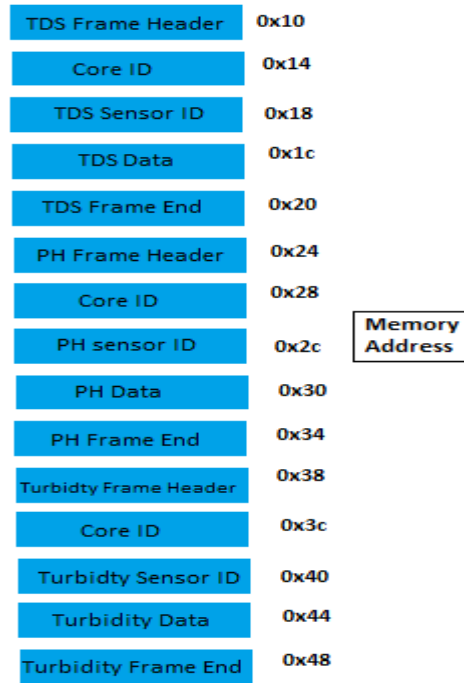


Figure 3.31 SWQS Application Memory Layout

The next step is to define the AXI bridge address and make it (0xff200000). Subsequently, the hardware registers span of Intel that is (0x40000000) is going to be added with the AXI bridge address. Figure 3.32 shows the offset of each component in the system. Finally, the target address, the result, will be used to access the FPGA proposed design registers by adding each register offset, as established in equation (3.1).

$$\text{Target address} = \text{Bridge offset} + \text{Hardware offset.} \tag{3.1}$$

LED Base	0x10040
Push Button Base	0x100c0
Shared Memory Base	0x0
Soft Reset Base	0x11a0
System ID Base	0x10000
FPGA RAM Offset	0x20000000

Figure 3.32 FPGA Components Base Addresses

The next stage is creating a virtual memory in any Linux-based application because Linux has a production memory that prevents direct access to an address in the system. Without virtual memory, segmentation faults can be caused. A Linux system call is used to achieve this known as `mmap()`, as shown below:

```
virtual_base = ( unsigned long *) mmap( NULL, HW_REGS_SPAN, ( PROT_READ
| PROT_WRITE ), MAP_SHARED, fd, ALT_AXI_FPGASLVS_OFST).
```

The virtual base represents the base memory address that can be utilized with any offset so that it can be accessed. Therefore, when the developer needs access to the shared memory, adding the virtual base to the shared memory base is compulsory.

After the address configuration, the header of the frame for each dataset will be checked. When the frame header exists, the processor can read the data after decoding the frame data. However, when the frame header is not existing, the data will be rejected by the processor as it might be invalid or corrupted. In the end, the collected data will be printed out, and it can be shown in a real-time environment.

3.9 THE EMBEDDED LINUX DESIGN FLOW OF SWQS

Embedded Linux is a complete distribution operating system deployed to run embedded devices such as the Internet of Things (IoT), tablets, smartphones, and Personal Assistance Devices (Salvador & Angolini, 2014). The kernel of Linux can be run on different processor architectures and SoC, such as Intel, ARM, and AMD. These five components included in the embedded Linux system are known as the Bootloader, Rootfs, kernel, services, and application.

3.9.1 Bootloader Compilation

A small software or mini operating system known as a Bootloader must be run before starting the Linux environment. The component of the system, such as Ethernet, memory, and the FPGA configuration, is initiated by the Bootloader, which gives the ability to start the loading and running Linux kernel environment. Note that the Bootloader includes platform-dependent services and responsibilities and some independent features.

U-boot is the Bootloader used in the proposed design because this Bootloader support Intel FPGA-SoC devices (Intel Corporation, 2014). Many development platforms have used U-Boot to support architectures, such as MIPS, ARM, AVR32, Microblaze, Nios, x86, and 68K. The user can use the U-boot to give data interactively as it has a shell. In addition, it supports scripting, and it is distributed under the GPLv2 license.

Figure 3.33 presents the Bootloader Generation Flow. Embedded Software tools and Intel FPGA-SoC software (SoC EDS) tools are used to generate the Bootloader. The Handoff folder is created after compiling the Quartus Prime project. The Handoff folder includes the hardware processor system's connection and components. SoC EDS and Handoff folder is used to generate the U-boot and pre-loader. When the Bootloader is generated, the "make" tool is used to create the binary files of the Bootloader.

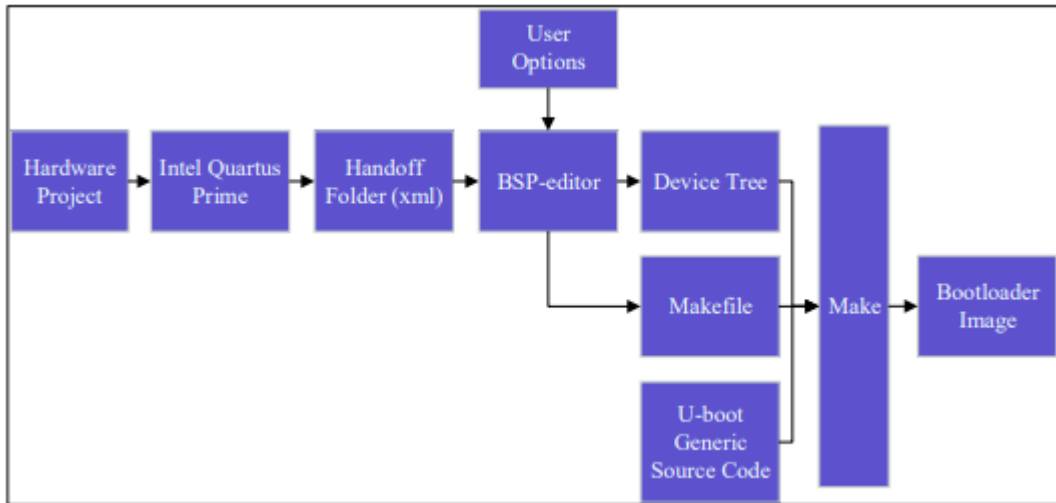


Figure 3.33 U-Boot Development Flow

3.9.2 Root File-system Creation

In this design, the default Rootfs of the golden top reference design from TERASIC can be utilized as it is. The golden top file is the top-level design file which contains the design, pin assignment, and I/O connection for each defined pin. No changing or modification is needed to be done. Other than that, DE10 nano SoC development kit Rootfs can be found and downloaded from the TERASIC website.

3.10 ADDING NEW CORE TO SWQS DESIGN

It is easy to add the core of Nios II with no more modifications required to the design. In the Platform Designer, all the connections of the Nios II core can be made internally, as shown in Figure 3.34.

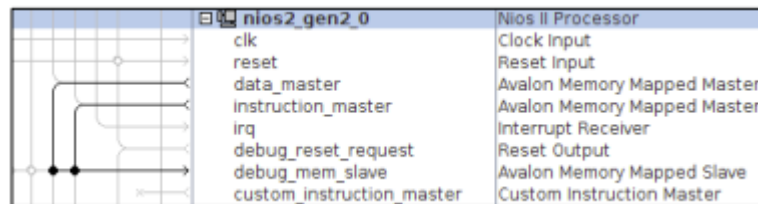


Figure 3.34 Nios II Core Connections

The clock source will then feed the clock connection; furthermore, the reset connection should be communicated with the rest of the master source. Avalon Memory-Mapped bus interface is the data master. All the peripheral devices must be connected to the Nios II processor using this master. When developing the firmware in the system.h file, the Nios II processor should have the connection of any IP or peripheral to the data master. Note that Avalon Memory-Mapped is the instruction master responsible for the firmware instructions transactions. This master should be communicated to the memory of the firmware only. The Interrupt Request (IRQ) connection represents the interrupt master. When any interrupt source exists in any IP or peripheral device connected to the processor by the data master, it can then be shown in the IRQ master. When this additional procedure is done, the Platform Designer should generate the .sopcinfo and .qsys files required for both software and hardware flows.

3.11 SUMMARY

This chapter showed the technical guidelines for Heterogeneous System Architecture (HSA). The chapter contained the flow design of the FPGA at both software and hardware levels. Furthermore, the Linux development with all commands required to create a completed Linux image has been explained briefly in this chapter. In addition, the process of adding more cores to the proposed design was elaborated on in this chapter. This chapter presented the hardware design and how the sensors will be connected to the FPGA board. In addition, how the data will be transmitted to the ARM processor and why the ARM processor is used in the proposed design. Moreover, software domains were also elaborated on in this chapter. This illustrates the importance of using more than one core for each sensor.

CHAPTER FOUR

RESULT AND ANALYSIS

4.1 INTRODUCTION

In this chapter, the results of the proposed design at each level are illustrated in Section 4.1. The discussion of the results is elaborated in Section 4.2, while Section 4.3 shows a benchmark of the proposed design with previous works. Finally, a summary of the chapter is presented in Section 4.4.

4.2 RESULTS

This study has proposed a new Smart Water Quality System (SWQS) hardware data acquisition system design based on the heterogeneous Field Programmable Gate Array-System-on-Chip (FPGA-SoC) platform. The system levels are the design level, design verification level, prototype implementation, data validation, and system integration. There is an intersection between the level and its item that has specific information; for example, the design level has an action item that is the design flow of FPGA hardware. Subsequently, the operation item has the results of the Quartus project compilation. In the end, the output item implementation of the hardware on the FPGA board.

4.2.1 The Results of the Quartus Project Compilation

Quartus prime lite version, a free software, has been used to develop the proposed design of Nios II processors' systems on FPGA. The result of the system compilation can be shown in Figure 4.1. It can be demonstrated from the figure that the design has consumed 9% of the logic elements available for the Cyclone V SoC device, which is a tiny portion of the available logic elements. Local memory for each processor core has utilized 57% of the available On-Chip Internal Memory. More processor cores can be added to this design based on design requirements.

Flow Summary	
Flow Status	Successful - Sat May 21 23:38:53 2022
Quartus Prime Version	18.1.0 Build 625 09/12/2018 SJ Lite Edition
Revision Name	DE10_NANO_SoC_GHRD
Top-level Entity Name	top_level
Family	Cyclone V
Device	5CSEBA6U23I7
Timing Models	Final
Logic utilization (in ALMs)	3,829 / 41,910 (9 %)
Total registers	5251
Total pins	172 / 314 (55 %)
Total virtual pins	0
Total block memory bits	3,235,072 / 5,662,720 (57 %)
Total DSP Blocks	0 / 112 (0 %)
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0

Figure 4.1 The Result of the System Compilation

4.2.2 The Implementation of Prototype

The prototype was tested on different types of liquids for real-time testing. The prototype design can be found in Figure 4.2. The following components were used for this experiment: Raspberry Pi 3 development kit, DE10 Nano FPGA-SoC development kit, pH sensor, TDS sensor, turbidity sensor, as well as a few wires and cables.

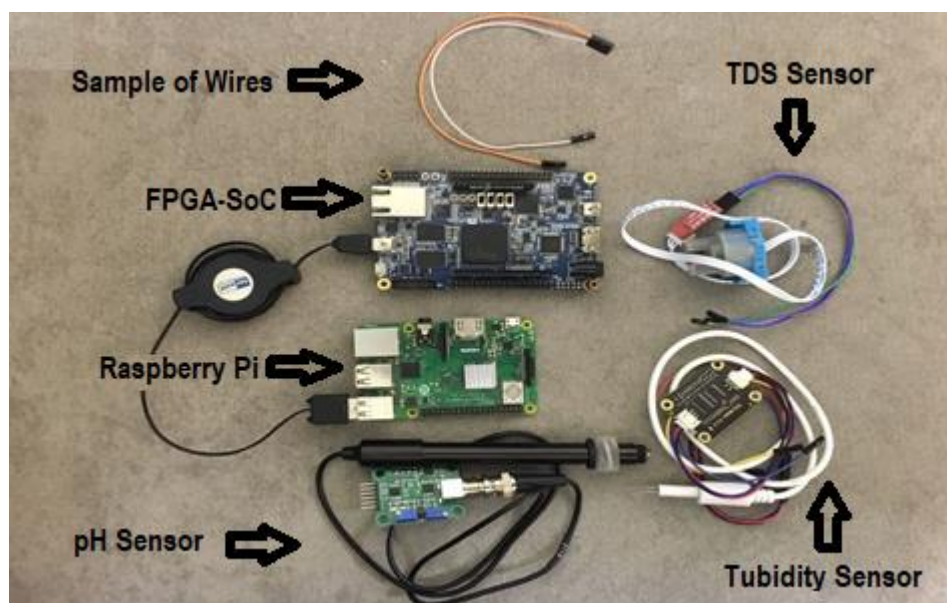


Figure 4.2 System Setup Components

Figure 4.3 illustrates the proposed design block diagram. Firstly, it presents the pH sensor, Total Dissolved Solids (TDS) sensor, and turbidity sensor are analog sensors. Therefore, the sensors need to be connected to ADC with an internal reference circuit and the sample-and-hold circuit to decrease the noise that might affect the data stability. The input signals of the pH sensor, TDS sensor, and turbidity sensor have been connected to the ADC_IN0 pin, ADC_IN1 pin, and ADC_IN3 pin, respectively. Once the data is collected and transferred to SoC, it can be viewed using the Raspberry Pi by accessing the Linux application of SoC. Therefore, USB Mini-B was utilized to connect Raspberry Pi with SoC. In addition, the Linux image has been burned on an SD card which can be used to display the data collected from the sensors on the screen connected to Raspberry Pi. Finally, each sensor must be connected to a 5V and ground pin.

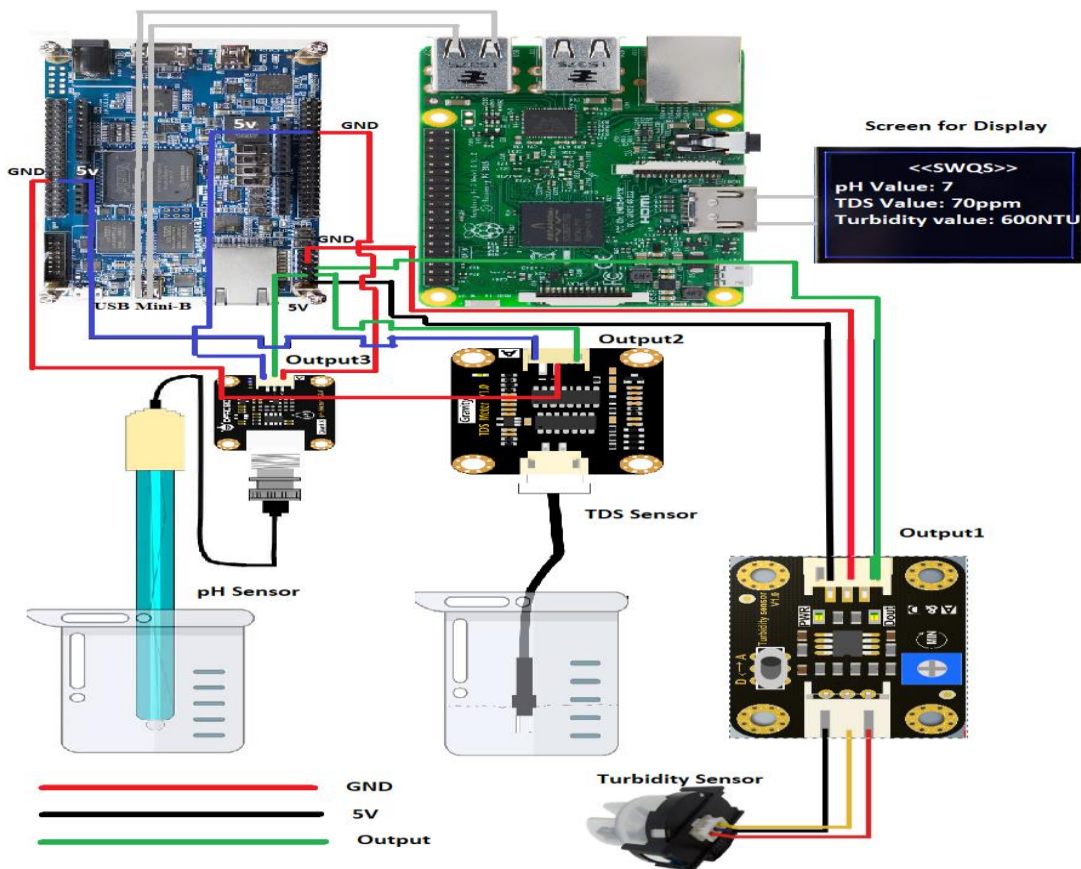


Figure 4.3 Design Block Diagram

The above figure demonstrates that the Raspberry Pi has another component, Liquid-Crystal Display (LCD), used in this experiment to avoid any use of a laptop or

PC and make the system portable and can be functioned using a power bank. The LCD is utilized to access both Raspberry Pi and FPGA terminals. The Raspberry Pi board was used for system augmentations. These augmentations are LCD, keyboard, and mouse. These augments can be considered proof of system configurability and the ability to integrate easily with any system or augment.

4.2.3 Design Verification Method

There are nine tests have been done on the proposed design for the verification process. They are:

- i. Core 0 data packetize test.
- ii. Core 0 DMA run test.
- iii. Core 1 pH test.
- iv. Core 1 data transfer test.
- v. Core 2 TDS test.
- vi. Core 2 data transfer test.
- vii. Core 3 turbidity test.
- viii. Core 3 data transfer test.
- ix. Mutex functionality test.

The data was printed out on the console to test the functionality of each core before sending the data to the Linux application on the ARM terminal. This step was done for 60 seconds for each sensor. It is essential to trace the errors on each embedded C code to avoid the debug when the system gets more complicated.

Figure 4.4 presents the block diagram of the core 1 pH test as well as the core 0 data transfer test. Core 1 was used to read data from pH sensors and send it to a shared memory. However, the shared memory is available once the mutual exclusion (mutex) is unlocked. When the system is idle, the data will be written to shared memory.

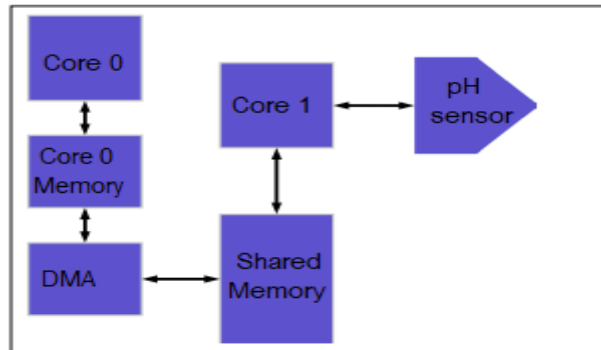


Figure 4.4 Core 1 pH Data Collection and Data Transfer Test Block Diagram

Figure 4.5 shows the core 1 pH test and the data transfer test result. The output of core 1 indicates that elaborate pH functionality and pH data transfer have been done successfully. Other than that, pH results on the pure water were measured to check the functionality of the system before sending it to the Linux application. It can be shown from the figure below that the pH value varies from 6.85 to 7.03, which is an acceptable range.

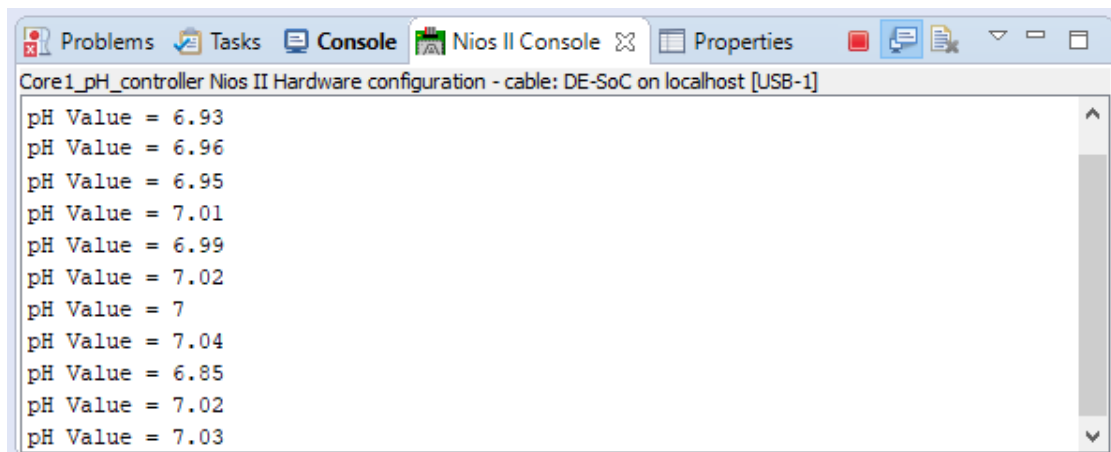


Figure 4.5 Readings of pH Sensor

Figure 4.6 illustrates the core 2 TDS data collection and transfer test diagram. The below diagram presents all the needed components for success in core 2 tests. Core 2 was utilized to get TDS data and write it to shared memory. On the other hand, the shared memory is available once the mutex is unlocked. When the mutex is available, the data will be written to the shared memory.

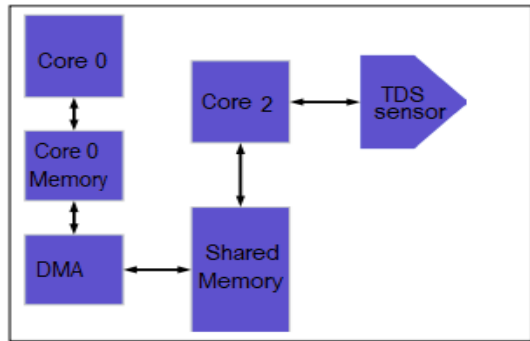


Figure 4.6 The Block Diagram of Core 2 TDS Data Collection and Data Transfer Test

The TDS sensor measured the TDS value of pure water, which was between 63.54 ppm and 71.34 ppm. However, when adding salt to the water, the value increased approximately between 1769 ppm and 1874 ppm. The results have been printed out by core 2. Figure 4.7 shows the TDS readings of the water with more salt.

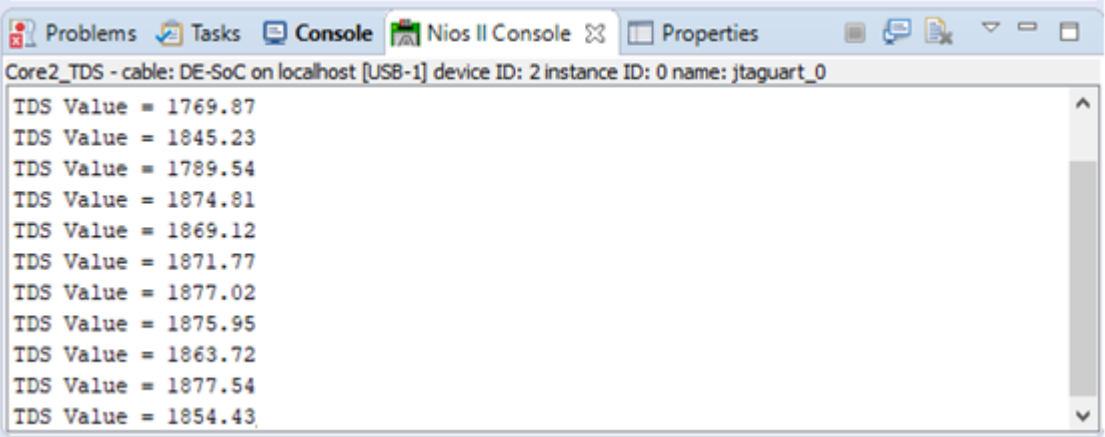


Figure 4.7 Readings of TDS Sensor

Figure 4.8 presents the block diagram of core 3 turbidity data collection as well as the tests of the data transfer. Core 0 has been utilized by core 3 to test the data transfer. In addition, core 3 has been connected to the turbidity sensor to achieve a data collection test and to prove the functionality of the sensor.

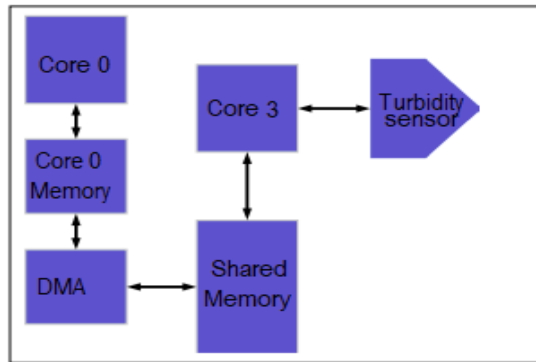


Figure 4.8 The Block Diagram of Core 3 Turbidity Data Collection and Data Transfer Test

Core 3 output can be observed in below Figure 4.9. The data of the turbidity sensor has been collected by core 3 and transferred through DMA to core 0. To check the functionality of the turbidity sensor, little dust was added to the pure water. Therefore, the turbidity value increased to reach between almost 1060 to 1180 mNTU.

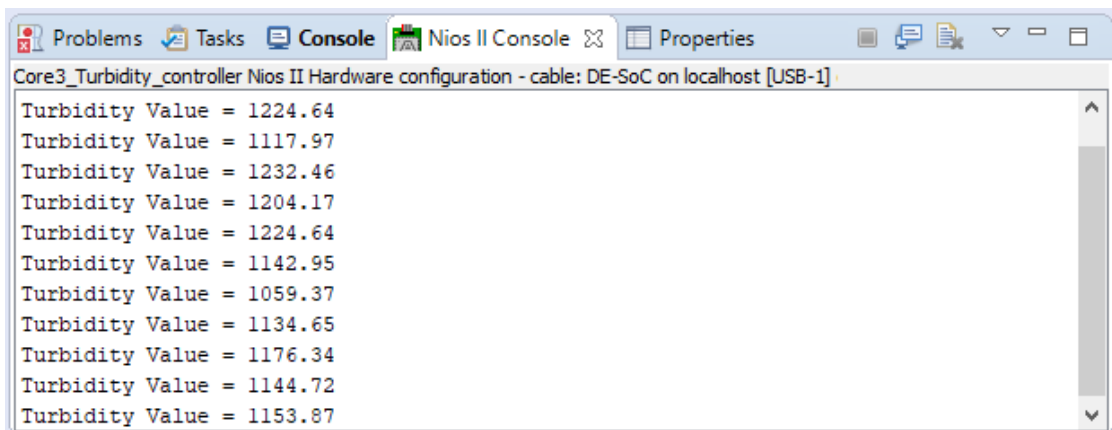


Figure 4.9 Readings of Turbidity Sensor

Once the test of each core is done separately, the next and final step is to send the data that was gotten from each core to the ARM terminal. The ARM terminal will read the data in parallel from each core to prevent latency. The experiment was done on pure water, lemon juice, and milk to test the pH sensor. Meanwhile, to test the TDS sensor, pure water and water with salt and the turbidity sensor were tested on pure water, water with little dust, and water with more dust. Note that the system has been tested successfully with good readings. The test was done several times on different time

periods to make sure the system was functioning well. In addition, the line graphs below for each sensor were measured for 60 seconds only.

The pH value was almost 7 for pure water; however, it decreased when measuring the pH of lemon juice and acidic liquid. Nevertheless, when measuring the pH value of milk, the values were above an alkaline liquid. Figures 4.10 and 4.11 present the installation of a pH sensor to measure different liquids.

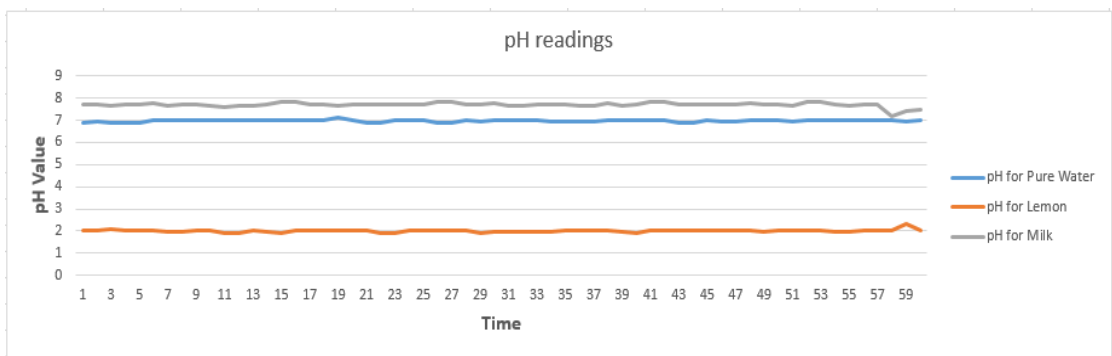


Figure 4.10 Line Graph of pH Values of Pure Water, Lemon Juice, and Milk

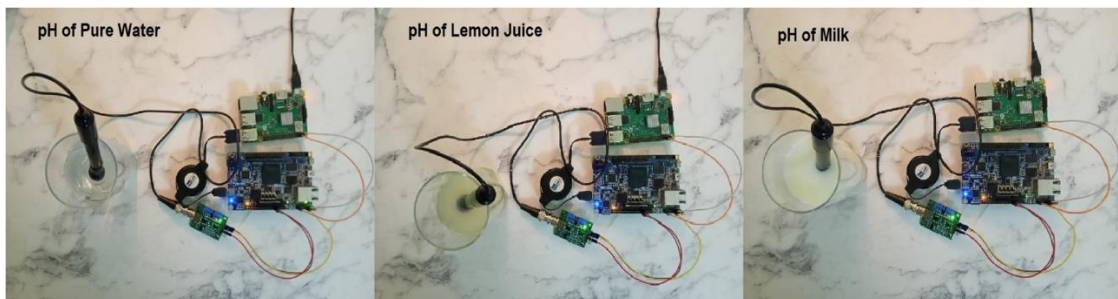


Figure 4.11 Testing the pH Sensor

For TDS, a sensor has been used to measure it. Two liquids, pure and salted water, have been utilized to measure TDS. Results show the purer the water, the less TDS value. TDS value is measured by ppm, which is parts per million. Pure water values were between 60 ppm and 70 ppm. However, adding salt to the water will increase the TDS values to nearly 1700 ppm, as presented in Figure 4.12. However, more particles in the water mean a great TDS value; therefore, it is crucial to the

parameter that needs to be measured to distinguish pure water from a not pure one. Figure 4.13 illustrates the real installation of the turbidity sensor.

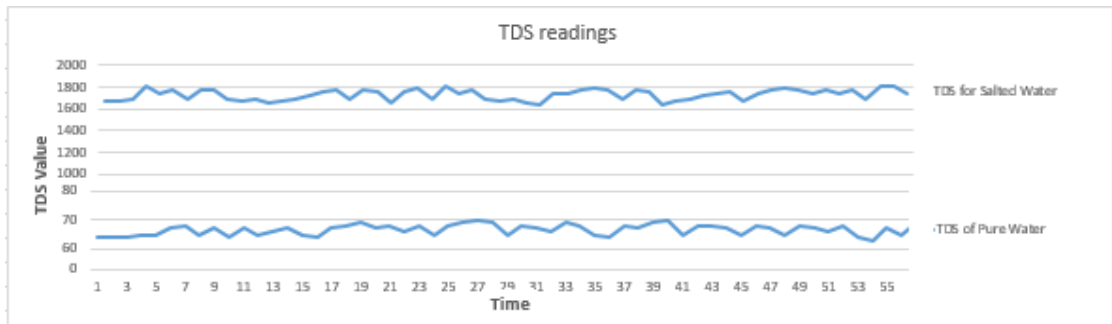


Figure 4.12 Line graph of TDS Values of Pure Water and Water with Salt

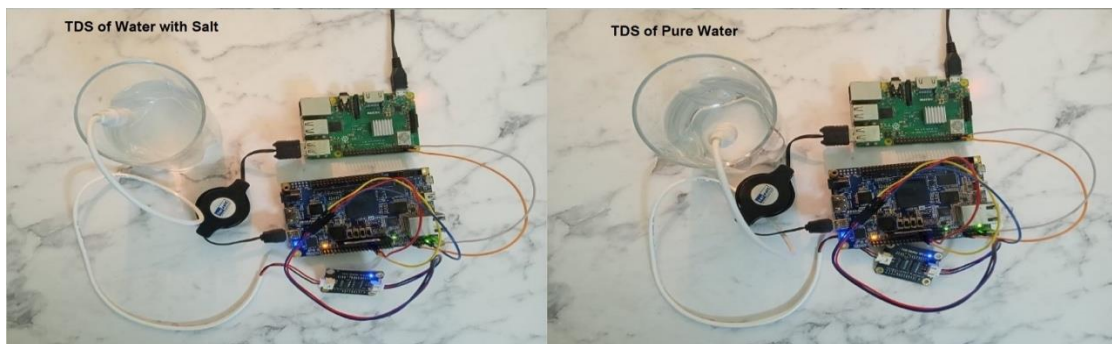


Figure 4.13 Testing of the TDS Sensor

Turbidity is the measure of the relative clarity of a liquid. It is an optical characteristic of water and measures the amount of light scattered by material in the water when a light is shined through the water sample. The higher the intensity of scattered light, the higher the turbidity. The turbidity is measured by Nephelometric Turbidity Units (NTU). Pure water showed a value of nearly 600 mNTU to 800mNTU. Nevertheless, the water with some dust showed more than 1100 mNTU, as shown in Figure 4.14. The measurements show that the more the turbidity value, the more particles in it, making it unsafe to drink. Figure 4.15 demonstrates the real installation of the turbidity sensor.

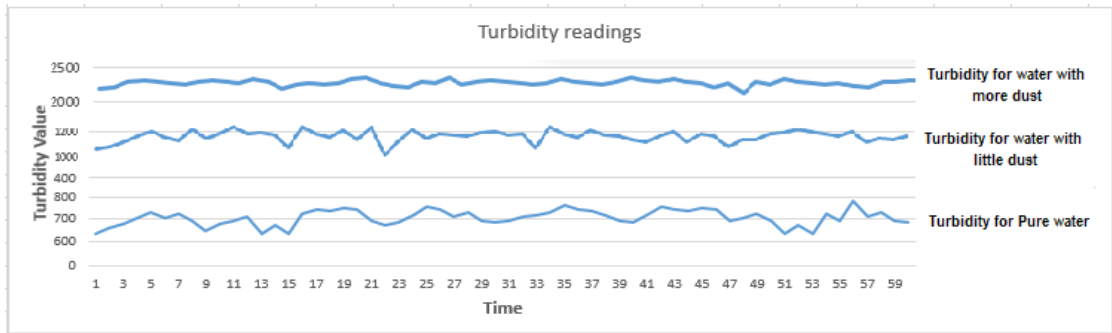


Figure 4.14 Line graph of Turbidity Values of Pure Water, Water with Little Dust, and Water with More Dust



Figure 4.15 Testing the Turbidity Sensor

4.2.4 The Results of the SWQS Linux-Based Application

SWQS Linux-based application was developed on Linux kernel SystemCalls C programming. After compiling the executable format of this application, it was converted to a Linux Terminal command. Therefore, the (-h) option must be familiar with the command arguments to run the Linux-based application.

All the execution possibilities of the Linux application have been tested successfully to make validation to the functionality of the application. The behavior of the system in case of the system cores controlling, system status monitoring, application behavior in case of the wrong command is used, and finally, printing usage message in case the user types of false arguments.

For instance, running the command “swqs -r 0”, the expectation is to run all the system cores. However, if the user specifies one of the cores of the sensors like “swqs -r 3”, core 0 and core 3 will be executed, showing the turbidity sensor’s data.

The SWQS Linux application was tested in a real-time environment to control the cores of processing elements on FPGA, decode the message sent from core 0, and finally show the data collected from sensors on the Linux Terminal, as shown in Figure 4.16. After integrating the sensors with the cores, the above test verifies the system's functionality.

```

root@de10-nano:~# ./APP
NIO5 II Reset Status = 0
System ID = 0xFACECAFE
Detecting Sensors on system
Sensors = TDS          pH          Turbidity
Core ID = 1
TDS_VALUE = 66.96
*****
Core ID = 2
Turbidity = 652
*****
Core ID = 3
pH = 6.96
*****
Core ID = 1
TDS_VALUE = 68.21
*****
Core ID = 2
Turbidity = 649
*****
Core ID = 3
pH = 7.02
*****
Core ID = 1
TDS_VALUE = 70.03
*****
Core ID = 2
Turbidity = 649
*****
Core ID = 3
pH = 7.01
*****
Core ID = 1
TDS_VALUE = 69.33
*****
Core ID = 2
Turbidity = 655
*****
Core ID = 3
pH = 6.99
*****
Core ID = 1
TDS_VALUE = 69.63
*****
Core ID = 2
Turbidity = 656
*****
Core ID = 3
pH = 7.05
*****

```

Figure 4.16 SWQS Results on Linux-Based Application

Raspberry Pi has another component, LCD, used in this experiment to avoid using a laptop or PC and make the system portable and function using a power bank. The LCD is utilized to access both Raspberry Pi and FPGA terminals. Meanwhile, the Raspberry Pi board was used for system augmentations. These augmentations are LCD, keyboard, and mouse. These augments can be considered proof of system

configurability and the ability to integrate easily with any system or augment. The proposed water quality system was proved as a smart system since it incorporates functions of sensing and controlling a system to describe or analyze a situation. Other than that, the proposed system provides a wide space of flexibility for developers to design and develop their applications from hardware (FPGA platform) and software (SoC) perspectives. The proposed platform is not tied to any programming language. It has an embedded Linux running on an SoC sub-system to give full adaption with any programming language or artificial intelligence (AI) platforms like TensorFlow and Caffe. The data can be streamed to the AI platform in a real-time environment.

4.3 DISCUSSION

After getting all the results and testing the proposed SWQS, this section discussed the proposed system's novelty and uniqueness in terms of software and hardware design.

4.3.1 Motivations of the Middleware Layer

This research design has invented a new layer between software and hardware known as the middleware layer. Software services to the application layer are provided in the form of API. There are two layers in the system: software and hardware layers. However, more knowledge is needed in hardware design skills, including Very High-Speed Integrated Circuit Hardware Description Language (VHDL), Verilog, and Verilog system, design constraints, etc. Moving toward the hardware layer will raise the complexity of the design. Also, when moving toward the software layer, the developer requires more skills in developing the software that contains the operating system, services of the systems, programming languages, etc. The design limitations will grow when moving to hardware design because of the controllers, buses, interfaces, and system components. Nevertheless, these limitations will decrease as moving to the software layer because everything will be controlled on the software level.

The middleware layer will make the work of the developers easier since there is less access to the hardware, and most access will be done via the middleware layer. The proposed design has derived APIs, which will assist the application layer in accessing

the hardware and sensor layers with less complexity. The proposed heterogeneous design provides software and hardware engineers with an excellent opportunity to design SWQS applications.

4.3.2 System Flexibility

FPGA was utilized to design the proposed system, providing more design flexibility. The designer might face the most common issue: the limited number of input/output peripherals on most boards. This is a critical issue when multiple sensor interfaces are needed to design the system. FPGA is a programmable board. Therefore, the developer may design the interfaces, controllers, and peripherals using Intellectual Property (IP), a pre-made component available on FPGA. For example, Figure 4.17 presents the components and interfaces needed for a single core in the proposed SWQS application. However, it can be easily added when a new interface is required. In addition, the new interfaces could be wired with external FPGA pins.

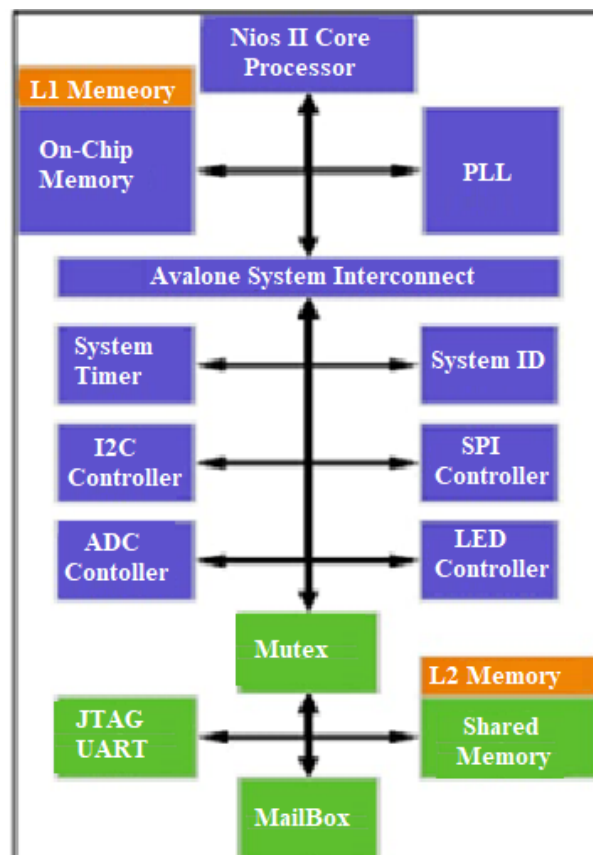


Figure 4.17 SWQS Proposed Single Design Core

4.3.3 Applications of the Design

The proposed design has been developed based on SWQS data acquisition and the system functionality. Achieving high-quality SWQS needs multiple sensors and sub-systems to be added to the design. Many applications can be extended to data processing, decision-making, performance analysis, machine learning, and deep learning. Using the embedded Linux operating system in the SWQS proposed system will simplify the application development. Note that any programming language could be used for developing the application.

4.4 SWQS DESIGN COMPARED TO PREVIOUS WORK

The full testing of the SWQS was done based on the testing plan, and the test has been done several times to ensure that it works well under different situations. However, it is strongly necessary to evaluate the SWQS design to ensure it is working well. SWQS design was evaluated by comparing it with previous work to prove that the SQWS design has increased the performance of monitoring the water quality. In addition, the proposed design will be benchmarked with (Myint et al., 2017), as both use the FPGA board.

The proposed design has applied Verilog language for the hardware design, which is more accessible than VHDL. It has been used by (Myint et al., 2017) as it is less complex and requires fewer coding lines. Other than that, Verilog is almost like the C language, which makes Verilog user-friendly (Digilant Blog, 2022).

In addition, both designs have utilized the Platform Designer tool to create the Nios II softcore, a configurable processor with an internal Central Processing Unit (CPU). Nevertheless, the proposed design used one Nios II processor for each to employ the option of parallelism, which is getting the data of all sensors simultaneously to decrease the time consumption. The proposed design used four cores. Core 0 is responsible for data synchronization, while core 1, core 2, as well as core 3 are responsible for collecting raw data from each sensor, calibrating the readings, segmenting the data in a format, and finally making it ready for the user application running on the ARM processor.

The proposed system used the Linux application to display the sensors' water quality data. The Linux application gives the ability to get the data easily on excel or connect with other software for machine learning or performance analysis. Nonetheless, (Myint et al., 2017) used eclipse Grafana software to display the data only.

The proposed prototype utilized a small screen for real-time displaying and saving the data in the SD card provided. In contrast, (Myint et al., 2017) used the XBee transmitter model to transfer the data remotely, which might be lost if the connection is disturbed.

4.5 SUMMARY

This chapter showed the proposed design's results, and the collected data from the system sensors were analyzed and discussed. Each test was done more than twice to check the system's functionality. In addition, this chapter discusses the application and results in terms of the middleware layer and the system's flexibility. In the end, the proposed system was compared to (Myint et al., 2017) in terms of functionality and results.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 CONCLUSION

Contaminated or polluted water is one of the leading causes of mortality. According to the World Health Organization (WHO), 2.2 billion people drank water without any safety-management services in 2017, with 144 million collecting water from untreated water bodies such as lakes, streams, and rivers (“2.1 Billion People Lack Safe Drinking Water at Home, More than Twice as Many Lack Safe Sanitation,” 2017). As a result, measuring water quality becomes an important issue. A conceptual layout of the project is given by stating the background, statement of the problem, and research objectives. A critical literature review is done to do the groundwork of what a Smart Water Quality System (SWQS) is to justify why this research is essential.

By the end of Chapter 2, objective one, which was investigating and analyzing the previous and related works on designing an SWQS and categorizing it in groups, is already achieved. This research methodology is divided into hardware and software design of the proposed SWQS using Heterogeneous System Architecture (HSA), a new computer platform infrastructure and associated software. This allows processors of different types and architectures to work efficiently and cooperatively in shared memory from a single source program.

The project proposed a new and novel SWQS hardware acquisition system design based on a heterogeneous platform using the System-on-Chip (SoC) and Field Programmable Gate Array (FPGA) platforms. The proposed design was tested from the hardware compilation and ended with the system integration testing. The design was utilized to provide an excellent data acquisition system validated using three water quality parameter sensors: pH sensor, Total Dissolved Solids (TDS) sensor, and turbidity sensor. The hardware design was implemented successfully on FPGA to take advantage of the flexibility and configurability of FPGA; however, the software application was implemented on the SoC platform to take advantage of the performance and programmability of the SoC platform. Hence, Quartus Prime was used to compile

the hardware design. Subsequently, we implement the prototype for real-time testing. The final step is to connect the prototype to a Raspberry Pi processor, which was connected to a screen to display the data. The system was portable, and it was powered using a power bank. Therefore, the design can measure water quality anywhere without worrying about the power source. The proposed design can be used as a heterogeneous multi-core system for many applications, one of which is the SWQS data acquisition system.

5.2 FUTURE WORK

This study showed the initial research challenges and provided an excellent solution to develop and implement an SWQS system in a heterogeneous computing platform based on FPGA-SoC architectures. Some future recommendations that can be used to improve the system are listed below:

- **System integration:** performance is the main requirement for the embedded system. Further design cycles must be considered in future studies, especially data integration, analysis, and decision-making. In addition, power consumption and the associated thermal management are essential issues that must be considered.
- **Application development:** the proposed design is a Linux-based application to control and monitor the hardware system. In addition, the data were read by the proposed design in real-time. For SWQS, many previous studies employed different sensors, and how to add them to the proposed design is a fundamental issue that needs to be focused on in the future. Additionally, several artificial intelligence (AI) could be integrated with the proposed system for many AI applications in terms of object detection, segmentation, and classification.
- **Optimization and analysis:** optimization and performance analysis are considered important issues that need more work to improve the overall system performance. For instance, it can be done by optimizing other indices like power, energy, or both to enhance the heterogeneous system performance.

- **Involve other sensors:** the main aim of the proposed design was to prove that the heterogeneous computing platform FPGA-SoC could implement SWQS with high performance and efficiency. Therefore, only three sensors have been used. However, more water quality sensors could be integrated into the proposed SWQS to measure other important water quality parameters.

REFERENCES

- 2.1 billion people lack safe drinking water at home, more than twice as many lack safe sanitation. (2017, July 12).
- Aaruththiran, M., Yujia, Z., & Bagherian, M. A. (2019). *Smartphone-based Real-Time Water Quality Monitoring System*. The University of Nottingham.
- AN 311: Standard Cell ASIC to FPGA Design Methodology and Guidelines. (2009, April). *Intel*. Altera Corporation.
- Besta, M., Stanojevic, D., Licht, J. D. F., Ben-Nun, T., & Hoefler, T. (2019). *Graph Processing on FPGAs: Taxonomy, Survey, Challenges*.
- Billah, M. M., Yusof, Z. M., Kadir, K., Ali, A. M. M., & Ahmad, I. (2019). Real-time Monitoring of Water Quality in Animal Farm: An IoT Application. *2019 IEEE International Conference on Smart Instrumentation, Measurement and Application (ICSIMA)*, 1–6. IEEE. <https://doi.org/10.1109/ICSIMA47653.2019.9057320>
- Birje, Miss. S. v., Bedkyale, Miss. T., Alwe, Miss. C., & Adiwarekar, Mr. V. (2016). Water pollution detection system using pH and turbidity sensors. *International Journal of Advanced Research in Computer and Communication Engineering*, 5(4).
- Brackenbury, L. E. M., Plana, L. A., & Pepper, J. (2010). System-on-Chip Design and Implementation. *IEEE Transactions on Education*, 53(2), 272–281. <https://doi.org/10.1109/TE.2009.2014858>
- Burgio, P., Bertogna, M., Olmedo, I. S., Gai, P., Marongiu, A., & Sojka, M. (2016). A Software Stack for Next-Generation Automotive Systems on Many-Core Heterogeneous Platforms. *2016 Euromicro Conference on Digital System Design (DSD)*, 55–59. IEEE. <https://doi.org/10.1109/DSD.2016.84>
- CC3200 is the industry's first single-chip microcontroller unit with built-in Wi-Fi. (2015, February 15).

- Cloete, N. A., Malekian, R., & Nair, L. (2016). Design of Smart Sensors for Real-Time Water Quality Monitoring. *IEEE Access*, 4, 3975–3990. <https://doi.org/10.1109/ACCESS.2016.2592958>
- Danh, L. V. Q., Dung, D. V. M., Danh, T. H., & Ngon, N. C. (2020). Design and Deployment of an IoT-Based Water Quality Monitoring System for Aquaculture in Mekong Delta. *International Journal of Mechanical Engineering and Robotics Research*, 1170–1175. <https://doi.org/10.18178/ijmerr.9.8.1170-1175>
- Dhaker, P. (2020, February). Wireless Water Quality Monitoring System. EC/TDS/PPM Meter On Limited Budget. (2008).
- Encinas, C., Ruiz, E., Cortez, J., & Espinoza, A. (2017). Design and implementation of a distributed IoT system for the monitoring of water quality in aquaculture. *2017 Wireless Telecommunications Symposium (WTS)*, 1–7. IEEE. <https://doi.org/10.1109/WTS.2017.7943540>
- Geetha, S., & Gouthami, S. (2016). Internet of things enabled real time water quality monitoring system. *Smart Water*, 2(1), 1. <https://doi.org/10.1186/s40713-017-0005-y>
- Gerstlauer, A., Haubelt, C., Pimentel, A. D., Stefanov, T. P., Gajski, D. D., & Teich, J. (2009). Electronic System-Level Synthesis Methodologies. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 28(10), 1517–1530. <https://doi.org/10.1109/TCAD.2009.2026356>
- Giesemann, F., Paya-Vaya, G., Blume, H., Limmer, M., & Ritter, W. (2014). A comprehensive ASIC/FPGA prototyping environment for exploring embedded processing systems for advanced driver assistance applications. *2014 International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS XIV)*, 314–321. IEEE. <https://doi.org/10.1109/SAMOS.2014.6893227>
- Gravity: Analog Turbidity Sensor For Arduino. (n.d.).

- Hwu, W.-M. W. (2016). Introduction. In *Heterogeneous System Architecture* (pp. 1–5). Elsevier. <https://doi.org/10.1016/B978-0-12-800386-2.00009-2>
- Intel® Quartus® Prime Standard Edition Handbook Volume 1: Design and Synthesis. (2018). Intel.
- J Greaves, D. (2011). System on Chip Design and Modelling. *Computer Science Tripos*, p. 130. England: University of Cambridge.
- Karak, T., Bhagat, R. M., & Bhattacharyya, P. (2012). Municipal Solid Waste Generation, Composition, and Management: The World Scenario. *Critical Reviews in Environmental Science and Technology*, 42(15), 1509–1630. <https://doi.org/10.1080/10643389.2011.569871>
- Khatri, P., Gupta, K. K., & Gupta, R. K. (2020). Assessment of Water Quality Parameters in Real-Time Environment. *SN Computer Science*, 1(6), 340. <https://doi.org/10.1007/s42979-020-00368-9>
- Kraxner, F. (2015, February 3). How will ocean acidification impact marine life?
- Lezzar, F., Benmerzoug, D., & Kitouni, I. (2020). IoT for Monitoring and Control of Water Quality Parameters. *International Journal of Interactive Mobile Technologies (IJIM)*, 14(16), 4. <https://doi.org/10.3991/ijim.v14i16.15783>
- Li, L. Y., Jaafar, H., & Ramli, N. H. (2018). Preliminary Study of Water Quality Monitoring Based on WSN Technology. *2018 International Conference on Computational Approach in Smart Systems Design and Applications (ICASSDA)*, 1–7. IEEE. <https://doi.org/10.1109/ICASSDA.2018.8477627>
- Myint, C. Z., Gopal, L., & Aung, Y. L. (2017). Reconfigurable smart water quality monitoring system in IoT environment. *2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS)*, 435–440. IEEE. <https://doi.org/10.1109/ICIS.2017.7960032>
- Ngom, B., Diallo, M., Gueye, B., & Marilleau, N. (2019). LoRa-based Measurement Station for Water Quality Monitoring: Case of Botanical Garden Pool. *2019*

IEEE Sensors Applications Symposium (SAS), 1–4. IEEE.
<https://doi.org/10.1109/SAS.2019.8705986>

Niswar, M., Wainalang, S., Ilham, A. A., Zainuddin, Z., Fujaya, Y., Muslimin, Z., ... Fall, D. (2018). IoT-based Water Quality Monitoring System for Soft-Shell Crab Farming. *2018 IEEE International Conference on Internet of Things and Intelligence System (IOTAIS)*, 6–9. IEEE.
<https://doi.org/10.1109/IOTAIS.2018.8600828>

O'Donnell, D. (2017, November 29). Understanding Conductivity Cell Constants.

Pasika, S., & Gandla, S. T. (2020). Smart water quality monitoring system with cost-effective using IoT. *Heliyon*, 6(7), e04096.
<https://doi.org/10.1016/j.heliyon.2020.e04096>

pH Meter. (2022, November 6).

Pujar, P. M., Kenchannavar, H. H., Kulkarni, R. M., & Kulkarni, U. P. (2020). Real-time water quality monitoring through Internet of Things and ANOVA-based analysis: a case study on river Krishna. *Applied Water Science*, 10(1), 22.
<https://doi.org/10.1007/s13201-019-1111-9>

Purkayastha, A. A., Shidhibhavi, S. A., & Tabkhi, H. (2018). Taxonomy of Spatial Parallelism on FPGAs for Massively Parallel Applications. *2018 31st IEEE International System-on-Chip Conference (SOCC)*, 55–60. IEEE.
<https://doi.org/10.1109/SOCC.2018.8618501>

Raju, K. R. S. R., & Varma, G. H. K. (2017). Knowledge Based Real Time Monitoring System for Aquaculture Using IoT. *2017 IEEE 7th International Advance Computing Conference (IACC)*, 318–321. IEEE.
<https://doi.org/10.1109/IACC.2017.0075>

Saravanan, K., Anusuya, E., Kumar, R., & Son, L. H. (2018). Real-time water quality monitoring using Internet of Things in SCADA. *Environmental Monitoring and Assessment*, 190(9), 556. <https://doi.org/10.1007/s10661-018-6914-x>

Staff, F. (2020, December 2). The Pros and Cons of Optical Dissolved Oxygen Sensors. *FishSens Magazine*.

Ultrasonic Level Sensor. (2022, April 15).

Vijayakumar, N., & Ramya, R. (2015). The real time monitoring of water quality in IoT environment. *2015 International Conference on Circuits, Power and Computing Technologies [ICCPCT-2015]*, 1–4. IEEE. <https://doi.org/10.1109/ICCPCT.2015.7159459>

What is a Raspberry Pi? (2015, August 20).

What is Arduino? (2018, February 5).

William D., M., & Dennis A., S. (2000). System-on-chip Design Methodology in Engineering Education. *International Conference Engineering Education*. Taipei, Taiwan.

Zhong, G., Niar, S., Prakash, A., & Mitra, T. (2016). Design of Multiple-Target Tracking System on Heterogeneous System-on-Chip Devices. *IEEE Transactions on Vehicular Technology*, 65(6), 4802–4812. <https://doi.org/10.1109/TVT.2016.2546957>

Ziener, D. (2018). *Improving Reliability, Security, and Efficiency of Reconfigurable Hardware Systems*.

What is FPGA. (2020). <https://www.intel.com/content/www/us/en/products/programmable/fpga/newto-fpgas/resource-center/overview.htm>

Gravity: Analog Turbidity Sensor For Arduino - DFRobot. (2021, January 12). Dfrobot. <https://www.dfrobot.com/product-1394.html?search=turbi>

A. (2022a, March 7). *Water level sensor types and works*. Apure. <https://apureinstrument.com/blogs/water-level-sensor-types-and-works/>

- USA. Intel. (2020). AN 307: Intel® FPGA Design Flow for Xilinx Users.
<https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/an/an307.pdf>
- Intel. (2021a). Intel FPGA and Programmable Devices.
<https://www.intel.com/content/www/us/en/products/programmable.html>
- Intel. (2021b). Skylake (microarchitecture). In Wikipedia.
[https://en.wikipedia.org/w/index.php?title=Skylake_\(microarchitecture\)&oldid=998421085](https://en.wikipedia.org/w/index.php?title=Skylake_(microarchitecture)&oldid=998421085)
- Intel Corporation. (2014). Booting and Configuration Introduction.
https://www.intel.cn/content/dam/www/programmable/us/en/pdfs/literature/hb/arria-10/a10_5400a.pdf
- Intel Corporation. (2015). Instantiating the Nios II Processor. Instantiating the Nios II Processor Intel PSG website. (2020).
<https://www.intel.com/content/www/us/en/prog>