



PHD THESIS

Design of Training Environment: An Emulator of Water Supply System in Turin Node of Italian Cyber Ranges

PHD PROGRAM IN COMPUTER SCIENCE: XXXI CYCLE

Author:

Bishwajeet Kumar PANDEY
bishwajeet.pandey@gssi.it

Advisor:

Prof. Paolo PRINETTO
paolo.prinetto@polito.it

February 2019

GSSI Gran Sasso Science Institute
Viale Francesco Crispi, 7 - 67100 L'Aquila - Italy

“Discretion will protect you and understanding will guard you.”

Holy Bible

“The good we secure for ourselves is precarious and uncertain until it is secured for all of us and incorporated into our common life”

Jane Addams

“No one secure us but ourselves. No one can and no one may. We ourselves must walk the path.”

Lord Buddha

Acknowledgements

I am thankful to Prof. Rocco De Nicola, from IMT Lucca, the first coordinator of the PhD program in Computer Science at GSSI, who give me an opportunity to get admission in PhD and introduce to my PhD Supervisor Prof Paolo Prinetto.

After admission, course work taught by Prof Patricia Lago from Vrije Universiteit Amsterdam, Prof Radu Clinescu from University of York, Prof Antonia Bertolino from ISTI Pisa, Prof Massimiliano Di Penta from University of Sannio, Prof Patrizio Pelliccione from University of L'Aquila, Michele Loreti From Universita Degli Studi Firenze, Prof Mirco Tribastone from IMT Luca, Prof Rocco De Nicola from IMT Lucca, Prof Shmuel Zaks from Technion Israel, Prof Fabrizio Grandoni from IDSIA Switzerland, Prof David Peleg, Weizmann Institute of Technology Israel, Prof Michele Flammini from Gran Sasso Science Institute L'Aquila and Prof Luca Aceto from Gran Sasso Science Institute L'Aquila makes me ready for this doctoral dissertation.

I would like to express all my gratitude to my PhD advisor, Prof. Paolo Prinetto, from Politecnico Di Torino Italy. His guidance, support and advice made my thesis work possible, and I am deeply indebted with him.

I would like to thank Prof. Luca Aceto and Prof. Michele Flammini, the coordinator of the PhD program in Computer Science at GSSI for their kindness and willingness to help me with everything I needed.

I would like to thank the other co-authors of some works which this thesis is based on: Dr Giuseppe Airo Farulla, University of Venice and Dr Marco Indaco, Aster Spa.

I would like to thank the dissertation committee members, for devoting their time to my dissertation defence.

I would like to thank my dear colleagues and friends from Politecnico Di Torino and Istituto Superiore Mario Boella, in whose offices I am pleasantly placed during my visits at Turin.

Last but not least, I am grateful to my family and I dedicate this thesis work to my parent and wife.

Bishwajeet Kumar Pandey

Abstract

John Keats said, “Nothing ever becomes real till it is experienced”. Therefore, we start our thesis with the case study of existing Water Management Cyber Physical System (WMCPS) in a suburban locality of Italy. We called this system as the EVA System.

Later, we review existing Communication Protocols, Sensors and Actuators used in WMCPS. Since then, many reviews and design of water management system have been published; but though useful books of reference, they do not meet the standards that this thesis is intended to supply.

In the first year, we have been collecting information and component with the intention of arranging them in such a way that anyone without much labour might gain the general idea of the water supply system using a training environment that my emulator provides.

After that, we develop small emulator of WMCPS using environmental sensor, controllers and actuators. Water Flow sensor, Water Temperature Sensor and Water Level Sensor are some example physical sensors used in our design. Pump and Relay are the most common actuators which we use in our system. We work with people of Laboratorio Nazionale di Cyber Security (Cyber Security National Laboratory) and CINI - Consorzio Interuniversitario Nazionale per l'Informatica (Italian National Inter-University Consortium for Informatics) and perform a systematic review and design of different attacks strategies for Water Management Cyber Physical System. Information Disclosure, Scanning: Accessible Folder, Exploitation: SQL Injection and Exploitation: Denial of Service (DOS) are four attacks strategies which we perform on our emulator with the help of the Cyber Security team of Leonardo Querzoni of Sapienza University of Rome. To understand communication among sensor, actuator and controller, we explore serial Communication among Controllers, Sensors and Actuators in Water management Cyber Physical System (WMCPS) using Raspberry Pi and Arduino Uno as a controller. RS485 Modbus communication took place with TTL to Modbus Converter on Arduino. RS485 Modbus communication took place with USB to Modbus Converter on Raspberry. TCP/IP Modbus communication is implemented on final stage among one master and four slaves. For a proper understanding of working of sensors and actuators, we simulate WMCPS using EPANET Simulator. After the simulation of water flow and testing of communication protocols, we design Raspberry Pi based Water Management Cyber Physical System with and without SECUBE. The Python script is used to update sensors reading and actuator status on the MySQL database server. Depending on sensors reading, our master sends the command to actuators. For example, if the water level of sensor reading is reaching the threshold at tank X, then master stop the pump that is sending water to the tank X. We design a website <http://cps4wm.info/> using PHP, CSS and HTML to show sensors reading and actuators status stored on MySQL cloud. In last, we explore the future scope of WMCPS with CISCO Cyber Range and Niagara Router.

List of Publications

In the following, the list of the author's publications, which some chapters of this doctoral dissertation are based on:

1. Bishwajeet Pandey, Giuseppe Airò Farulla, Marco Indaco, and Paolo Prinetto, "Survey of Computational Ability of Sensors-Actuators in Water Management Cyber-Physical System", Journal of Advanced Research in Dynamical and Control Systems, Vol. 10, 04-Special Issue, 2018, <http://www.jardcs.org/backissues/abstract.php?archiveid=4809>
2. Bishwajeet Pandey, Giuseppe Airò Farulla, Marco Indaco, Ludovico Iovino and Paolo Prinetto, "Design and Review of Water Management System using Ethernet, Wi-Fi 802.11n, Modbus, and other communication standards", Wireless Personal Communications, 5 Feb 2018, DOI: 10.1007/s11277-018-5380- https://link.springer.com/article/10.1007/s11277-018-5380-7

Contents

Acknowledgements	iii
Abstract	iv
List of Publications	v
List of Figures	xiii
List of Tables	xvii
1 Introduction	1
1.1 Cyber Physical Systems (CPS)	2
1.2 Cyber Physical Systems (CPS) for Water Management	3
1.3 Security in Cyber Physical Systems (CPS)	3
1.3.1 Definitions	3
1.3.2 Security issues in Cyber Physical Systems (CPS)	4
1.3.3 Security issues in Cyber Physical Systems (CPS) for Water Management	4
1.4 State-of-the-Art and Related Works	5
1.5 NIST Framework for Critical Infrastructure Cybersecurity	9
1.6 Italy’s National Strategic Framework for Cyberspace Security	11
1.7 Our Contribution	12
1.8 Dissertation Organization	13
2 Case Study of Existing Water Management Cyber Physical System	17
2.1 EVA System	17
2.2 System Development Life Cycle of EVA System in Perspective of System Engineer	18
2.3 Preliminary Investigation	20
2.4 Feasibility Study	22
2.4.1 Technical Feasibility	23
2.4.2 Operational Feasibility	23
2.4.3 Economic Feasibility	23
2.5 System Development Life Cycle of EVA System in Perspective of Computer Engineer	23
2.5.1 Phase 1: Requirement	23
2.5.2 Phase 2: Analysis:	24
2.5.3 Phase 3: Design:	27
2.5.4 Phase 4: Coding:	27
2.5.5 Phase 5: Testing:	29
2.5.6 Phase 6: Maintenance	31
2.6 VegaBAR 14	41

2.7	VegaWell 52	42
2.8	Vegason 61	42
2.9	Honeywell Hawk CPU	43
2.10	BELIMO Rotary Actuator GR24A-SR-5	44
2.11	Electromagnetic Measurement: OPTIFLUX 2000 Sensor and IFC050/100 Converter	45
2.12	ProMinent, DULCOMETER® diaLog DACa	47
	2.12.1 Amperometric variables, measuring ranges depending on sensors	48
	2.12.2 Potentiometer measured variables, measuring ranges corresponding to transmitters	49
2.13	Prominent GammaX	49
2.14	Communication Standards	49
2.15	Conclusion	49
3	Reviews of Communication Protocols, Sensors and Actuators in Water Management Cyber Physical System	51
3.1	Communication Protocols	51
	3.1.1 Ethernet	51
	3.1.2 WiFi	51
	3.1.3 Modbus	52
3.2	Sensors	52
	3.2.1 Flow Sensors	52
	3.2.2 Temperature Sensors	52
	3.2.3 Level Sensors	53
3.3	Actuators	53
	3.3.1 Pump	54
4	A systematic review and Design of Different Attacks Strategies for Emulator of Water Management Cyber Physical System (WM-CPS)	59
4.1	Introduction	59
4.2	Past Attacks on Water Supply before 1994	60
4.3	Past Cyber Security Attacks between 1994 and 2016	61
	4.3.1 Attacks on Arizona’s Theodore Roosevelt Dam	62
	4.3.2 Attacks on Maroochy Water Services (MWS)	63
	4.3.3 Attacks on Harrisburg water treatment plant	64
	4.3.4 Attacks on Tehama Colusa Canal Authority (TCCA)	64
	4.3.5 Attacks on Water Plant in South Houston	64
	4.3.6 Attacks on Bowman Dam	65
	4.3.7 Attacks on Kemuri Water Company (KWC)	66
4.4	Present Cyber Security Attacks on Emulated Water Management System	66
	4.4.1 Information Disclosure	67
	4.4.2 Forced Browsing - Accessible Folder	67
	4.4.3 Blind SQL Injection - SQL-Injection	68
	4.4.4 Delayed SQL Execution (MySQL): DOS	68
4.5	Future Cybersecurity Practices for the Water Sector	69
4.6	Conclusion and Future Scope	69
5	Serial Communication among Controllers, Sensors and Actuators in Emulator of Water management Cyber Physical System (WM-CPS)	71
5.1	INTRODUCTION	71

5.2	OVERALL SYSTEM ARCHITECTURE	72
5.3	DESIGN OF HARDWARE AND SOFTWARE COMPONENTS	74
5.3.1	Design of Sensor Node	74
5.3.2	Design of Actuator Node	74
5.4	SERIAL COMMUNICATION Among CONTROLLER, Sensor Node and Actuator Node	74
5.4.1	Programming the Arduino from a Raspberry Pi	74
5.4.2	Serial communications to the Raspberry Pi for Sensor	76
5.4.3	Serial communications from the Raspberry Pi for Actuator	78
5.5	CONCLUSION AND FUTURE SCOPE	78
6	Implementation of RS485 Modbus and TCPI/IP in Emulator of Water Manage- ment Cyber Physical System	81
6.1	INTRODUCTION	81
6.2	MODBUS COMMUNICATION: HARDWARE BASED	85
6.3	MODBUS COMMUNICATION: SOFTWARE BASED	87
6.4	CONCLUSION	89
6.5	FUTURE SCOPE	90
7	Simulation of Water Management Cyber Physical System using EPANET before Design of Emulator	91
7.1	EPANET: Water Quality Simulator	91
7.1.1	Physical Peripherals	92
7.1.1.1	Junctions	92
7.1.1.2	Tanks	94
7.1.1.3	Reservoirs	94
7.1.1.4	Emitters	94
7.1.1.5	Pipes	95
7.1.1.6	Pumps	95
7.1.1.7	Minor Losses in Pump	96
7.1.1.8	Valves	97
7.1.2	Non-Physical Components	98
7.1.2.1	Curves	98
7.1.2.2	Patterns	98
7.1.2.3	Controls	99
7.2	EPANET-MSX: Water Quality Simulator	100
7.3	Merlion: Water Quality Simulator	100
7.4	WST: Water Security Toolkit	101
7.4.1	Configuration File	102
8	Design of Emulator for Water Management Cyber Physical System: using Rasp- berry Pi without SECUBE	109
8.1	Connect with Eduroam/Wi-Fi of ISMB	109
8.1.1	Assign Static IP to Master and Slaves	112
8.1.2	Install PyMySQL	112
8.2	Flow Sensor	112
8.3	Pump and Relay	113
8.4	Waterproof Temperature Sensor	114

8.4.1	Python Program to Update Water Temperature on MySQL Server	117
8.5	Environment Temperature and Humidity Sensor	117
8.6	Level Sensor	117
8.7	Slave Controller at Lake Tank	118
8.8	Slave Controller at Purifier Tank	118
8.9	Slave Controller at Clean Tank	118
8.10	Slave Controller at House Tank	119
8.11	Master Controller	120
9	Design of Emulator for Water Management Cyber Physical System: using Rasp- berry Pi with SECUBE	123
9.1	What is Secube?	123
9.1.1	STM32F4 Processor	123
9.1.2	Lattice MachXO2-7000 FPGA	124
9.1.3	EAL5+ Certified Smart Card	124
9.2	About SECube Board	124
9.3	About Software related to SECube Board	125
9.3.1	AC6-Tools	125
9.3.2	STM32CubeMX	125
9.3.3	ST-Link/v2 in-circuit programmer	126
9.4	Slave Controller at Lake Tank with SECUBE	126
10	Web Interface and Telepot of Emulator for Water Management Cyber Physical System	127
10.1	PHP	128
10.1.1	PHP Program to force https	128
10.1.2	PHP Program to Establish connection with database on MySQL Server	129
10.1.3	PHP Program to Fetch Data from waterflow table	129
10.1.4	PHP Program to read the value of Level Sensor of four Tank	129
10.1.5	PHP Program to read the value of Temperature Sensor in Purifier Tank	130
10.1.6	PHP Program to Update reading of Environment Temperature and Humidity Sensor on our Project Website	131
10.1.7	PHP Program to Update reading of Flow Sensor on our Project Website	131
10.1.8	PHP Program to Print Pump Status on our Project Website	132
10.1.9	PHP Program to Print Valve Status on our Project Website	132
10.2	Javascript	133
10.2.1	Javascript Code to Generate Google Chart from MySQL tables	133
10.3	HTML	134
10.4	MySQL	135
10.5	Telegram	135
10.5.1	actuators.sh	137
10.5.2	controlOFF.sh	137
10.5.3	controlON.sh	137
10.5.4	sensors.sh	138
10.5.5	sensors_onlyDistance.sh	139
10.5.6	sensors_onlyFlow.sh	140
10.5.7	sensors_onlyTemperature.sh	140

11 Future of Our Emulator	141
11.1 Make It Publicly Accessible	141
11.2 Replace with Real Sensor and Actuators	141
11.3 Increase Number of Sensors and Actuators	141
11.4 Explore Other Communication Standards	142
11.5 Integrate Niagara JACE Router	142
11.6 Design Other Environment for Turin Node of Cyber Ranges	142
A Appendix	149
A.1 Python Program to Update Water Temperature on MySQL Server	149
A.2 Python Program to test sensors and actuators attached with Lake Tank	150
A.2.1 Functions	150
A.2.2 Variables	151
A.3 Python Program to test sensors and actuators attached with Purifier Tank	154
A.3.1 Functions	154
A.3.2 Variables	155
A.4 Python Program to test sensors and actuators attached with Clean Tank	159
A.4.1 Functions	159
A.4.2 Variables	159
A.5 Python Program to test sensors and actuators attached with House Tank	162
A.5.1 Functions	162
A.5.2 Variables	163
A.6 Python Program execute at Master controller to manage sensors and actuators attached with four slaves	166
A.6.1 Functions	166
A.6.2 Variables	166
A.7 Python Program to test sensors and actuators attached with Lake Tank using SEcube	170
A.8 Code of https://cps4wm.info/temperaturesensor.php	179
A.9 Python Code to Manage TeleBot on Telegram	184

List of Figures

1.1	Cyber Physical Systems [8]	2
1.2	Water Management Using Cyber Physical System	3
1.3	Application of Cyber Physical Systems in Water Management [76]	5
1.4	Password Model [20]	8
1.5	Italian CyberSecurity Framework [61]	11
1.6	Telegram Bot of Our System Working on Android Phone	15
2.1	Overview of EVA System	18
2.2	Business Process Model of EVA System	19
2.3	Functionality of EVA System	21
2.4	PUMP as an Actuator	22
2.5	System Development Life Cycle of EVA System	24
2.6	Flow Chart to Start/Stop PUMP to Fill Tank of EVA System	25
2.7	Flowchart for checking contamination in drinking water supplied by EVA system	26
2.8	Flow Chart of Deciding Water is potable or not Based on Sensor Reading in EVA System	27
2.9	Flow Chart to Control Actuators of EVA System	28
2.10	Flow Chart for Display of Current Status of EVA System	29
2.11	Entity Relationship Diagram of EVA System	30
2.12	Process Flow Diagram from Sensors to Niagara Framework [18]	32
2.13	Danfoss Frequency Converter [18]	34
2.14	Danfoss Soft Starter [18]	34
2.15	8 Different PUMPs used in EVA System	35
2.16	CAPRARI E6-8 E6SX PUMP [18]	36
2.17	CAPRARI E6-8 E8PX PUMP [18]	36
2.18	CAPRARI E6-8 E6RX PUMP [18]	37
2.19	Caprari E6-8: MACX6-2A PUMP [18]	38
2.20	Caprari E6-8: MACX6-2B PUMP [18]	38
2.21	E6-8 MACX8 PUMP [18]	39
2.22	HV PUMP [18]	40
2.23	Inlet and Outlet Directions: 1 is Standard and 2-4 are On Request [18]	41
2.24	Electronics - VEGABAR 14 [18]	42
2.25	VegaWell 52: Ceramic Measuring Cell-based Absolute Submersible Pressure Sensor [18]	43
2.26	Vegason 61 Sensor for continuous level measurement and its Display Board [18]	43
2.27	BELIMO Rotary Actuator GR24A-SR-5 [18]	44
2.28	OPTIFLUX 2000 Sensor [18]	45
2.29	Internal structure of OPTIFLUX 2000 Sensor: 1: Induced voltage, 2: Electrodes, 3: Magnetic field, 4: Coils [18]	46
2.30	IFC050 Converter [18]	46
2.31	IFC 100 Krohne Signal converter for Electromagnetic flow meters [18]	47

2.32	ProMinent, DULCOMETER® diaLog DACa. [18]	48
2.33	OPC: Open Protocol Communication [32]	50
3.1	Flow Sensor in WMCPS	53
3.2	Temperature Sensor and Cyber Security Attack on Water Management System [16]	54
3.3	Ultrasonic Sensor measure level of Water in Tank	55
3.4	Design of Centrifugal pump [38]	56
4.1	Water Terrorism before Cyber Security Attacks on Water System	61
4.2	Reported Incident of Cyber Terrorism related to Water	62
4.3	Theodore Roosevelt Dam [77]	62
4.4	Attacks on Maroochy Water Services	63
4.5	Attacks on Harrisburg Water Treatment Plant	64
4.6	Attacks on Tehama Colusa Canal Authority	65
4.7	Three Information Accessible to Hackers of Bowman Dam	65
4.8	Working of Kemuri Water Company's (KWC's) Water System	66
4.9	Wireless Sensors and Actuators Networks (WSANs) Based Water Management System	66
4.10	List of Recommended Cybersecurity Practices	70
5.1	Serial Communication in Sensor and Actuator Network	72
5.2	MEGA2560 R3 Arduino Board with ATmega16U2 chip [11]	73
5.3	Raspberry PI 3 Model B with 1.2 GHz Quad Core CPU [66]	73
5.4	XCSOURCE 5V/9V/12V/24V Channel Relay Shield Module	74
5.5	Actuator Node: Relay for Start and Stop of Pump	75
5.6	Level Sensor Reading on Serial Monitor	76
5.7	Level Sensor Reading on RPi Terminal	76
5.8	Flow Sensor Reading on Serial Monitor of Arduino	78
6.1	USB 2.0 to Modbus RS485 Converter	82
6.2	Communication via USB-Modbus and Modbus-UART	82
6.3	Rear side of TTL to Modbus RS485 Converter Module	84
6.4	Front side of TTL to Modbus RS485 Converter Module	85
6.5	Modbus in Water Management Cyber-Physical System: Actuator	86
6.6	Work Flow of Water Management Cyber Physical System: Actuator	86
6.7	Work Flow of Water Management Cyber-Physical System: Sensor	87
6.8	Output of Slave Purifier	88
6.9	Output of Slave Lake	89
6.10	Output of Master.py	90
7.1	Physical Peripherals in a Typical Water Management System	92
7.2	Results of Node Outputs	93
7.3	Results of Link Outputs	93
7.4	Plot of Pressure on Junction Home	93
7.5	Plot of Head in Clean Tank	94
7.6	Plot of Demand in Reservoir: Node 1	95
7.7	Unit Headloss in Pipe 6	95
7.8	Flow For Pump in Water Distribution System	96
7.9	Velocity through Valve in Water Distribution System	97

7.10	Curve Editor	98
7.11	Water Demand in Interval of 4 Hours	99
7.12	Pattern of Water Demand in Interval of 6 Hours	100
7.13	Layout of Sample Water Distribution Network	102
7.14	Location of Two Boosters in Water Distribution Network	104
7.15	Location of Four Boosters in Water Distribution Network	105
7.16	Location of Six Boosters in Water Distribution Network	106
7.17	Location of Eight Boosters in Water Distribution Network	107
7.18	Location of Ten Boosters in Water Distribution Network	108
8.1	Architecture of Emulator of Water Management Cyber Physical System	110
8.2	Implemented Emulator of Water Management Cyber Physical System	111
8.3	YFS201 Water Flow Sensor	113
8.4	Working of YF-S201 sensor [79]	113
8.5	Relay Terminals and Pins	114
8.6	High Voltage Output Connector of Relay Module	115
8.7	Relay is used to control High Voltage Pump	115
8.8	Waterproof Temperature Sensor	116
8.9	Environment Temperature Sensor	117
8.10	The HC-SR04 Ultrasonic Sensor	118
8.11	Ultrasonic Sensor for Water Level Measurement [58]	119
8.12	System at Lake Tank	120
8.13	System at Purifier Tank	120
8.14	System at Clean Tank	121
8.15	System at House Tank	121
9.1	SECube Block Diagram [81]	124
9.2	SECube in between Slave and Master Controller	126
10.1	Home Page of https://cps4wm.info/	127
10.2	Login as a root@63.141.225.189 to VPS	128
10.3	Web Page Dedicated to Temperature Sensor on Project Website	134
10.4	List of Tables in MYSQL Database of http://cps4wm.info	135
10.5	Description of DistanceLOG Table	136
10.6	Distance of water from Level Sensor	136
10.7	Telegram Bot of Our System Working on Android Phone: Sensor Reading	138

List of Tables

1.1	Part 1: A selection of NIST cybersecurity-related publications [5]	9
1.2	Part 2: A selection of NIST cybersecurity-related publications	10
2.1	Components of EVA System	18
2.2	Specification of CAPRARI E6-8 E6SX, E6PX, E6RX PUMP	37
2.3	Parts of Caprari E6-8: MACX6-2A, MACX6-2B PUMP	39
2.4	Specification of E6-8 MACX8 PUMP	40
2.5	Technical Data Electric Motors IP 55 [18]	40
2.6	Specification of BELIMO Rotary Actuator GR24A-SR-5 [18]	44
3.1	Evolution of Water Pump [[67]	55
4.1	Attacks Strategies Used in Past Attacks and Attacks Strategies on Emulated System	60
4.2	Security Attacks on Emulated Water Management Cyber-Physical System	67
4.3	Common Vulnerability Scoring System (CVSS) with Severity 1	67
4.4	Forced Browsing: Accessible Folder	68
4.5	Common Vulnerability Scoring System (CVSS) with Severity 9	68
4.6	Common Vulnerability Scoring System (CVSS) with Severity 10	69
5.1	Code of Level Sensor for Arduino written in Python	75
5.2	Code of Level Sensor for Raspberry Pi written in Python	75
5.3	Serial Communication between Arduino with Flow Sensor and RPi	77
5.4	Serial Communication between Arduino with Actuator and RPi Arduino Code for Relay-PUMP	79
6.1	Different Variety of Interface	83
6.2	Pin of TTL to RS485 Converter	83
6.3	Interconnection of MCU and RS485 module	84
6.4	Pin setting when Raspberry is Transmitter, Arduino is Receiver	87
6.5	Pin setting when Raspberry is Receiver and Arduino is Transmitter	88
7.1	Tank Properties in Water Distribution System (Fig. 7.1)	94
7.2	Properties of Network Link:Pump	96
7.3	Minor Loss Coefficients for selected Fittings	96
7.4	Valve Type & its specific Setting	97
7.5	Some Example of Simple Control	99

Water is the vehicle of nature.

Leonardo Da Vinci

1

Introduction

Water when marriage with the earth made crops possible. And, when its alliance with the man leads to souls to be purified. According to Rig Veda, water is a giver of health, strength, long life, wealth and immortality. It is the upholder of all lives and the saviour of everything living or dead on the earth. Rig Veda also claims that before the creation of the universe, there was said to be nothing but the bottomless, uninterrupted, limitless water. The same water was in an egg from which everything else emanated. According to modern science, fossil evidence found in western Australia, eastern South Africa and elsewhere indicate that single-celled bacteria developed in water of oceans some 3.5 billion years ago. Above all, water is an essential component of inorganic mixtures from which chemists, in their quest to understand the origin of life, attempt to produce complex organic molecules. The main motive of our work is to develop an emulator of the secure water distribution system to use as a training environment. To manage a critical infrastructure like the distribution of water, we use a cyber physical system (CPS). To enhance security in critical systems like CPS is a demand of current era [8]. Any threats to cybersecurity of water management CPS will create a severe impact on the life of people. This threat is particularly troublesome because water is more critical than even power in daily life. Manhattan U.S. Attorney Preet Bharara said after a recent Cyber Attack on New York Dam in March 2016: "We now live in a world where devastating attacks on our financial system, our infrastructure and our way of life can be launched from anywhere in the world, with a click of a mouse" [56]. So, any intruder in water management CPS will not only get information about water-level and temperature but also able to operate the chemical dosing devices and add chemicals more

than threshold values defined by the World Health Organization (WHO). Similarly, any hacker after getting access to water management CPS for water distribution in a city will force the whole city to live without water for a while. In order to save CPS and finally humanity in the broader sense, it is essential to improve security measures to improve the cyber security of infrastructure behind water management CPS. The core goal of our thesis is to make it more secure than it was. We have to take care of the following fundamental principles in order to make secure CPS: prevention, preparation, planning, administration of incident, rehabilitation, alleviation, arbitration, after attack analysis, and learning for future [69].

1.1 Cyber Physical Systems (CPS)

Cyber Physical Systems (CPS) is made of computational units in order to control physical processes [8]. In other words, CPS is an integration of physical systems, communication systems, and computation system as shown in Fig. 1.1. CPS plays a pivotal role in the following 18 sectors of critical infrastructure and services: Farming and Food, Banking, Chemical, Commercial Enterprise, Communications, Automation in Manufacturing, Water Dams, Defense, Emergency Response, Power, E-Governance, Health care and assisted leaving, Information Technology, Archaeology, Nuclear Facilities, Waste Management, Postal and Shipping, Intelligent Transportation, and Water Management [69].

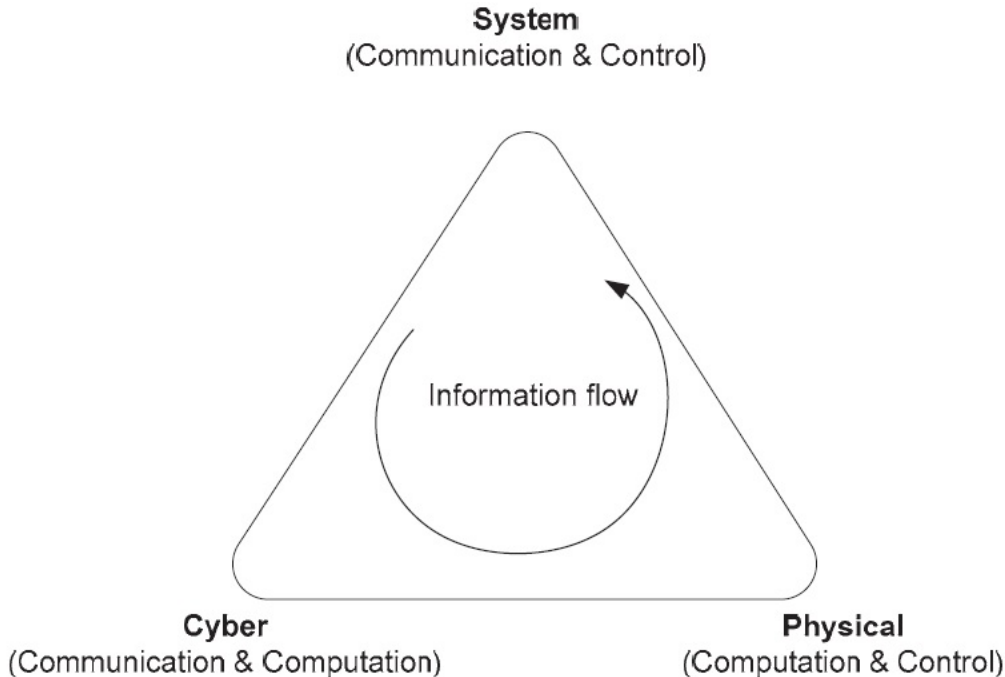


FIGURE 1.1: Cyber Physical Systems [8]

The unification of physical systems and networked computing has opened a new era of engineered systems, especially known as Cyber-Physical Systems (CPS) [3]. According to Lee and Seshia [44], the term "cyber-physical systems" was given by Helen Gill at the National Science Foundation of the

United States in 2006. Whereas, the concept of CPS dated back to 1948 when Norbert Wiener coined a term called cybernetics [60].

1.2 Cyber Physical Systems (CPS) for Water Management

Water is not only valuable resources but also one of the important critical infrastructure, where CPS plays a major role. CPS is mainly used in the distribution of water as water is an essential daily life supply. CPS is mainly used in the distribution of water to the existing population as shown in Fig. 1.2. This CPS must be ready to scale in order to take care of the growing population due to urbanization and migration toward city.

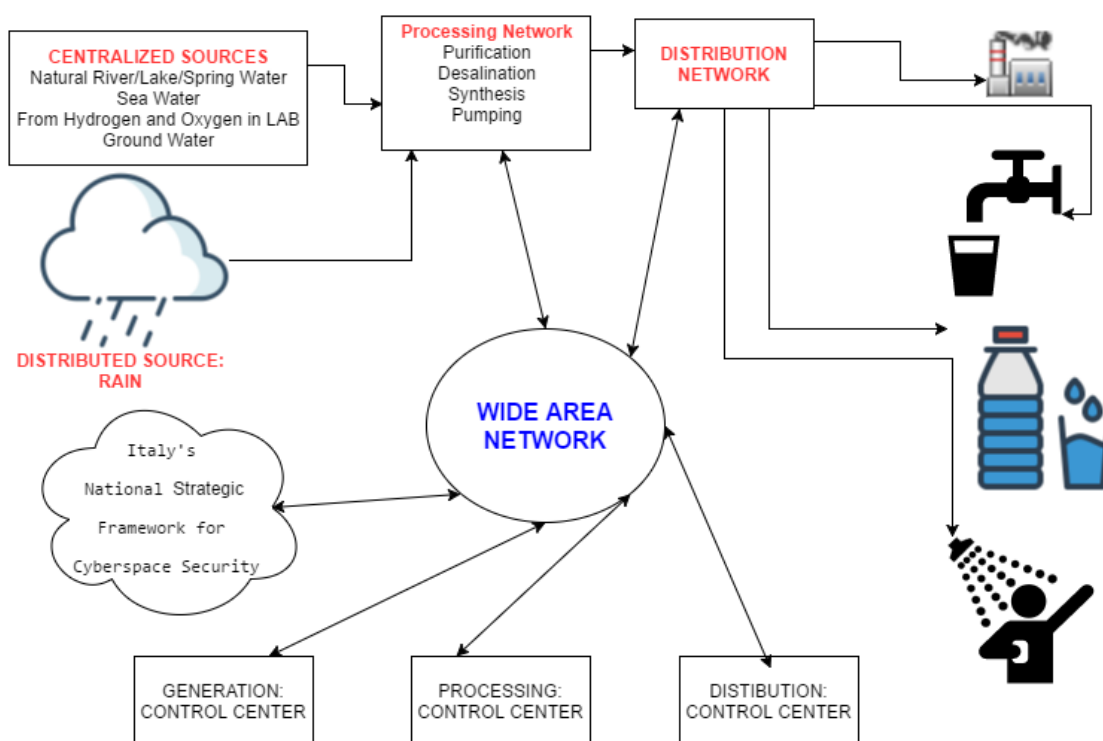


FIGURE 1.2: Water Management Using Cyber Physical System

1.3 Security in Cyber Physical Systems (CPS)

1.3.1 Definitions

There is no absolute definition of security. Its definition change with time by time. A system which was taken as secure yesterday never treats as safe today. Similarly, the contemporary definition of security will not pass a test of the future. It is just like the theory of "Fittest to survive" of Darwin, only fit system which can stand against the threat of that time always survive. To survive, it has

to evolve something new in order to compete with tomorrow threats. For example, at the beginning of the evolution of human society, there was a demand for security from natural calamity. Fear of nature leads human society to formation and belief in nature God. Those God become obsolete when we surpass our awe of life and develop our protection system against nature. Destructive attacks are possible on the financial system, infrastructure and our way of life in the current scenario. These attack can be launched from anywhere in the world, with a click of a mouse lead to the formation of the army of not only soldiers but also an ethical hacker. Time is not far when there will four types of armed force in every country: Army, Navy, Airforce and Cyberforce.

1.3.2 Security issues in Cyber Physical Systems (CPS)

Role of CPS in human life is increasing day by day. Therefore, its security concern is gaining more consideration. Human is imperfect creation so as CPS made by the human. Early CPSs were able to deal with data imperfection (loss and delays of packets, collision, quantification, etc.), only, but do not consider the possibility of transmission of false data with the wrong intention, SQL injection and Denial of Service (DOS) Attacks. Now, there is a chance that data acquired by the sensors or calculated by the controllers may be changed by attackers in the following ways:

- Learn data during transmission in communication channels;
- Insert malicious content into intercepted data, and then send it again as if the data were original;
- SQL Injection Attacks on Database;
- Launch the so-called Denial-of-Service (DoS) attacks.

1.3.3 Security issues in Cyber Physical Systems (CPS) for Water Management

When access of CPS goes into hand of intruder, then he will be able to control actions that may cause catastrophic damages. In one case, they force the population to die with thirst who leave far from natural water resources like spring, river and sea. In another example, they may insert chemical by operating chemical dosing devices associated with water management CPS. They may also create some other following issues:

- The attack on the water level sensors and water temperature sensor are possible with manipulation of the cryptography algorithm;
- Readings of a temperature sensor will give a false reading if the attacker activates a heater near the sensor;

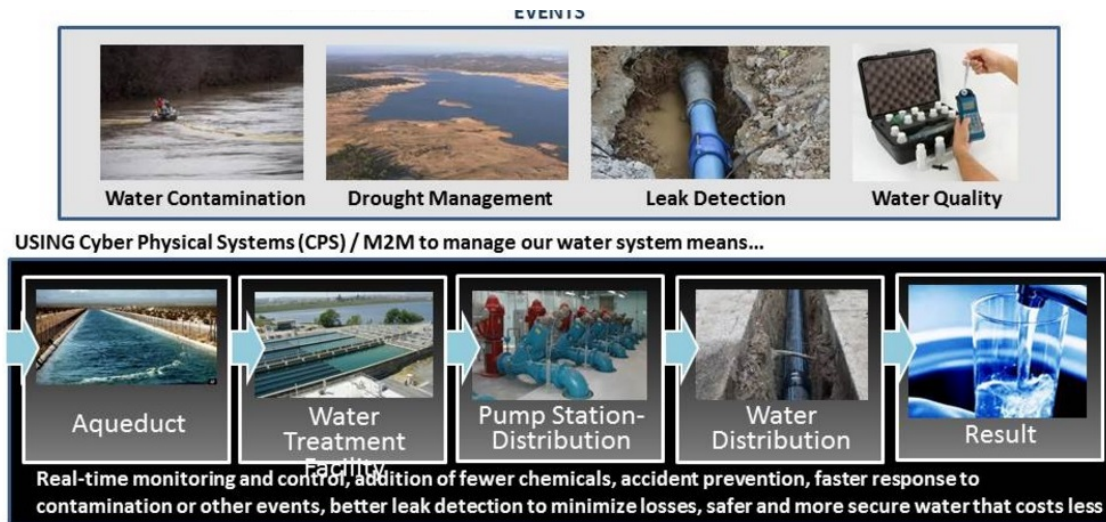


FIGURE 1.3: Application of Cyber Physical Systems in Water Management [76]

- If the attacker will record previous reading or use machine learning or system identification to generate a fake sequence of readings and replay those fake reading then the system operator will assume everything is fine;
- Hacker after getting access to water management CPS for water distribution in the city will force the whole city to live without water for a while;
- Also, there is a chance for possible contamination to spread epidemic among the the consumer of water and create artificial leak with an intention to wastewater in the way.

And finally, there is a limited way of construction but there is an infinite way for destruction. It is not possible to list all possible threat but it is possible to concentrate on the security of our system, which we shall do in our research.

1.4 State-of-the-Art and Related Works

The SmartAmerica Challenge project introduces a new framework for water allocation using cellular-based CPS [76]. It monitors water quality. Along with that, it has a faster response to possible contamination and precise leak detection, as shown in Fig. 1.3 [76].

There are many water management CPS already in use by different municipality. We also studied EVA System in Italy [18] and proposed a design with Controllers (Raspberry Pi, Arduino), Sensors(Physical and Chemical) and Actuators. Also, use the Python Library for Security supported on Raspberry and SECUBE to enhance security on our proposed system.

Anas Motii et al. [8] introduce an MDE approach to enhance security in CPS at the early design stage. It uses two different views of security patterns: abstract and concrete. They use an abstract

view to select relevant solutions. Then, the concrete view is used to evaluate various implementation alternatives. Their modelling also depends on accepted OMG standards like UML and MARTE. The MDE is illustrated in a CPS system using Papyrus UML tool.

Ling Shi [72] discuss the existence of strong incentives for unlawful attacks on the CPSs. For instance, economic benefits (e.g., Reduce water bill, Not Pay water bill). Latest CPSs uses wireless communications, LAN for information transfer and administrative control. This advancement increase security loopholes and then the attacker will able to go ahead with various types of popular attacks.

Sandeep K Shukla et al. [73] discuss CPS design methodology usually following a model-based engineering approach. Following are steps to design a CPS:

1. Mathematics based model of the Real life system
2. Theoretic model of the control system
3. The Abstract model of target CPS
4. The Implementation model of CPS

Prasanna Kannappan et al. [60] demonstrated how the resilience of supervisory CPSs will improve through machine learning. During regular operation, individual agents learn from their supervisor's commands. These learned agents will able to recover the system in case of administrator fails or communication to the slave is down.

Kaiyu Wab et al. in [3] studied CPS security to recognise issues in security control, and investigate them to use in the improvement of security and survivability of CPS.

Marco Rocchetto et al. in [65] design a framework to classify attacker models and show that those profiles cover most types of attackers.

Anam Sajid et al. [7] outline security loopholes of SCADA systems in an Internet of Things based cloud system. Older SCADA systems are already facing loopholes in security. Internet of Things based cloud computing enable SCADA systems are going to increase security and deployment issues and require more research effort [7].

Avik Dayal et al. in [14] develop a virtual backend known as Virtual SCADA or VSCADA. VSCADA simulate field device at the physical layer of the SCADA. They use HMIs in order to access the data related to simulation from OPC server on the computer.

Avik Dayal et al. [15] describe a virtual distributed testbed that can be reconfigured to simulate the SCADA of a power system, or a transportation system or any other critical systems provided a back-end domain specific simulator for such systems are attached to it [15].

Domenico Cotroneo et al. in [23] analyze the principles behind modelling and evaluation approach to support a variety of design for security monitor and control of the system.

Vincent Urias et al. in [82] at Sandia National Labs has introduced a concept of emulated machines in cyber-security analysis in addition to the current use of physical hardware, and virtualization.

Chee-Wooi Ten et al. in [20] proposes a risk assessment of SCADA systems at system, scenarios, and entry points. It also relies on the attack model, firewall and password models. These model are basic modelling of security for SCADA based CPS.

Yilin Mo et al. [86] use MDE to detect integrity attacks on the sensors of SCADA systems.

Frank Jiang et al. [34] discuss data acquisition from the underwater acoustic sensory protocol. This protocol is based on unique underwater features and is able to achieve significant throughput (packet delivery ratio) with lower communication overhead.

Bela Genge and Christos Siaterlis [28] discuss a framework for virtual experiment with the Smart Grid for evaluating the effects of cyber attacks on the Smart Grid.

Siddharth Sridhar et al. [74] describe the importance of cybersecurity infrastructure in power grid to prevent and tolerate cyber attacks on the power grid.

Yang Yanjiang et al. in [85] proposed security solutions in ICS with the design of lightweight cryptographic primitives and they also propose to outsource time-consuming computations from field devices to nearby devices with higher computational capacity.

Carlos Romas et al. in [19] propose to use Cyber-Physical Intelligence in Intelligent SCADA systems at various levels of Power System, from the Generation, Transmission, and Distribution.

Peeyush Jain et al. in [33] provides an overview of security issues and threats in SCADA networks.

Advancement of an attack undergoes different levels before success. These levels are: begin, activate, entry into the root, network proliferation, and device breakage. These levels may be formalized with a model like Cinco Model [4], Petri-nets [20], or UML [8]. The success of each initial level impacts the probability of success at the next level. For example, the chance to get entry as superuser depends on the progress at the activated level. The password model illustrated as Fig. 1.4 is based on two different status nodes and two different transitions. Solid bar and empty bar are used to model the attempt of invasion and response to attack respectively. To model defence like account temporary disables based on fix number of attempts which is simulated with the $N(\text{password policy threshold})$ tokens using Petrinet.

Raspberry Pi was only one controller in the water management Cyber-Physical System [17]. In this work, we are harnessing the benefit of both Raspberry Pi and Arduino. We have used only one flow sensor in WM-CPS [17]. In this work, we are going to use both the level sensor and

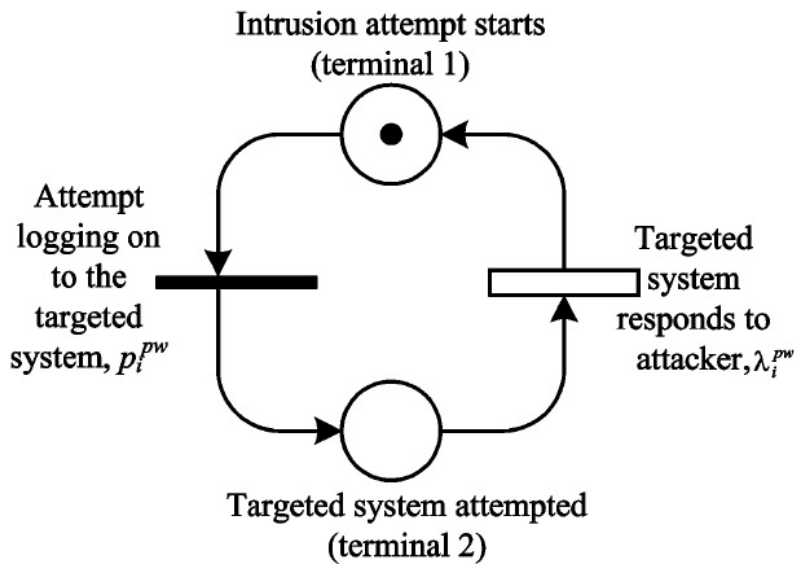


FIGURE 1.4: Password Model [20]

flow sensor. Ethernet and Wi-Fi are communication standards used for the communication between sensor, actuator and controller in WM-CPS [17]. In this work, we are using the serial communication standard. The Wireless sensor network system is developed using both open source hardware platform, i.e., Raspberry Pi and Arduino for environmental monitoring applications [26]. In drip irrigation system, water slowly drips to the roots of the plants through narrow tubes and valves. For that, Arduino microcontrollers are used to receive the on/off commands from the raspberry pi using ZigBee protocol. The use of ultrasound sensors and solenoid valves make a smart drip irrigation system [53]. Andrew K Dennis written a book called “Raspberry Pi home automation with Arduino.” In his book, he has developed many home automation project with a combination of RPi and open source Arduino hardware platform [22]. The fire alarm monitoring system is a real-time system that detects the presence of smoke in the air due to fire and captures images via a camera installed inside a room when a fire occurs with the help of Raspberry Pi and Arduino Uno [48]. The system monitor was surrounding weather conditions including humidity, temperature, climate quality, and also the filter fan switch control in the chicken farm using Raspberry Pi and Arduino Uno [35]. A novel, low-cost autonomous vehicle platform based on a combination of a Raspberry Pi mini-computer, an Arduino micro-controller, and a Zumo track-driven robot chassis is developed [39]. A robotic agent platform is developed that embed software agent into a hardware device like Raspberry Pi and Arduino [43]. In smart city solutions to water management using water sensors, processing of the data from the sensors need a simple development board such as the Arduino [83]. Raspberry Pi has solved the problem faced by the microcontroller with low memory and computation capabilities when dealing with sophisticated algorithms inside the sensor nodes [68]. Architecture design for Smart water management is proposed, and implementation detail of Smart water monitoring system using Raspberry Pi and Arduino is discussed [71].

1.5 NIST Framework for Critical Infrastructure Cybersecurity

TABLE 1.1: Part 1: A selection of NIST cybersecurity-related publications [5]

Publication number	Publication title	Publication date
800-12	An Introduction to Computer Security: the NIST Handbook	October 1, 1995
800-14	Generally Accepted Principles and Practices for Securing Information Technology Systems	September 1, 1996
800-27 Rev.	An Engineering Principles for Information Technology Security (A Baseline for Achieving Security), Revision A	June 4, 2015
800-30	Guide for Conducting Risk Assessments	September 12, 2015
800-34	Contingency Planning Guide for Federal Information Systems	May 10, 2015
800-35	Guide to Information Technology Security Services	October 3, 2015
800-36	Guide to Selecting Information Technology Security Products	October 3, 2015
800-37	Guide for Applying the Risk Management Framework to Federal Information Systems: a Security Life Cycle Approach	February 10, 2015
800-39	Managing Information Security Risk: Organization, Mission, and Information System View	March 11, 2015
800-41	Guidelines on Firewalls and Firewall Policy	September 9, 2015
800-44	Guidelines on Securing Public Web Servers	September 7, 2015
800-45	Guidelines on Electronic Mail Security	February 7, 2015
800-46	Guide to Enterprise Telework and Remote Access Security	June 9, 2015
800-47	Security Guide for Interconnecting Information Technology Systems	August 2, 2015
800-50	Building an Information Technology Security Awareness and Training Program	October 3, 2015
800-60	Guide for Mapping Types of Information and Information Systems to Security Categories	August 8, 2015
800-61	Computer Security Incident Handling Guide	August 12, 2015
800-65	Recommendations for Integrating Information Security into the Capital Planning and Investment Control Process	July 9, 2015
800-82	Guide to Industrial Control Systems Security	May 15, 2015
800-83	Guide to Malware Incident Prevention and Handling for Desktops and Laptops	July 13, 2015

Since 1990, NIST is publishing special publications in the 800 series for the cybersecurity community. Out of those publications, the Framework for Improving Critical Infrastructure cybersecurity (NIST 2014) and the Special Publication 800-82-Guide to Industrial Control Systems Security (NIST 2015) is widely referred.

TABLE 1.2: Part 2: A selection of NIST cybersecurity-related publications

800-92	Guide to Computer Security Log Management	September 6, 2015
800-94	Guide to Intrusion Detection and Prevention Systems (IDPS)	July 12, 2015
800-100	Information Security Handbook: A Guide for Managers	October 6, 2015
800-114	User's Guide to Securing External Devices for Telework and Remote Access	November 7, 2015
800-122	Guide to Protecting the Confidentiality of Personally Identifiable Information (PII)	April 10, 2015
800-128	Guide for Security-Focused Configuration Management of Information Systems	August 11, 2015
800-153	Guidelines for Securing Wireless Local Area Networks (WLANs)	February 12, 2015
800-160	Systems Security Engineering Guideline	May 12, 2014
800-161	Supply Chain Risk Management Practices for Federal Information Systems and Organizations	April 15, 2015
800-171	Protecting Controlled Unclassified Information in Nonfederal Information Systems and Organizations	June 15, 2015
SP 800-175B	Guideline for Using Cryptographic Standards in the Federal Government: Cryptographic Mechanisms	August 2016
SP 800-176	Computer Security Division 2014 Annual Report	August 2015
SP 800-177	Trustworthy Email	September 2016
SP 800-178	A Comparison of Attribute Based Access Control (ABAC) Standards for Data Service Applications: Extensible Access Control Markup Language (XACML) and Next Generation Access Control (NGAC)	October 2016
SP 800-179	DRAFT Guide to Securing Apple OS X 10.10 Systems for IT Professionals: A NIST Security Configuration Checklist Announcement and Draft Publication	June 2016
SP 800-181	NICE Cybersecurity Workforce Framework (NCWF): National Initiative for Cybersecurity Education Announcement and Draft Publication	November 2016
SP 800-182	Computer Security Division 2015 Annual Report	July 2016
SP 800-183	Networks of 'Things'	July 2016
SP 800-184	DRAFT Guide for Cybersecurity Event Recovery Announcement and Draft Publication	June 2016
SP 800-185	DRAFT SHA-3 Derived Functions: cSHAKE, KMAC, TupleHash, and ParallelHash	August 2016

In Table 1.1 and 1.2, we have listed a small selection of NIST publications which have particular relevance to the two aforementioned NIST publications. A comprehensive list of NIST CSRC publications including specialist

1.6 Italy's National Strategic Framework for Cyberspace Security

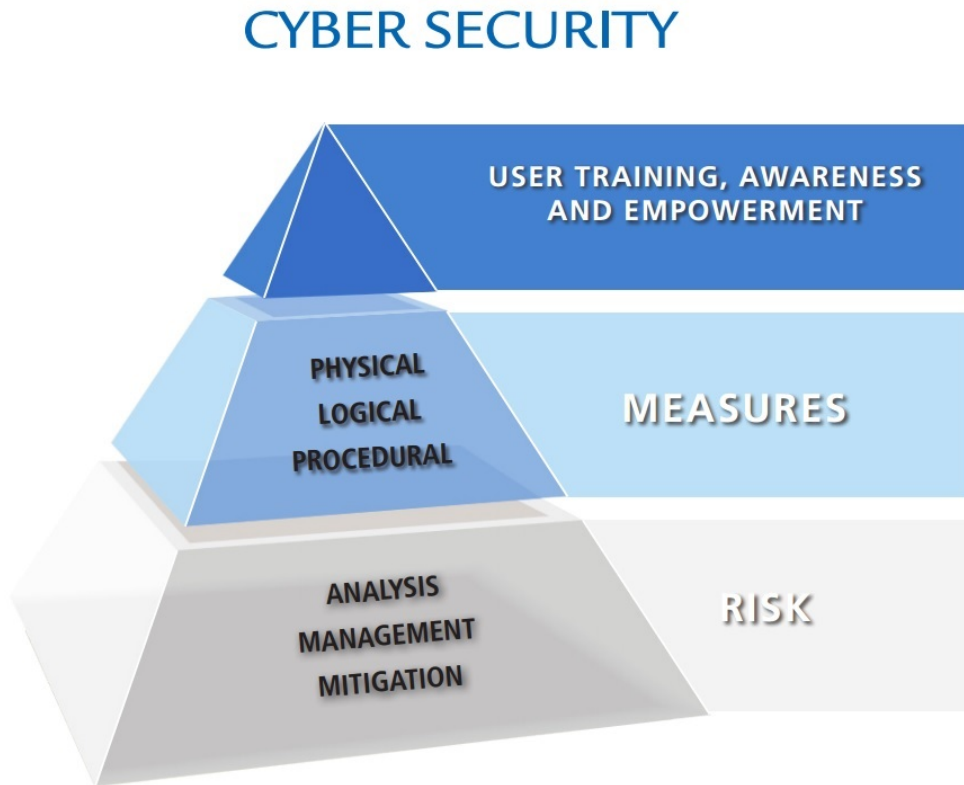


FIGURE 1.5: Italian CyberSecurity Framework [61]

There are six strategic and eleven operational guidelines in Italy's National Strategic Framework for Cyberspace Security. This framework covers user training, risk and measures as shown in Fig.1.5. Following are six strategic guidelines:

1. The improvement of the technical, operational and analytic power of all institutions worried about cybersecurity, to influence the national capability to analyse, avoid, mitigate and react to the multi-dimensional cyber threat.
2. The capacity building to protect critical infrastructure and strategic assets from cyber attacks, with the final goal to ensure their business processes and the full agreement with the global requisite, security standards and protocols.
3. The assistance for all public-private establishment to promote the security of intellectual property and technical innovations.
4. The promotion of the culture of security among citizens and institution, also leveraging the expertise of academia, to raise awareness of the cyber threats among users.
5. The reinforcement of our power to effectively contrast online criminal activities, in compliance with national and international norms.

6. The backing of international cooperation in the field of cyberspace, with particular attention to initiate underway in the international organisations of which Italy is a member with its Allies.

1.7 Our Contribution

1. We perform a case study of an existing water management cyber physical system and get the idea about how this water supply system was, and how in future it will be?
2. Reviews of major Communication Protocols in using in water management cyber physical system, where Sensors and Actuators are stationed. Then we chose reliable components so that the emulator could withstand the harsh operating conditions for nearly three years before surrendering to the elements.
3. We have done a systematic review and design different attacks strategies for Emulator of water management cyber physical system (WM-CPS).
4. We find an efficient way of Serial Communication among Controllers, Sensors and Actuators in Emulator of water management cyber physical system (WM-CPS).
5. We implement RS485 Modbus and TCP/IP communication protocol for communication among sensors, actuators, slave controller and master controller in Emulator of water management cyber physical system.
6. EPANET is a Water Quality Simulator. Before going to invest in sensors and actuators, we simulated our emulator on EPANET.
7. We organised the component of Emulator in a circle. Four subsystems with slave controller had been tied around the circumference of the circle, to act as a whole emulator. The sensors and actuators were stationed inside the subsystem. And the master controller was at the centre of the circle. Sensors first, slave controller seconds and the most vulnerable master controller at the inside.
8. We integrated five SECUBE in the emulator. One SECUBE with every slave controller and one SECUBE with the master controller. SECUBE encrypted sensors readings from the slave and send to master controller, where it was decrypted. SECUBE encrypted actuators command from master and transmit to the master controller, where SECUBE attached with slave controller again decrypted it.
9. We design the web interface of our system with domain <https://cps4wm.info/>. The web interface of emulator shows the sensors reading and status of actuators. Also, we design a bot that runs in Telegram. Where we can execute commands in Android Mobile Phone.

10. Nothing is perfect, and nothing is absolute. There are chances for improvement every time. Integration of Niagara router in our emulator and compatibility of our emulator with Cyber Range is future work that we are planning to do at postdoctoral stage.

1.8 Dissertation Organization

This chapter presents an overview of prior works and challenges behind the design. The remaining of this dissertation is organised as follows.

Chapter 2 presents a Case Study of Existing Water Management Cyber Physical System: EVA System. This system supports Dulcometer, and four individual sensors called VEGABAR, VEGAWELL, VEGASON and OPTIFLUX 2000. Dulcometer is a single device with inbuilt 7 ION Sensors and 4 Physical sensors. The following actuators are also in use in EVA system: Danfoss Frequency Converter, Danfoss Softstarter, Caprari Pump, Belimo Rotary Actuator (GR24A-SR-5), IFC050/100 Converter.

Chapter 3 presents a survey of Communication Protocols used for communication among Sensors and Actuators in the Water Management Cyber Physical System (WM-CPS). In our work, various communication standards like Ethernet/IP, Modbus/TCP, Modbus/ RTU, Wi-Fi 802.11n/IP, Wi-Fi 802.11n/UDP, and Wi-Fi 802.11n/TCP were taken under consideration. We also surveyed the different variety of flow sensor, temperature sensor and level sensor. We also studied the evolution of water pump from Archimedes water screw in 287-212 BC to Submersible sewage pump of 1956 AD.

Chapter 4 presents a systematic review and Design of Different Attacks Strategies for the Water Management Cyber Physical System. In water management System (WMS), a hacker manipulates all chemical and physical characteristics of drinking water with compromising of both sensor and actuators. This hacking creates threats to human life. To deal with, experts take countermeasures to secure their system. But the destructive forces continue the process of insertion, modification, and an abridgement in their current attack strategies and cybersecurity expert discovered that his countermeasure is fell short of a fact in the present scenario. Therefore, countermeasures have to update itself incrementally to survive. In 1994, the first cyber security attack on water system took place. In this work, seven incidents of cyber terrorism after 1994 on WMS is analysed. We have listed other physical attack on WMS before 1994. We have also developed an emulated WMS. Our emulated WMS consists of different sensors, actuators, and web-interface. Information disclosure, forced browsing, SQL injection and denial of service are four different attack strategies taken into consideration in this chapter. We proposed a solution for the loopholes, we found in emulated WMS. Finally, this chapter analyzes the compatibility of our solution with recommended cybersecurity practices developed by the American Water Works Association (AWWA).

Chapter 5 presents an overview of Serial Communication among Controllers, Sensors and Actuators in Water management Cyber Physical System (WMCPS). Sensors and Actuator Network (SAN) is a viable solution to many innovative applications. Water Management Cyber-Physical System (WMCPS) is one of them. This chapter 5 described a SAN that we have developed using open-source controllers, i.e., Arduino and Raspberry Pi. This system is using serial communication for communication between controllers. Raspberry is a master controller and Arduino is slave controller. All sensors and actuators are connected to Arduino. Raspberry read sensor information from Arduino and pass a command to Actuator connected to Arduino. Flow sensor and Level sensor are used in this work to measure the rate of flow of water and level of water in the water tank respectively. Pump and relay are actuators used to fill the water tank. Raspberry execute a command on Arduino for starting and stopping of the pump using serial communication. Command passed from Raspberry is based on the click of ON/OFF Pump button on <http://www.cps4wm.info>. Reading of flow sensors and level sensors along with on/off command for Pump stored in MySQL database.

Chapter 6 presents an Implementation of RS485 Modbus and TCPI/IP in Water Management Cyber Physical System. In this chapter, TCP/IP, RS485, UART, and USB are four different interfaces used for Modbus communication in our Water Management Cyber-Physical System (WM-CPS). Arduino and Raspberry are two controllers. Arduino connects with the UART to RS485 Modbus converter. Raspberry Pi (RPi) connects with USB to RS485 Modbus converter, and it is recognised as `/dev/ttyUSB0` in the Rasbian operating system. When Arduino needs to transmit sensor reading, we connect `TE_EN` of RS485 module to VCC. When Arduino has to receive a command for the actuator, we connect `TE_EN` of RS485 module to GND. When Arduino gets then RPi transmits and vice versa. RPi uses serial python library. Formulation of actuator command is based on the click of "TURN PUMP ON/OFF" HTML button on the home page of <http://cps4wm.info> written in PHP. Our system connects with the internet using eduroam available on our University Campus.

Chapter 7 presents a simulation report of water management cyber physical system using EPANET simulator before designing an emulator of it.

Chapter 8 presents a design report of an emulator for Water Management Cyber Physical System: using Raspberry Pi without SECUBE.

Chapter 9 presents a report on the design of emulator for Water Management Cyber Physical System: using Raspberry Pi with SECUBE.

Chapter 10 presents the design steps of Web Interface for an emulator of water management cyber physical system. It also describes the design of Telegram based Bot that run on the Android phone as shown in Fig. 1.6.

Chapter 11 presents the possibility of extension of our system with Cyber Range and Niagara in the near future.

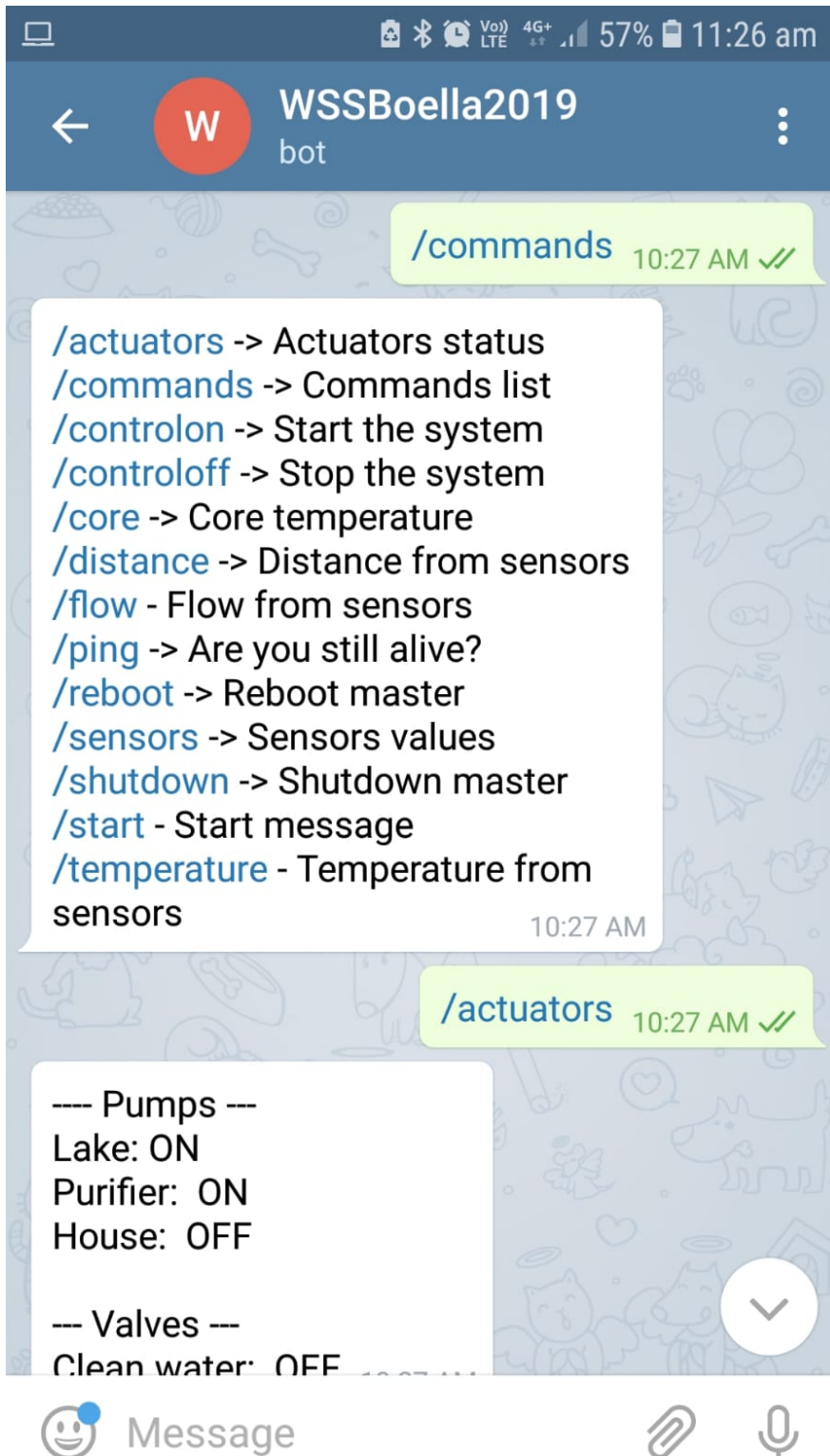


FIGURE 1.6: Telegram Bot of Our System Working on Android Phone

*Nothing is softer or more flexible than water, yet
nothing can resist it.*

Lao Tzu

2

Case Study of Existing Water Management Cyber Physical System

2.1 EVA System

HAWK CPU (Powered by Niagara Framework) is used to read the value of following sensors (as shown in Fig. 2.1 and Table 2.1) employed in EVA with the help of JACE router and multiflex IO Board:

- VegaWell: Normal Sensor Used For Level Measurement of Water.
- VegaBAR: Monitor Incoming Pressure Of Water In Pipe
- VegaSON: Ultrasonic Sensor Used For Level Measurement of Water.
- Dulcometer: Measures pH, ORP, Chlorine, Conductivity, Fluoride, Temperature
- OPTIFLUX: Monitor Rate of Water Flow Based on the reading of sensors, it triggers following actuators (as shown in Fig. 2.1 and Table. 2.1):
- Soft Starter: Reduce Load and Torque of an AC Motor during Start-Up
- Frequency Converter: Used For Maintenance of Net Pressure in Electric Pump
- Rotary Actuators: Maintain Desired Water Flow

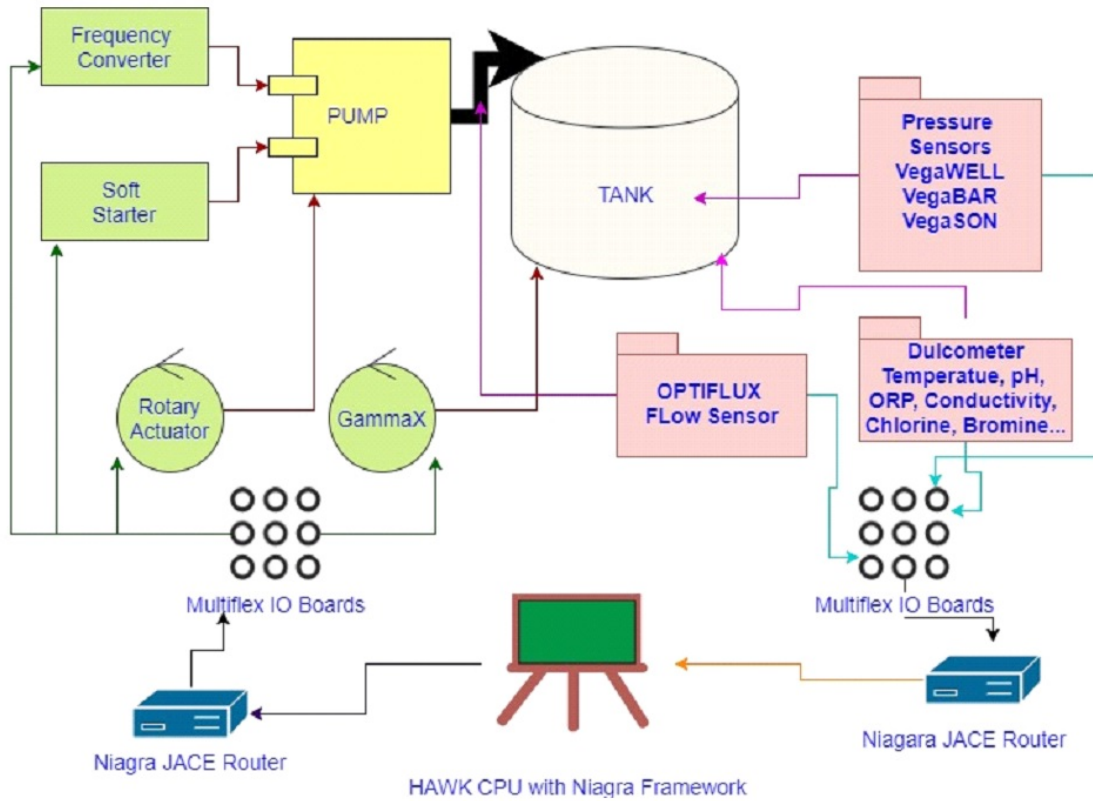


FIGURE 2.1: Overview of EVA System

- GammaX: Magnetic Dosing Pump Used For Disinfection of Contamination
- Pump: Fill the Water Tank

TABLE 2.1: Components of EVA System

Actuators	Sensors	Controller	Software
PUMP	VegaWELL	HAWK CPU	Niagara Framework
Frequency Converter	VegaBAR	JACE Router	Easy LINK
Soft Starter	VegaSON	Multiflex IO Boards	Energy LINK
Rotary Actuators	OPTIFLUX		
GammaX	Dulcometer		

2.2 System Development Life Cycle of EVA System in Perspective of System Engineer

EVA system is a water management system. This system is working in a suburban area of Italy. The Niagara framework powers it. This system supports four individual sensors called VEGABAR, VEGAWELL, VEGASON and OPTIFLUX 2000. It also supports Dulcometer. Dulcometer is a single



FIGURE 2.2: Business Process Model of EVA System

device with inbuilt 7 ION Sensors and 4 Physical sensors. Along with these sensors, the following actuators are also in use in EVA system: Danfoss Frequency Converter, Danfoss Softstarter, Caprari Pump, Belimo Rotary Actuator (GR24A-SR-5), IFC050/100 Converter. To nullify the effect of both chemical and biological contamination in water, this system also administers the mixing of implants in water. The main functionality offered by the EVA system is following and also shown in Fig. 2.3:

- Reading of Sensors
- Send commands to the actuators, manual or automatic.
- View and store information about the water supply.
- General diagnosis of chemical and biological contamination
- Administer proper dosage of implants to neutralise the effect of contaminants.
- Centralisation of states, alarms, measurements, and controls.
- Self-diagnosis of the control system.
- Scanning of EVA System and then Enumeration of security loopholes to make it more secure network.

Following are some essential characteristics features of EVA System:

- Maximum system capacity: 360m³/h.
- Measured quantities: pH, Redox, Bromine (Br), Chlorine (Cl₂), Chlorine Dioxide (ClO₂), Chlorite Ion (ClO₂⁻), Conductivity, Dissolved Oxygen, Fluoride, Hydrogen Peroxide, ORP, Ozone (O₃), Peracetic acid, and pH. For example, ensure the density of the measured substance like sodium hypochlorite: 1.2 g/ml, free chlorine concentration need to 0.20ppm.
- Power supply by solar system or by any other way in the submerged part of a water plant.
- Thermal compensation for pH and process sensor for chlorine dioxide CDP, compensation pH for chlorine.
- Digital inputs for processing control signals, such as water level, remote monitoring, and contamination level monitoring.
- Control outputs for dosing pumps and electronic, electromagnetic valves.
- Disturbance processing: simple adjustment of water parameters in running water flow.
- Possibility of adjusting the nominal value of the controller when changing process conditions via either the remote control or the mA signal of a PLC.

PUMP is also an actuator, and its working is based on two actuators called soft starter and frequency converter also shown in Fig. 2.4:

System development life cycle (SDLC) begins with the recognition of requirement. Identification of necessity starts with a preliminary investigation.

2.3 Preliminary Investigation

This existing EVA System is developed for the management of water supply, and its advantages and disadvantages were studied in depth. We made the following assumption after preliminary investigation.

- This system is supporting some ION sensors available in DULCOMETER (Measured variables: Bromine (Br), Chlorine (Cl₂), Chlorine Dioxide (ClO₂), Chlorite Ion (ClO₂⁻), Hydrogen Peroxide, and Ozone (O₃))
- This system is capable of measuring pressure, flow rate, water level, pH, conductivity, ORP, and temperature of the water.

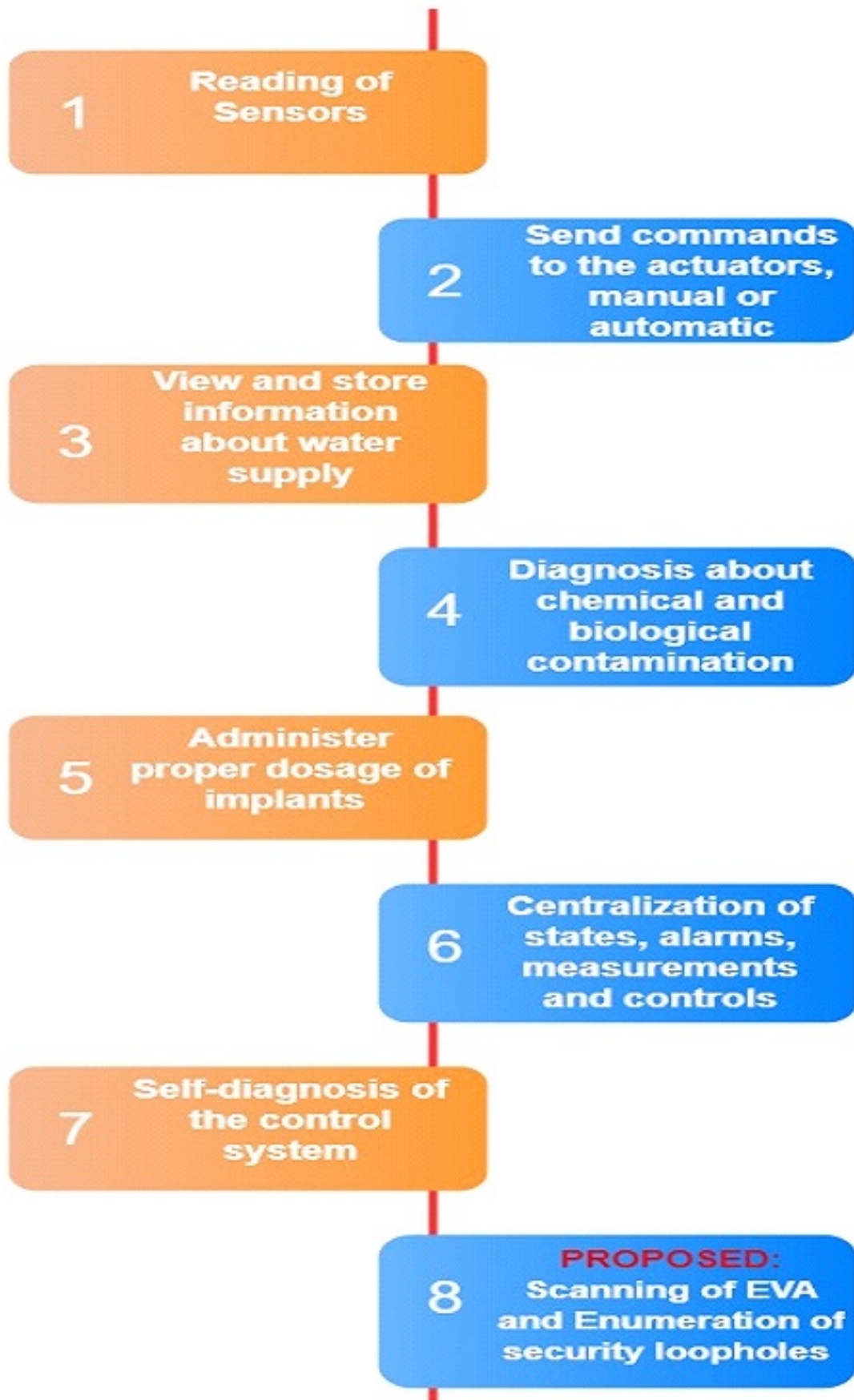


FIGURE 2.3: Functionality of EVA System

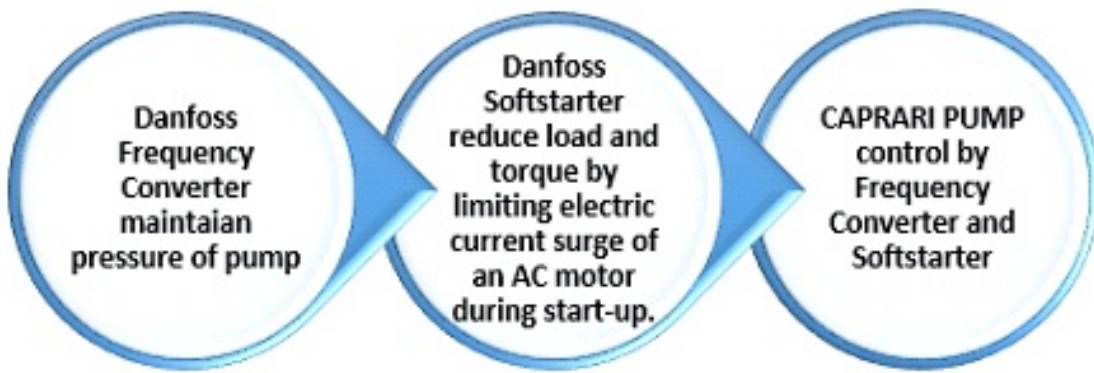


FIGURE 2.4: PUMP as an Actuator

- Following actuators are also in use in EVA system: Danfoss Frequency Converter, Danfoss Softstarter, Caprari Pump, Belimo Rotary Actuator (GR24A-SR-5), IFC050/100 Converter.
- To nullify the effect of both chemical and biological contamination in water, this system also administers the mixing of implants in water with the usage of GammaX.
- This EVA system is using eight different E6-8 series PUMPs. CAPRARI manufactures all pumps. These are E6RX, E6SX, E6PX, MACX6-2A, MACX6-2B, MACX8, HVU18%50, HV65-80.
- Following are three types of soft starter available on the website of Danfoss: MCD 500, MCD 201/202 and MCD 100. In the given report, they have not mentioned that which soft starter, they are using?
- Are they using BACnet or LonWorks or Modbus?
- Is WAN Wireless Area Network or Wired Area Network?
- Among 20 protocols, which protocol is used by EVA System? If the protocol is LonTalk then which Neuron chip EVA system is using?

2.4 Feasibility Study

A valuable result of the preliminary investigation is the assurance that the proposed system is either feasible or not feasible. The feasibility study is an assessment of the project according to its usefulness, impact on the institution, the ability to fulfil user expectations and optimal use of resources. The objective of the feasibility study is to get the solution to the problems that will apply in a broader context. There are three prime phases of feasibility study :

- Technical Feasibility

- Operational Feasibility
- Economic Feasibility

2.4.1 Technical Feasibility

EVA system display real-time consumption of water, its sensor reading, and status of actuators and controllers with the help of Niagara Framework.

2.4.2 Operational Feasibility

It is a method to determine an estimate of how satisfactory response that user community, supervisor of EVA is likely to have in the EVA System. In case of EVA System, there is not any requirement of proper training for user community as their essential work related to water, e.g., drinking, and washing are not going to be affected by addition in EVA system. The user community will welcome the changes if they get into notice that their water supply system is now more secure than the previous one.

2.4.3 Economic Feasibility

It is the most frequently used method for estimating the investment required for the system. It is also called the cost-benefit analysis. During this study, the profit anticipated from the project compares with its manufacturing cost. If there is a chance of profit-making, then it is feasible to implement it. Current EVA system has budgeted for all essential sensors and actuators. Therefore, our EVA system is a doable project from an economic perspective.

2.5 System Development Life Cycle of EVA System in Perspective of Computer Engineer

Computer engineer thinks in terms of waterfall model for system development life cycle. The waterfall model is based on a procedural, sequential approach to system development that starts with the Requirement. It has the following 6 phases, as shown in Fig. 2.5: Let us suppose; this EVA System has used the Waterfall model.

2.5.1 Phase 1: Requirement

Requirement phase is divided into two sub-phases. One is conception, and other is initiation.

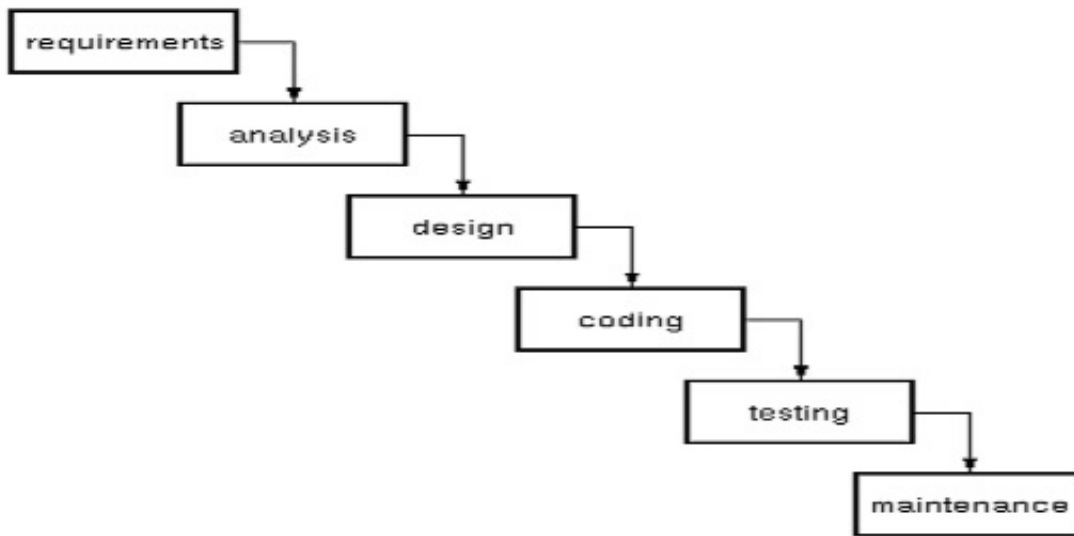


FIGURE 2.5: System Development Life Cycle of EVA System

- Conception: This phase is focused on the description of the aim, underlying problem, objectives, goals, targets and possible benefits from the EVA system.
- Initiation: The exact needs of the client are recorded in this phase. This record in the form of a document is known as System Requirement Specification (SRS). SRS is a blueprint that describes "What" the system under design will do?

2.5.2 Phase 2: Analysis:

In this phase, we analyse the following parameters:

- Inputs of the EVA System
- Outputs of the EVA System
- The process to get the outputs from the given set of data.
- Reports like Chemical Composition report, Contamination report, Log of Sensor Reading, Log of Actuators Command.
- Permissible Criteria

The flowchart illustrates a step by step solution to a given problem. Specific boxes represent every step. These boxes are connecting by the arrow in a given order. Flow Chart to on/off PUMP to fill the tank of Eva system is shown in Fig. 2.6.

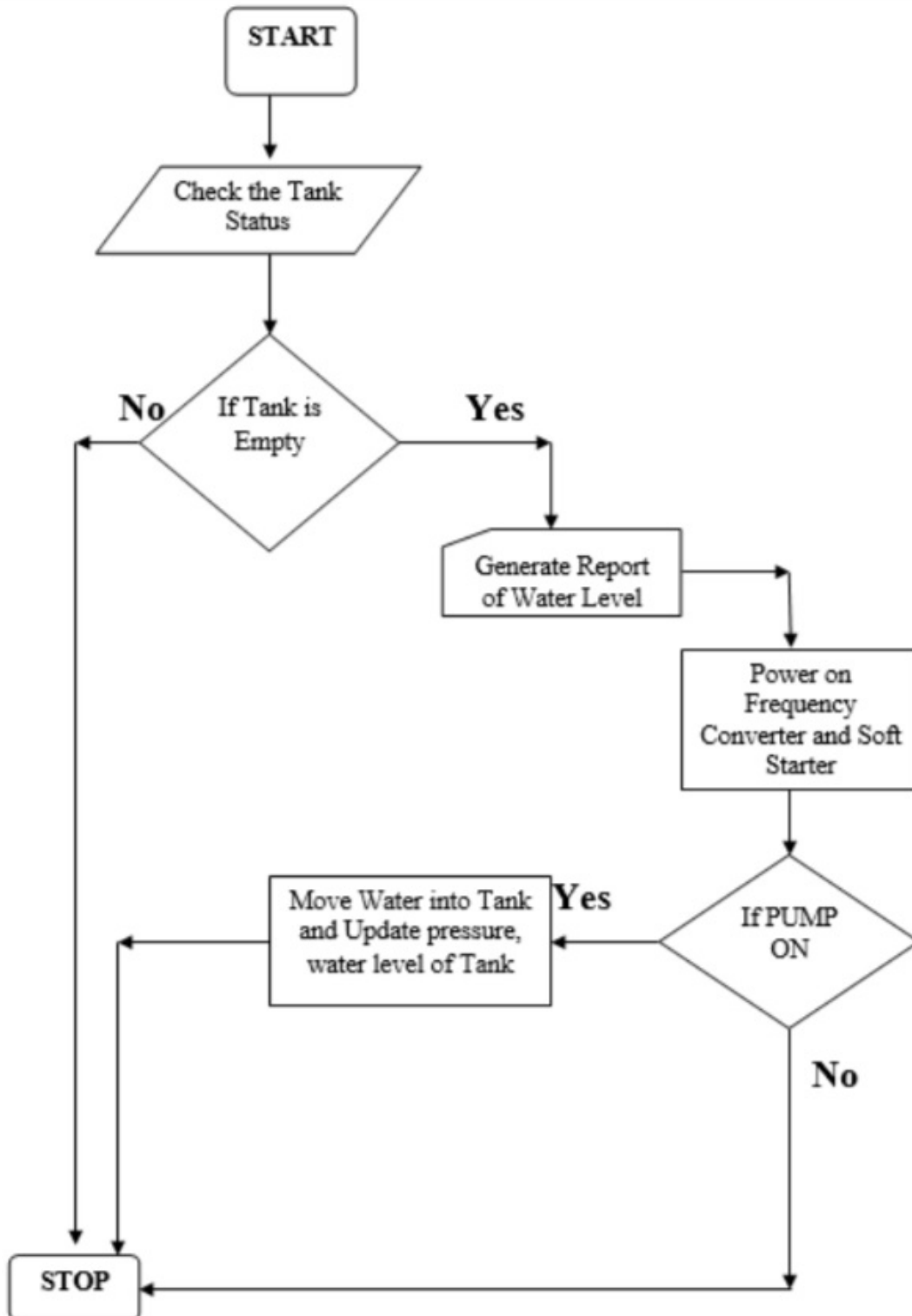


FIGURE 2.6: Flow Chart to Start/Stop PUMP to Fill Tank of EVA System

Flow Chart for checking contamination in drinking water supplied by the EVA system is shown in Fig. 2.7. This flowchart also explains the steps to insert the proper dosage of implants to neutralise both chemical and biological contaminant and make water potable again.

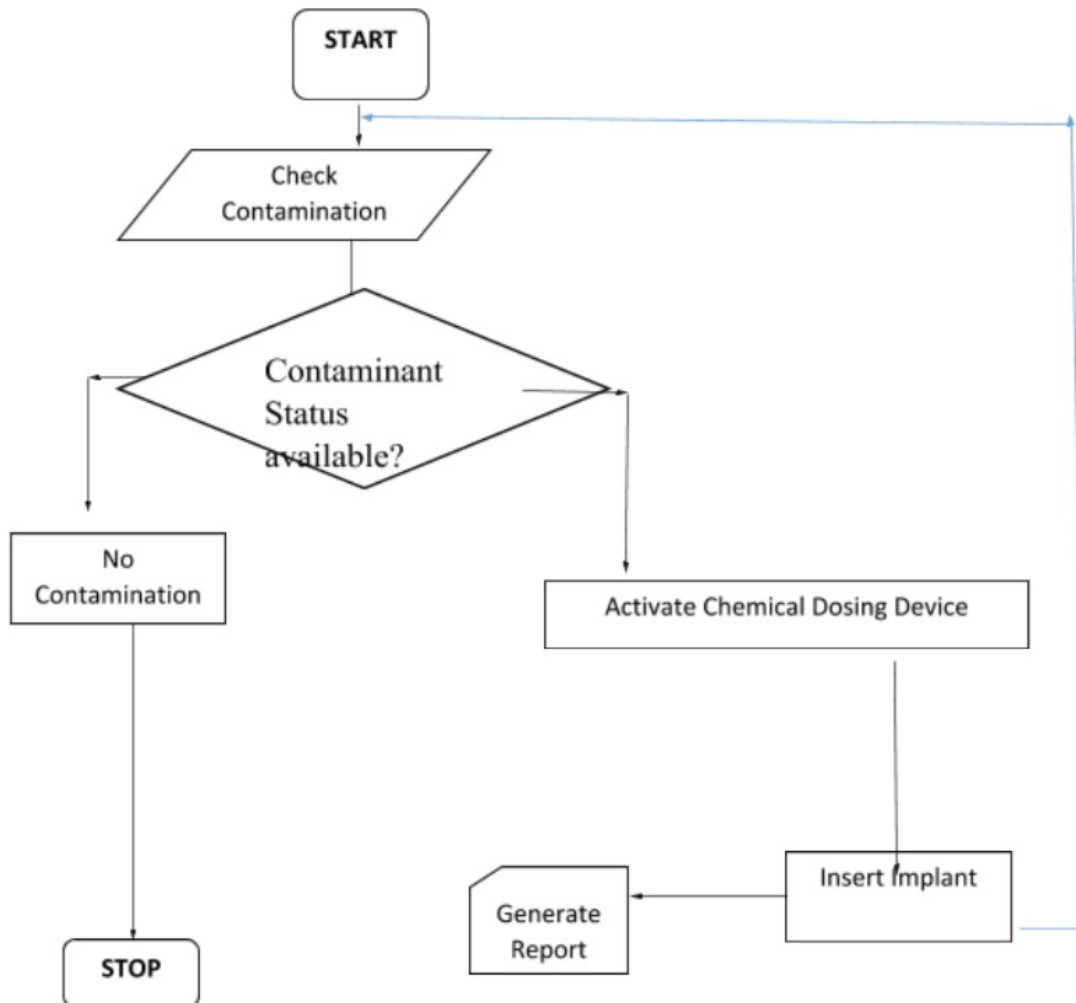


FIGURE 2.7: Flowchart for checking contamination in drinking water supplied by EVA system

Flow Chart for generating a chemical composition of drinking water supplied by the EVA system and also getting the reading of sensor is shown in Fig. 2.8. This flowchart also shows the steps to compare sensor reading and permissible reading for safe drinking water.

Flow Chart for passing commands to actuators in the EVA system is shown in Fig. 2.9. This flowchart also shows the steps to deal with both manual and automatic control of actuators and generating the report for analysis the effect on the water for the execution of a command on actuators.

Flow Chart for displaying the current status of actuators, sensors in the EVA system is clear from Fig. 2.10.

An Entity Relationship Diagram (ERD) describes how entities (either concepts or things) related to each other. ERD of EVA System is shown in Fig. 2.11.

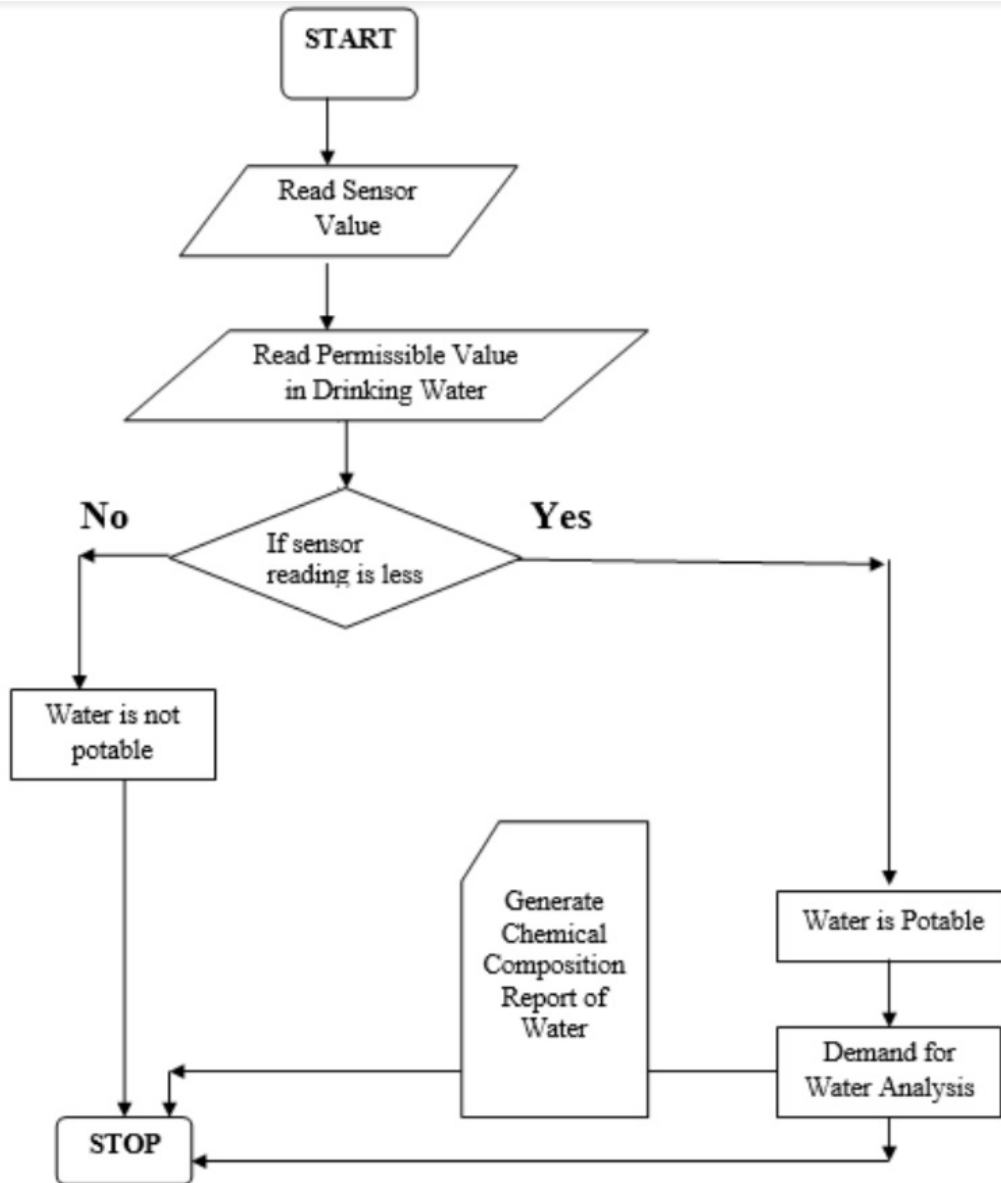


FIGURE 2.8: Flow Chart of Deciding Water is potable or not Based on Sensor Reading in EVA System

2.5.3 Phase 3: Design:

The main goal of this stage of SDLC is to transform the requirements specification into a structure using hardware (sensors, actuators, and controllers) and software (Niagara, EnergyLink, and Easylink). This design phase is already discussed in Section 1 of this report.

2.5.4 Phase 4: Coding:

In this phase, scripts or program must have written for all the modules of the system. These script-s/programs need to able:

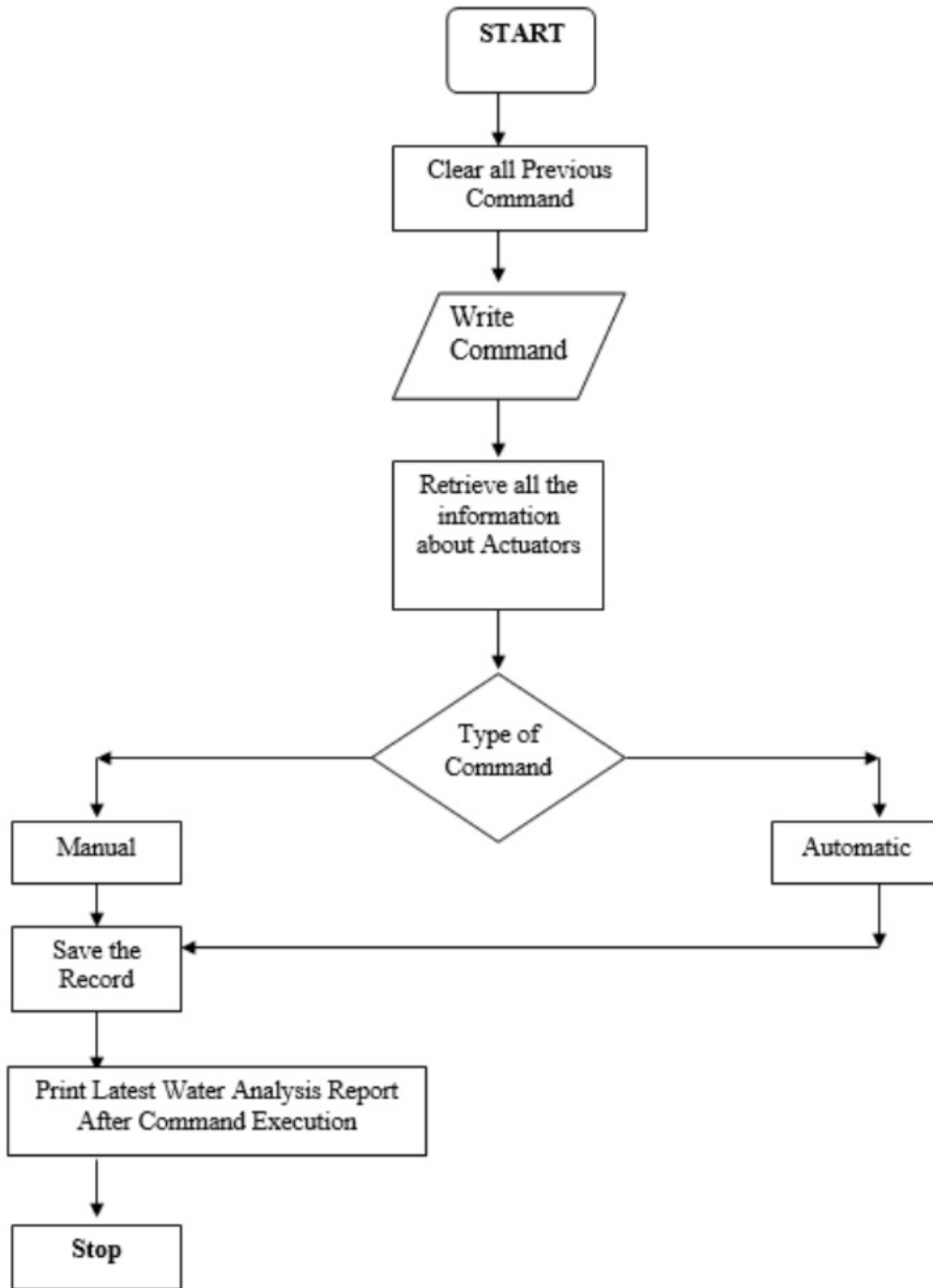


FIGURE 2.9: Flow Chart to Control Actuators of EVA System

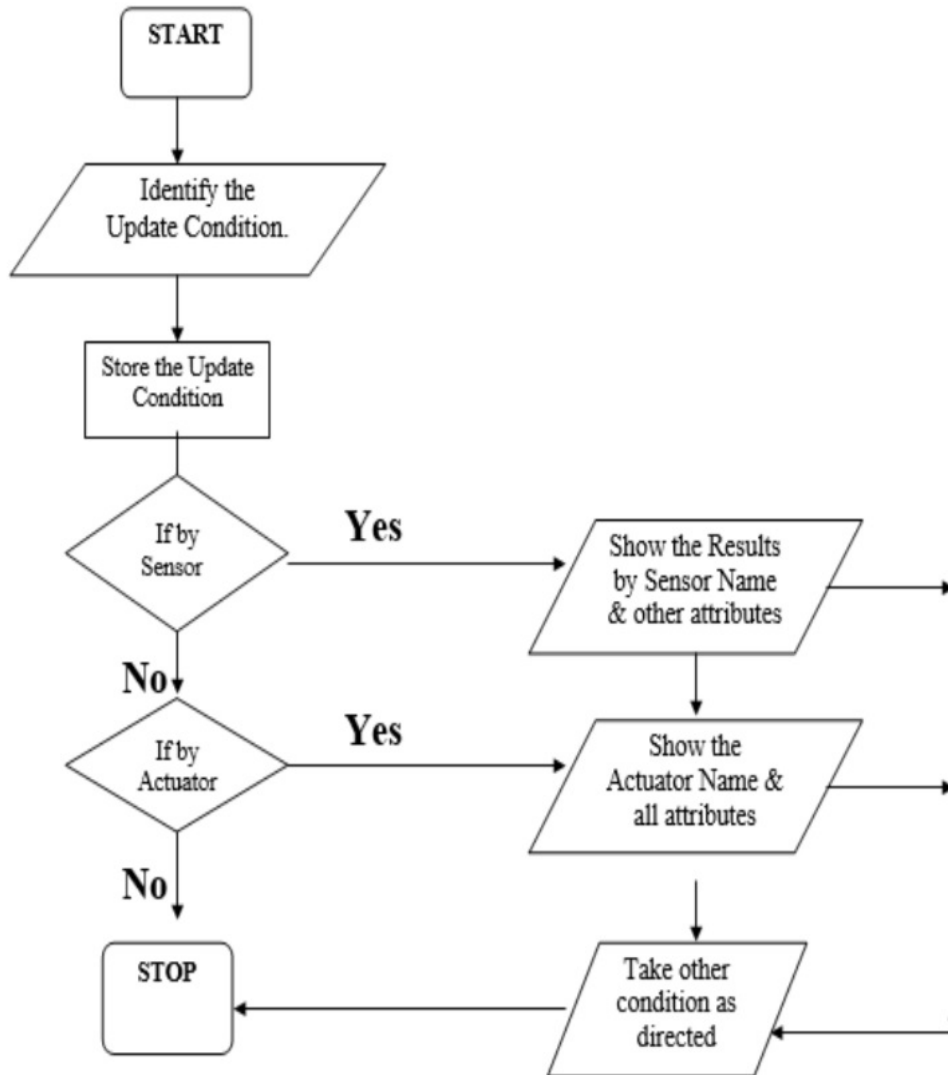


FIGURE 2.10: Flow Chart for Display of Current Status of EVA System

- to read sensor,
- to control actuators,
- to store the value in the database and
- to update the website.

2.5.5 Phase 5: Testing:

Testing for EVA System is power on every component (sensor, actuator and controller) with the intention of finding errors and running scripts to get reading of every sensor and to give the command to every actuator during Unit Testing. The entire EVA system is tested as a whole system during Integration Testing. Since EVA System is a real-time system, one of the main test points will reflect

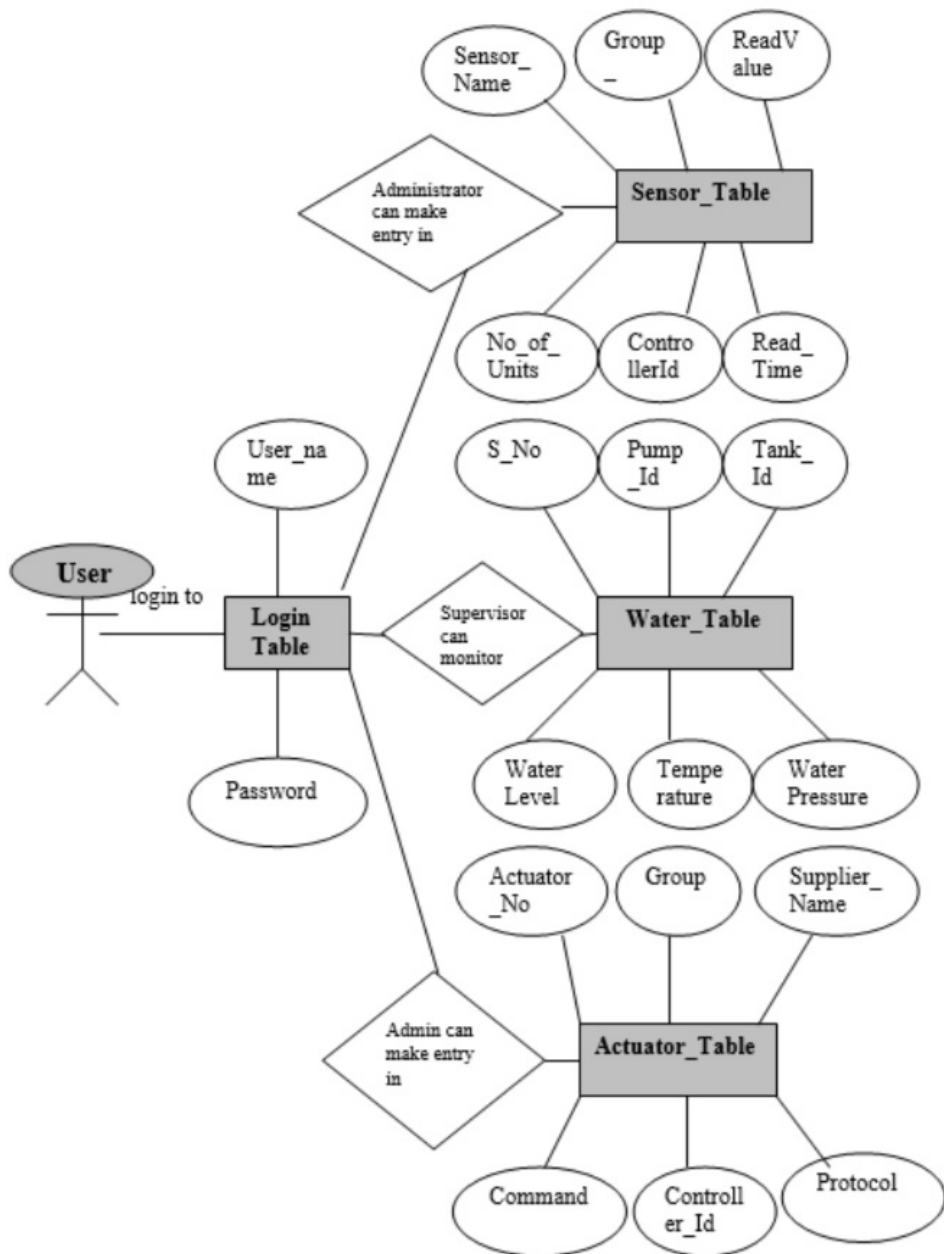


FIGURE 2.11: Entity Relationship Diagram of EVA System

its importance to supply water to its end-users. It should ensure that the response-time to on/off actuator, response-time to get reading of sensors and response-time to activate controllers should be minimal.

2.5.6 Phase 6: Maintenance

Maintenance of EVA System is a continuous process.

- Analysis of EVA System in Perspective of Electrical Engineer: To power on/off all hardware (sensors, actuators, controllers, and so on), we require a proper supply of electricity in the EVA System. If we use energy efficient equipment, then energy efficiency of the EVA system will increase. We can also propose to integrate various energy efficient design techniques in the design of the EVA system. These are clock gating, dynamic voltage scaling, frequency scaling, and output load scaling, and so on.
- Software used in EVA System: Niagara Framework Tool: Honeywell Hawk CPU is a compact platform utilising the NiagaraAX Framework® for the integration of Heating, Ventilation, Air Conditioning (HVAC) systems. In this EVA system, NiagaraAX Framework® is used to control Dulcometer (a single device with inbuilt 7 ION Sensors and 4 Physical sensors). Along with Dulcometer, Niagara is also used to manage Pressure and Water Level Sensor (VEGABAR, VEGASON, and VEGAWELL) and water flow sensor (OPTIFLUX 2000) used in EVA System. In addition to management of sensors network, Niagara is also used to manage Workstation, frequency converter, soft starter, rotary actuators, PUMP, signal converter as is clear from Fig. 2.12.

There are many sensors networks called LonWorks Networks. Every individual LonWorks network has Multiplex I/O boards that receive reading from the sensor and pass control signal to actuators. A group of these LonWorks networks acts on the principle of Wide Area Network (WAN). Niagara controls this WAN..

- Easy Link

EasyLink is a software module through which users can see and interact in real time with systems, to carry out any implementation or forcing of remote devices such as valves or motors. EasyLink relies on the AX platform Web Supervisor and also allows us to easily set up a multi-site and multi-customer supervision system. We are also able to create custom HTML5 views from Queries, tables, and charts. All configuration is done via the web, using the browser. The operations to be performed are automated to avoid human errors or repetitive activities.



FIGURE 2.12: Process Flow Diagram from Sensors to Niagara Framework [18]

- Energy Link

Energy Link is a software package for Analyzing real-time data from the field and also implement an energy saving plan in the water management system.

- Hardware used in EVA System:

For proper management of water supply, treatment and distribution facilities, it is necessary to use various types of the measurement device, transmission device, and historicization system. This EVA system is using many instruments such as frequency converters and soft starter applied to the motors of centrifugal and submersible pumps, pressure transducers from the pipe, suspended pressure transducers, ultrasonic sensors for continuous level measurement, gauges, electromagnetic flow rates, water analysis systems and dosing reagents for disinfection, and remote control systems with various transmission technologies (WI-FI, GPRS, radio frequencies ...). This EVA system also stores all the collected data for complete management of the entire integrated water cycle.

- DANFOSS FREQUENCY CONVERTER
- DANFOSS SOFT STARTER
- CAPRARI PUMP: E6-8 PUMP, E6RX PUMP, E8PX PUMP, MAC6-2C PUMP, MAC6-2B PUMP, MAC6-2A PUMP AND HV65-80 PUMP
- VEGABAR 14
- VEGAWELL 52
- VEGASON 61
- HONEYWELL HAWK CPU
- BELIMO ROTARY ACTUATOR GR24A-SR-5
- ELECTROMAGNETIC MEASUREMENTS: OPTIFLUX 2000 SENSOR AND IFC050/100 CONVERTER
- PROMINENT DULCOMETER
- Prominent GammaX
- Danfoss Frequency Converter (Convertitori Di Frequenza)

Drives/frequency converters and soft starter applied to the motors of centrifugal and submersible pumps. Frequency converters control the electric pumps and mainly used for maintenance of net pressure, as shown in Fig. 2.13. The frequency converter is now installed by default on the new plants but is also useful where we want to improve energy efficiency by reducing fuel consumption by prolonging the life of motors due to less stress during starts and stops. The considerable variation in the daily load in the water treatment plants makes it economically profitable. The next generation VLT AQUA Drive realistically offer cost savings in the first year between 10 and 30

- Danfoss Soft Starter

The use of soft starters for the control of submersible pumps is complete and economical solution for starting and stopping the three-phase asynchronous motors. An extremely cost-effective and compact soft starter provides a total motor starting solution. Danfoss drives soft starters (as shown in Fig. 2.14) are used to reduce the load and torque temporarily by limiting electric current surge of an AC motor during start-up. Following are three types of VLT® soft starter available on the website of Danfoss:

- MCD 500



FIGURE 2.13: Danfoss Frequency Converter [18]

- MCD 201 and MCD 202
- MCD 100



FIGURE 2.14: Danfoss Soft Starter [18]

- Caprari E6-8 PUMP: E6RX, E6SX, E6PX, MACX6-2A, MACX6-2B, MACX8, HVU18%50, HV65-80

Although water distribution works on the principle of gravitation. In some cases, they are using the pumping system when relative pressurisation of some utilities placed at an elevation level equal to or higher than the tank itself. In this report, they have described eight different PUMPs, as shown in Fig. 2.15-Fig. 2.21. It is not clear from a report of the EVA system about the PUMP used in the EVA system.



FIGURE 2.15: 8 Different PUMPs used in EVA System



**E6-8
E6SX**

Pump construction and materials
Construction de la pompe et matériaux
Costruzione pompa e materiali

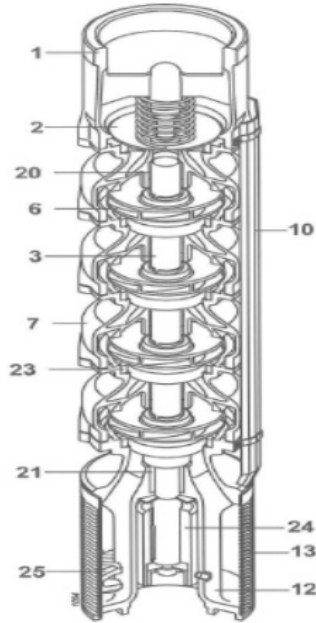


FIGURE 2.16: CAPRARI E6-8 E6SX PUMP [18]

**E6-8
E8PX**



Pump construction and materials
Construction de la pompe et matériaux
Costruzione pompa e materiali

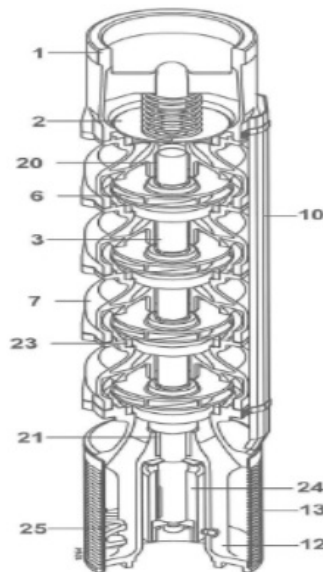


FIGURE 2.17: CAPRARI E6-8 E8PX PUMP [18]



E6-8
E6SX

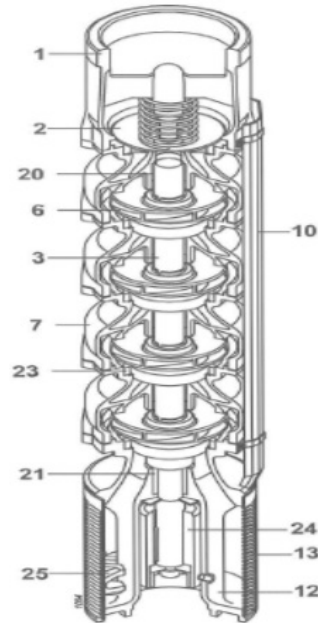


FIGURE 2.18: CAPRARI E6-8 E6RX PUMP [18]

TABLE 2.2: Specification of CAPRARI E6-8 E6SX, E6PX, E6RX PUMP

Position	Parts of E6-8 E6SX	Parts of E6-8 E6PX	Parts of E6-8 E6RX	Materials
1	Valve Casting	Valve Casting	Valve Casting	Stainless Steel
2	Conical Valve	Conical Valve	Conical Valve	Stainless Steel
3	Pump Shaft	Pump Shaft	Pump Shaft	Stainless Steel
5(20)			Shaft bearing bush	Stainless Steel/rubber
6	Impeller	Impeller	Impeller	Stainless Steel
7	Diffuser Unit	Diffuser Unit	Diffuser Unit	Stainless Steel
9			Stage Casing	Stainless Steel
10	Cable Guard	Cable Guard	Cable Guard	Stainless Steel
12	Suction Casing	Suction Casing	Suction Casing	Stainless Steel
13	Strainer	Strainer	Strainer	Stainless Steel
20	Shaft bearing bush	Shaft bearing bush		Stainless Steel/rubber
21	Bearing Bush			Bronze
21			Shaft Bearing Bush	Stainless Steel/rubber
23	Wear Ring	Wear Ring	Wear Ring	Stainless Steel/rubber
24	Coupling	Coupling		Stainless Steel
25	Defender	Defender	Defender	-



Motor construction and materials
Construction du moteur et matériaux
Costruzione motore e materiali



E6-8
MACX6-2A

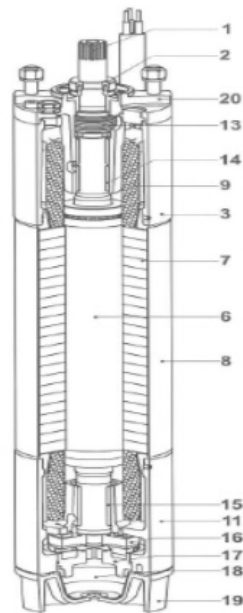


FIGURE 2.19: Caprari E6-8: MACX6-2A PUMP [18]

E6-8
MACX6-2B

Motor construction and materials
Construction du moteur et matériaux
Costruzione motore e materiali

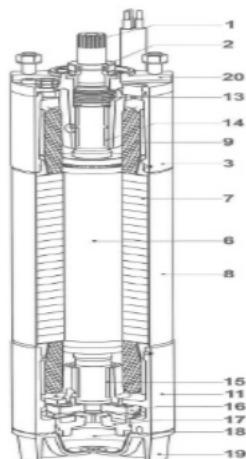


FIGURE 2.20: Caprari E6-8: MACX6-2B PUMP [18]

TABLE 2.3: Parts of Caprari E6-8: MACX6-2A, MACX6-2B PUMP

Position	E6-8 MACX6-2B	Materials	E6-8 MACX6-2A	Materials
1	Shaft	Stainless Steel	Shaft	Stainless
2	Sand Guard	Rubber	Sand Guard	Ruubber
3	Connecting Flange	Stainless Steel		
6	Rotor	Electrical Steel	Rotor	Electrical Steel
7	Stator	Electrical Steel	Stator	Electrical Steel
8	Stator Shell	Stainless Steel	Stator Shell	Stainless Steel
9	Winding	HT Wire	Winding	HT Wire
11	Lower Bracket	Stainless Steel	Lower Bracket	Stainless Steel
13	Mechanical Seal	Silicon Carbide	Mechanical Seal	Silicon Carbide
14(15)	Shaft Sleeve	HT Composite	Shaft Sleeve	HT Composite
16	Thrust Bearing	Brass	Thrust Bearing	Brass
17	Bearing Foot Slip	Cast Iron	Bearing Foot Slip	Cast Iron
18	Diaphragm	Rubber	Diaphragm	Rubber
19	Diaphragm Cover	Stainless Steel	Diaphragm Cover	Technopolymer
20	Upper Bracket	Stainless Steel	Upper Bracket	Stainless Steel

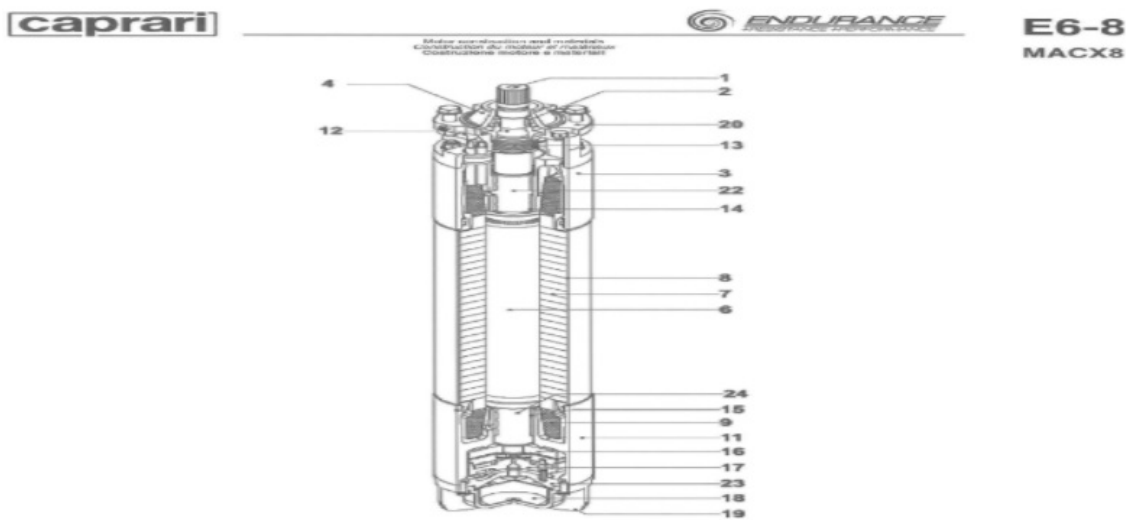


FIGURE 2.21: E6-8 MACX8 PUMP [18]

HVU18-50 and HV65-80 are two different PUMP used in the EVA System, as shown in Fig. 2.22. In Fig. 2.22, 1 is Electric Motor. 2 is Rigid Coupling. 3 is Coupling Guard. 4 is Lantern Bracket. 5 is Shaft Coupling. 6 is Discharge Bowl. 7 is Stuffing Box. 8 is Packing. 9 is Impeller. 10 is Pump Shaft. 11 is the Intermediate Bowl. 12 is Stage Casing. 13 is Wear Ring. 14 is Bearing. 15 is Shaft Bush. 16 is Suction Bowl.

Voltage Variations is $\pm 10\%$, and Maximum Altimetric Level is 1000. Maximum Ambient Temperature is 40, and Maximum Relative Humidity is 78.

TABLE 2.4: Specification of E6-8 MACX8 PUMP

Position	Parts of E6-8 MACX8	Materials
1	Shaft	Stainless Steel
2	Sand Guard	Rubber EPDM
3	Upper Bracket	Stainless Steel
4	Protection	Rubber
6	Rotor	Electrical Steel
7	Stator	Electrical Steel
8	Stator Shell	Stainless Steel
9	Winding	PVC insulated copper
11	Lower Bracket	Stainless
12	Mechanical Seal Cover	Stainless Steel
13	Mechanical Seal	Silicon Carbide
14(15)	Bearing Bush	Bronze
16	Thrust Bearing	Stainless Steel
17	Thrust bearing foot slip	Cast Iron
18	Diaphragm	Rubber
19	Diaphragm Cover	Stainless Steel
20	Connecting Fiange	Stainless Steel
22(24)	Shaft Sleeve	Chrome Plated Steel
23	Motor Bracket	Stainless Steel

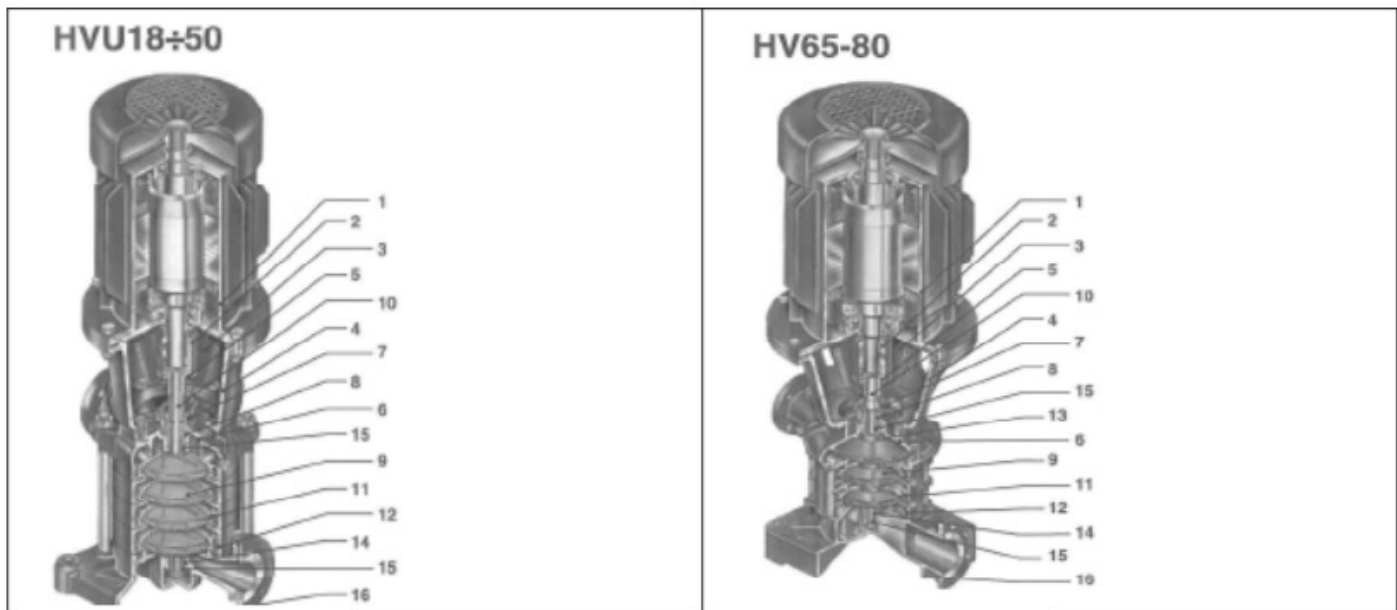


FIGURE 2.22: HV PUMP [18]

TABLE 2.5: Technical Data Electric Motors IP 55 [18]

Motor Power kW	7.5	15	22	37	55	90
Max number of starts equally distributed	15	12	10	6	5	4

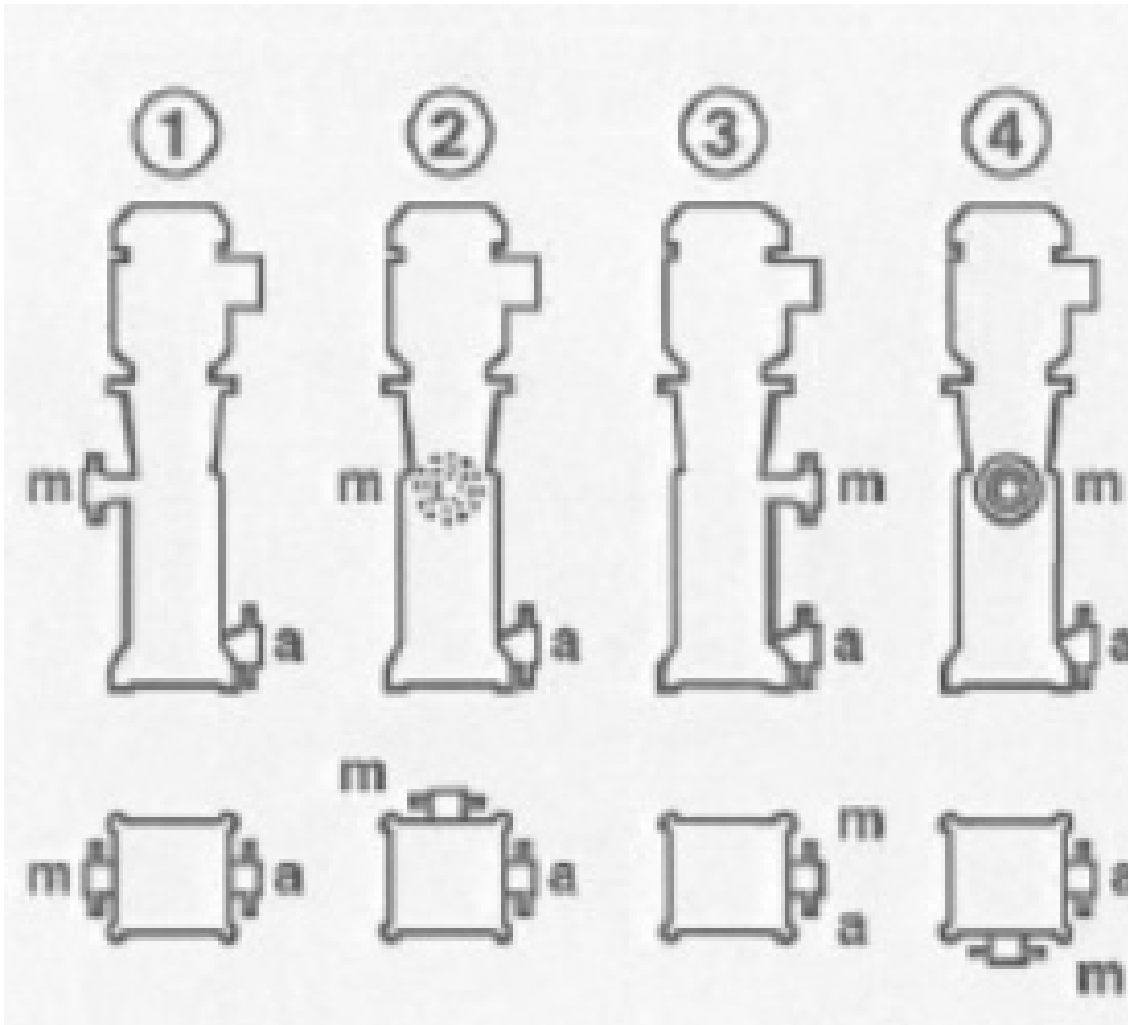


FIGURE 2.23: Inlet and Outlet Directions: 1 is Standard and 2-4 are On Request [18]

2.6 VegaBAR 14

In our system, pressure transducers installed on tank inlet pipe (with which we monitor incoming pressure). These pressure transducers mounted in downstream due to the pressurisation system (to maintain the correct pressure towards the Users connected to it). The sensor element in VegaBAR is the CERTEC measuring cell with the rugged ceramic diaphragm. VEGABAR 14 is a pressure transmitter, as shown in Fig. 2.24.

Operating voltage of VegaBAR is in range of 8-30 V DC. In principle, the VegaBAR used in this studied system has measurement ranges ranging from 1bar to 40bar depending on the height of tanks at altitude.



FIGURE 2.24: Electronics - VEGABAR 14 [18]

2.7 VegaWell 52

VegaWell 52 is hydrostatic absolute pressure transmitter as shown in Fig. 2.25. VEGAWELL 52 is using ceramic based measuring cell. The ceramic based measuring cell is a perfect choice for level measurement of water in industry.

Technical data (Measuring-range):

- Distance: 600 m
- Pressure: 0-60 bar
- Output: 4-20mA
4-20mA/HART-two-wire
- Ambient-temperature: -40-80 °C

2.8 Vegason 61

Vegason 61 is Ultrasonic pressure transducers, as shown in Fig. 2.26. These ultrasonic sensors or suspended pressure transducers installed inside the storage tank (for continuous level measurement).



FIGURE 2.25: VegaWell 52: Ceramic Measuring Cell-based Absolute Submersible Pressure Sensor [18]

The operation is based on the transmission of ultrasonic pulses to the surface of the fluid, which is reflected from the surface of the tank and recaptured by the acoustic converter. The time between the transmission and reception is proportional to the tank level. It has a display on board where the measurement is shown.

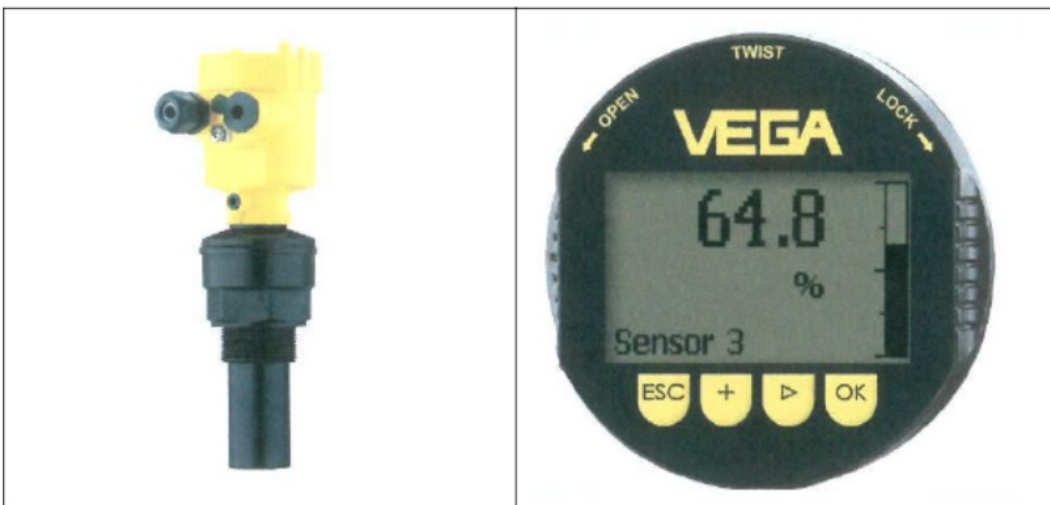


FIGURE 2.26: Vegason 61 Sensor for continuous level measurement and its Display Board [18]

2.9 Honeywell Hawk CPU

The HAWK CPU is utilising the NiagaraAX Framework especially for Heating, Ventilation, Air Conditioning (HVAC). Based on the collected data, the controller (PLC) can determine which devices enable to optimize functionality and consumption. Plants are fully managed by PLC, such as Honeywell Hawk PLC and supervision is Entrusted to Coach-Arena-AX type systems on EasyLink and EnergyLink.

2.10 BELIMO Rotary Actuator GR24A-SR-5

To maintain the desired water flow, they are using the Belimo Rotary Actuator, as shown in Fig. 2.27 to match the rotation of rotary valves and butterfly valves. Whereas, a butterfly valve consists of a pair of semicircular plates, attached with transverse spindle and installed inside a pipe to manage the flow.

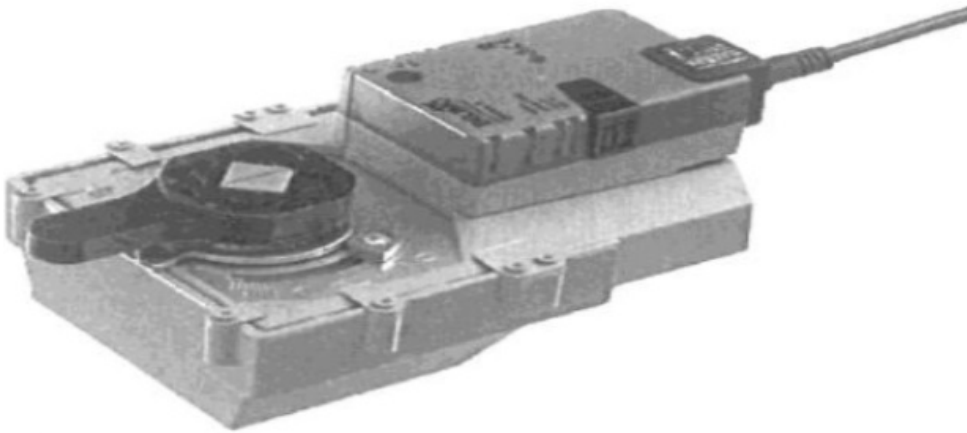


FIGURE 2.27: BELIMO Rotary Actuator GR24A-SR-5 [18]

Table 2.6 is a specification of rotary actuators shown in Fig. 2.27 used to match the rotation of valves.

TABLE 2.6: Specification of BELIMO Rotary Actuator GR24A-SR-5 [18]

Electrical Data	Nominal Voltage	AC/DC 24V
	Frequency	50/50 Hz
	Voltage Range	19.2..28.8V
	Dynamic Power Consumption	4.5W
	Static Power Consumption	1.5W
	Connection Supply/Control	Cable 1 m, 4 x 0.75 mm ²
	Parallel operation	Yes
	Position Indicators	Mechanically, Integrated two-section
Functional Data	Torque Motor	Min 40 Nm
	Position Accuracy	+5%
	Manual Override	Gear Disengagement with push-button
	Running time motor	150 s/90
	Sound Power Level Motor max	45dB(A)
Safety	Weight	2.5 kg

2.11 Electromagnetic Measurement: OPTIFLUX 2000 Sensor and IFC050/100 Converter

Optiflux 2000 Sensor is installed on tank inlet and outlet pipes to monitor flows. The operating principle of the electromagnetic gauges used in this sensor is based on Faraday's law. Fluid with a certain conductivity pass through a magnetic field (generated within the Sensor from two coils) of this sensor and this sensor generates a voltage proportional to the velocity of the fluid that is intercepted from the electrodes the sensor itself.



FIGURE 2.28: OPTIFLUX 2000 Sensor [18]

The signal converter is shown in Fig. 2.30-Fig. 2.31 connected to the OPTIFLUX 2000 sensor, as shown in Fig. 2.28- 2.29 amplifies, filters and converts the generated voltage signal inside the sensor itself to transmit a pulse and current output signal. Then these outputs, combined with the Modbus communication available on the converter board, are used by the supervisor of the water management system to manage data and process them to define operating logic and to save data field.

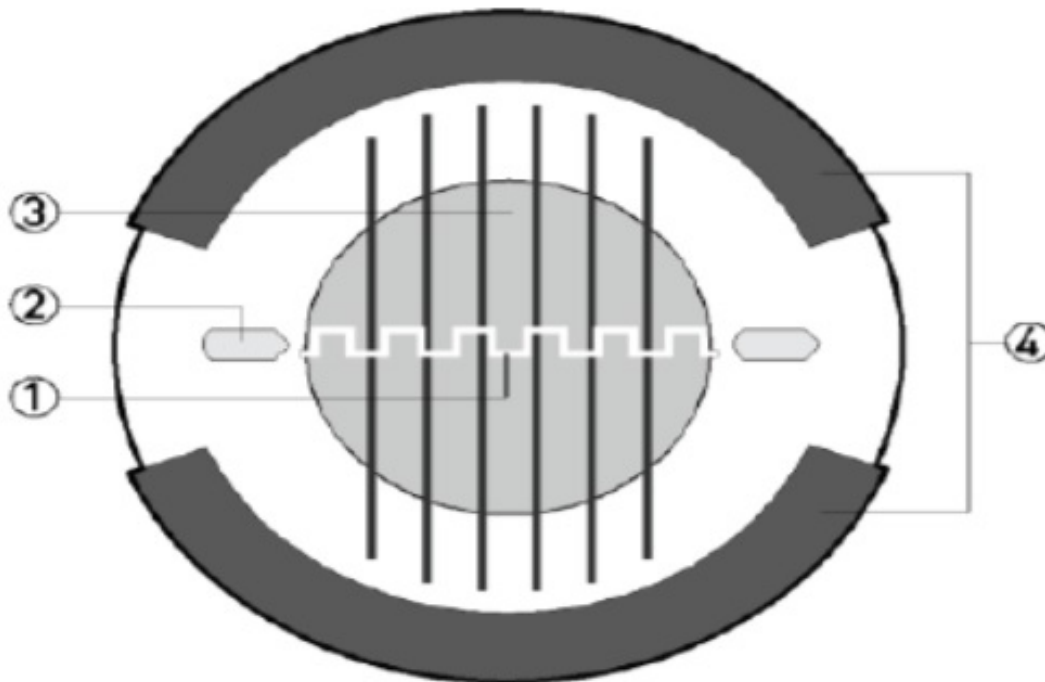


FIGURE 2.29: Internal structure of OPTIFLUX 2000 Sensor: 1: Induced voltage, 2: Electrodes, 3: Magnetic field, 4: Coils [18]



FIGURE 2.30: IFC050 Converter [18]

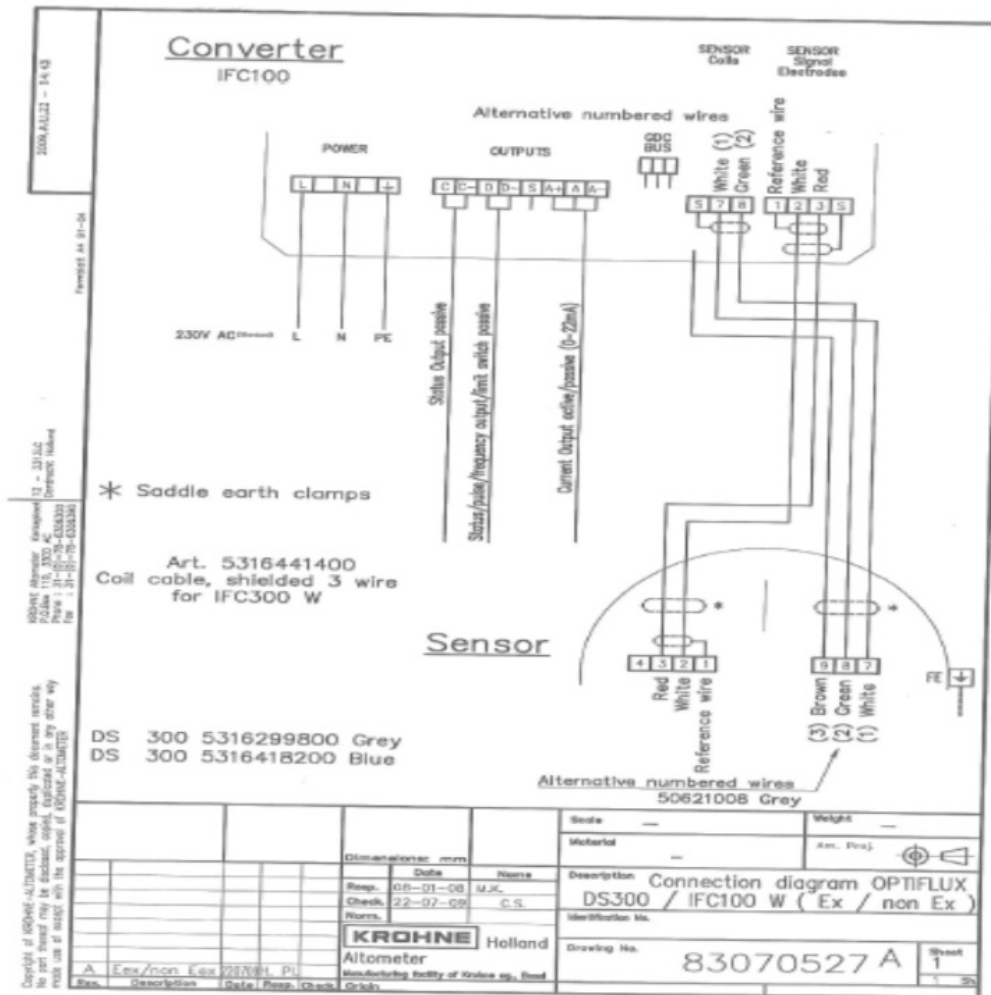


FIGURE 2.31: IFC 100 Krohne Signal converter for Electromagnetic flow meters [18]

2.12 ProMinent, DULCOMETER® diaLog DACa

The DULCOMETER® diaLog DACa is a measuring and regulation device provided by ProMinent, as shown in Fig. 2.32. It is a compact water analysis device that allows us to observe the quality of the water supplied. DaCa allows interfacing with the remote control system which, by reading the free chlorine values and pH coming from the analysis tool, records the data and provides the control of the dosing pump in the field by stabilising as much as possible the process of Final disinfection. Both DULCOMETER DACa measurement and adjustment channels are configurable according to the customer's specific needs.

- Measured variables: Bromine (Br), Chlorine (Cl₂), Chlorine Dioxide (ClO₂), Chlorite Ion (ClO₂⁻), Conductivity, Dissolved Oxygen, Fluoride, Hydrogen Peroxide, ORP, Ozone (O₃), Peracetic acid, and pH.



FIGURE 2.32: ProMinent, DULCOMETER® diaLog DACa. [18]

2.12.1 Amperometric variables, measuring ranges depending on sensors

- Bromine (Br)
- Chlorine (Cl₂)
- Chlorine Dioxide (ClO₂)
- Chlorite Ion (ClO₂⁻)
- Dissolved Oxygen
- Hydrogen Peroxide
- Ozone (O₃)
- Peracetic acid

2.12.2 Potentiometer measured variables, measuring ranges corresponding to transmitters

- Conductivity
- pH: 0.00-14.00
- ORP voltage: -1500 to +1500 mV
- Temperature: 0-150 °C

2.13 Prominent GammaX

The magnetic dosing pump used for disinfection regulation is the latest generation, the GammaX by ProMinent. A magnet moves the axis of the pump back and forth by turning it on and off. This movement is transmitted to the dosing membrane in the dosing head. Two check valves prevent the reflux of the dosing liquid during pumping. The performance of dosage of the membrane in pumps can be precisely adjusted via the stroke length and the running frequency. Also, using an analog signal 4-20mA output from the server, the pump increases or decreases the dosage to keep the chlorine constant (0.20ppm) in the final disinfection process.

2.14 Communication Standards

To ensure a high degree of compatibility with communication standards such as Ethernet, TCP / IP, LONWORK, KNX, MODBUS, and BACnet, this EVA system work on the principles of OPC (Open Control System). Most of the remote control sub-systems of EVA system is equipped with 4G routers with IP address. Supervisor server is specially configured to manage these remote control sub-systems of the EVA system. Whereas, OPC allows Windows programs to share information with industrial equipment, as shown in Fig. 2.33.

OPC is always realized in server/client pairs. The OPC server is a software that translates PLC based industrial communication protocol into the OPC protocol. [32].

2.15 Conclusion

EVA system has Dulcometer (a single device with inbuilt 7 ION Sensors and 4 Physical sensors). Along with Dulcometer, EVA system also has Pressure Sensor (VEGABAR, VEGASON, and VEGAWELL) and water flow sensor (OPTIFLUX 2000). Niagara is used to control the network of these sensors in

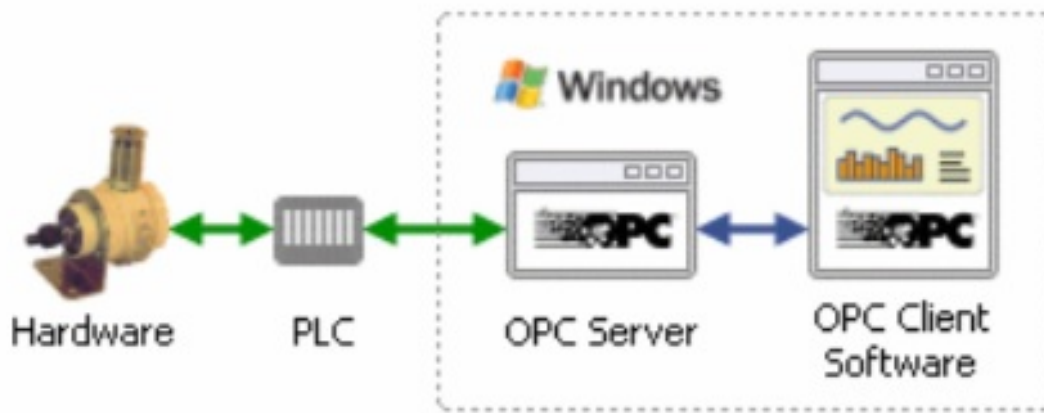


FIGURE 2.33: OPC: Open Protocol Communication [32]

addition to managing frequency converter, soft starter, rotary actuators, PUMP, a signal converter and so on. Our Existing EVA system is capable of measuring pressure, flow rate, water level, pH, conductivity, ORP, and temperature of the water. Following actuators are also in use in EVA system: Danfoss Frequency Converter, Danfoss Softstarter, Caprari Pump, Belimo Rotary Actuator (GR24A-SR-5), IFC050/100 Converter. Soft starter and frequency converter are used with PUMP. To nullify the effect of both chemical and biological contamination in water, this system also administers the mixing of implants in water with the help of chemical dosing device called GammaX. This EVA system is using eight different E6-8 series PUMPs.

*water is an upholder of all lives and the saviour of
everything living or dead on earth.*

Rig Veda, 3000 BC

3

Reviews of Communication Protocols, Sensors and Actuators in Water Management Cyber Physical System

3.1 Communication Protocols

3.1.1 Ethernet

Controllers in Water distribution system having a proven capacity to manage both Ethernet and TCP/IP stack [63]. Information is gathered from local sensors by local PLCs [36]. And then Ethernet/IP protocol is used to send collected sensor information to a master PLC. Local sensors were aggregated into racks with easily expandable and replaceable I/O modules. Each of these racks communicates to the PAC via Ethernet using Ring Topology [36].

This chapter contains material from the previously published works [17], and [16] coauthored with Giuseppe Airo Farulla, Paolo Prinetto, Ludovico Iovino, Marco Indaco, copyrighted by Springer and Journal of Advance Research in Dynamical & Control Systems

3.1.2 WiFi

Wi-Fi or Wireless Fidelity is a network standard for wireless networking with IEEE 802.11 devices. 802.11 has “radio frequency” used for transmission in Wi-Fi. In order to connect to the Internet, Wi-Fi compatible devices are using either the WLAN network or wireless access point (hotspot). The system in [52] integrates Wi-Fi network with the stationary

sensor in order to get the reading of sensors. In comparison with Zigbee, Wi-Fi is a communication medium that delivers high data rate communication [62]. Frequency allocation in 802.11s wireless mesh networks along with multiple terminals and antennas are able to manage critical components like tanks, reservoirs, and pumps [55]. IEEE 802.11ah is used to control indoor meter devices and gateways to other IEEE 802.11ah applications seated on poles. Therefore, water meters can be read remotely and eliminate the necessity of manual monthly reading [13].

3.1.3 Modbus

Modbus RTU mode of transmission is available in Raspberry Pi [17]. Modbus is created as an external node of the Eclipse SCADA Infrastructure Model (ESIM) on Eclipse SCADA simulator [17]. To connect Arduino PLC to Modbus communication protocol enabled devices; some hardware modules are connected to the Arduino to allow it to as a Modbus master node [70]. In our work, we are using both Arduino and Raspberry Pi. Interconnection of equipment from different manufacturers using the Modbus protocol is possible with a gateway between Modbus-TCP and Profibus-DP [2]. Communication between master PLC and slave Arduino based on Modbus Protocol is investigated and also methods to organise data packet in Modbus is also discussed in [40]. In [80], a low-cost home automation system based on Arduino microcontroller has been introduced that works with Modbus protocol. Python based integration of ZigBee protocol in single board computer Raspberry Pi via Modbus help in automation of electric meter reading and monitoring system [1]. Lightweight version of the MODBUS protocol is used for wireless sensor networks [50]. Modbus TCP is a highly used industrial networking protocol that has an excellent performances [9]. The different method for authenticating sensors connected using Modbus TCP has developed in [24]. We are taking benefit of these in our WM-CPS. The paper [42], present the measures to embed the Modbus protocol into the ZigBee stack.

3.2 Sensors

3.2.1 Flow Sensors

In water test networks [21], Pressure sensors and flow sensors help us in the monitoring of flows and pressures in any water distribution network.

To detect leakage in the water management system, we put two flow sensors on two ends of the pipe. After that, we compare reading of both sensors. And if the reading of both sensors is equal, then there is no leakage in pipe. Otherwise, there is leakage as shown in Fig.3.1. If a hacker, open a nozzle, orifice or emitter to wastewater in between path then reading of two flow sensor will differ and help us to detect there is an attack on water management CPS [16].

3.2.2 Temperature Sensors

Drinking of cold-water ($3^{\circ}C$) and water of room-temperature ($22^{\circ}C$) led to a decrease in heart rate ($P < 0.01$) and an increase in the heart attack ($P < 0.05$). There is no decline in heart rate and increase the probability of heart attack with the drinking of body-tempered water ($37^{\circ}C$) [34]. Temperature also plays a vital role in the separation of inorganic anions, acids, and base in samples of water [41]. The Pt-1000 is a resistive sensor with one k resistor, which helps us to detect the temperature of water in the range of $0^{\circ}C$ - $100^{\circ}C$ [47]. Depending on reading of Pt-100 sensor, we can decide this water is drinkable or not because we can't drink if temperature is greater than $57.8^{\circ}C$ as even beverages such as coffee, soup, and tea are served around $71.1^{\circ}C$, otherwise it can lead to burning of tongue and vital organ of digestion system [27]. The general water distribution system cannot supply water of more than $45^{\circ}C$ temperature as shown in 3.2. Human body temperature is $37^{\circ}C$. A human can't be able to drink it or bath in it. It is an assumption that supply

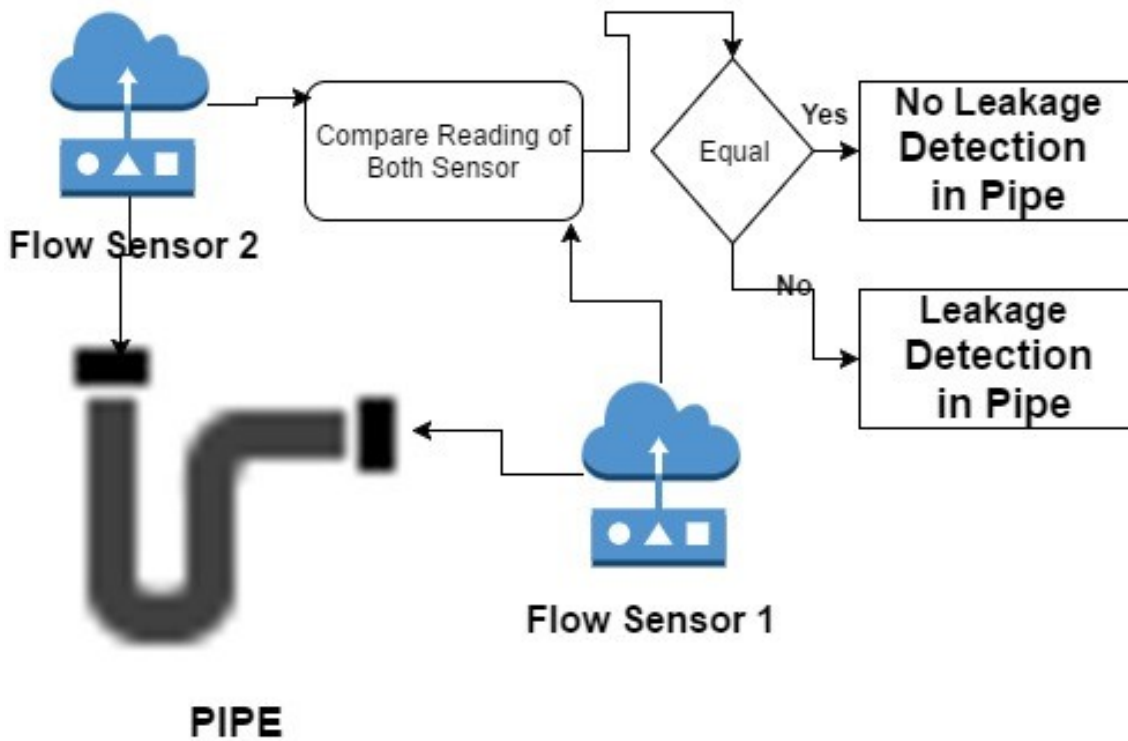


FIGURE 3.1: Flow Sensor in WMCPS

with $45^{\circ}C$ will lose its temperature during distribution network and when it reaches to the human user, it will perfectly match the human body temperature. Whereas, some hot beverage requires a temperature of $57.8^{\circ}C$, for that user need to use a single heater or boiler. During the attack on the water management cyber-physical system, a hacker may able to set that temperature limit beyond $45^{\circ}C$. In that scenario, the user will face the problem of burn if they directly drink it or directly bath in it. If the temperature sensor detects the temperature of drinking water is greater than $45^{\circ}C$, then it should initiate an alarm to stop water boiler or heater. If the end user gets water of more than $45^{\circ}C$ temperature, then we may assume that our system is compromised.

3.2.3 Level Sensors

Ultrasonic Sensor senses the level of water if installed on top of the water tank, as shown in 3.3. We shall subtract the distance of water from the height of water tank and find the level of water in the tank. This sensor consists of one or more ultrasonic transmitters (basically speakers), a receiver, and a control circuit. The transmitters emit a high-frequency ultrasonic sound, which bounces off anything that has higher density than air. That ultrasonic noise is reflected and detected by the receiver on the sensor. That return signal is then processed by the control circuit to calculate the time difference between the signal being transmitted and received. This half of time difference can subsequently be multiplied with speed of sound, i.e. 340 m/s, to calculate the distance in m between the sensor and the reflecting object.

3.3 Actuators

During cyber-attack, a hacker may try to damage system infrastructure and its actuators. Access to actuators during cyber-attack, give a leading edge to hackers and eventually result in damage to system infrastructure. Therefore, it is essential to think about the cybersecurity of actuators. Following actuators are in use in water management CPS: Pump,

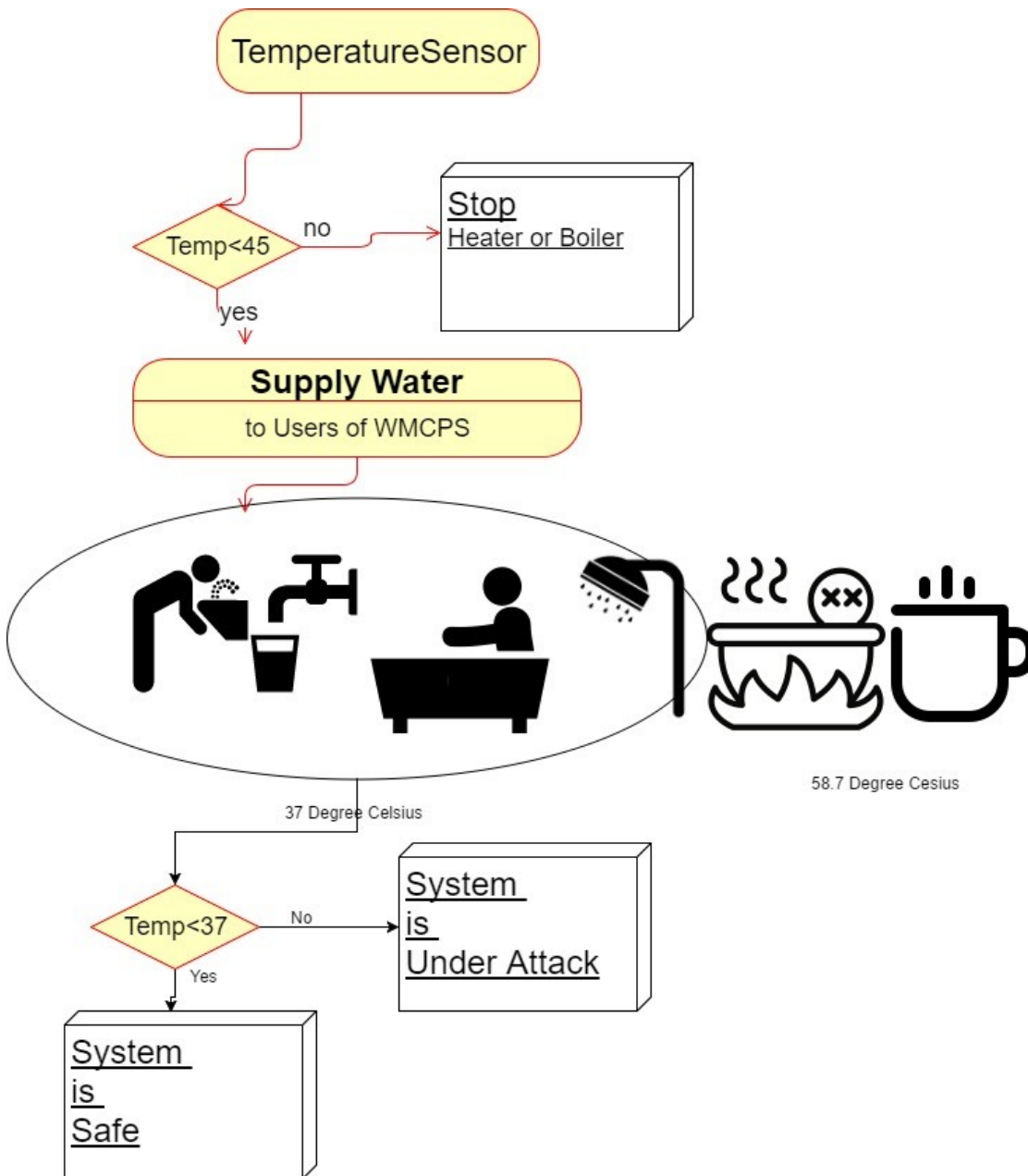


FIGURE 3.2: Temperature Sensor and Cyber Security Attack on Water Management System [16]

Gate, Emitter, Compressor, Weir, Valve, Siphon, Condenser, Flow splitter, Chemical dosing device, Water turbine, Heat pumps, Boiler, Heat Exchanger, Rack and Pinion, Hydraulic brake, Lift cylinder, and Dampers [16].

3.3.1 Pump

Pumps provide flow. Water lifting devices are in service of humanity since 3000 BC [10]. In an ancient period of history, water wheels and chutes have developed then and used with the muscle power of animals to move the wheels [57]. During the ages, different variety of pumps were invented based on need and demand of that time and those pump are in today use. Time, the name of invention and inventor of 27 pumps are shown in Table 3.1. [87].

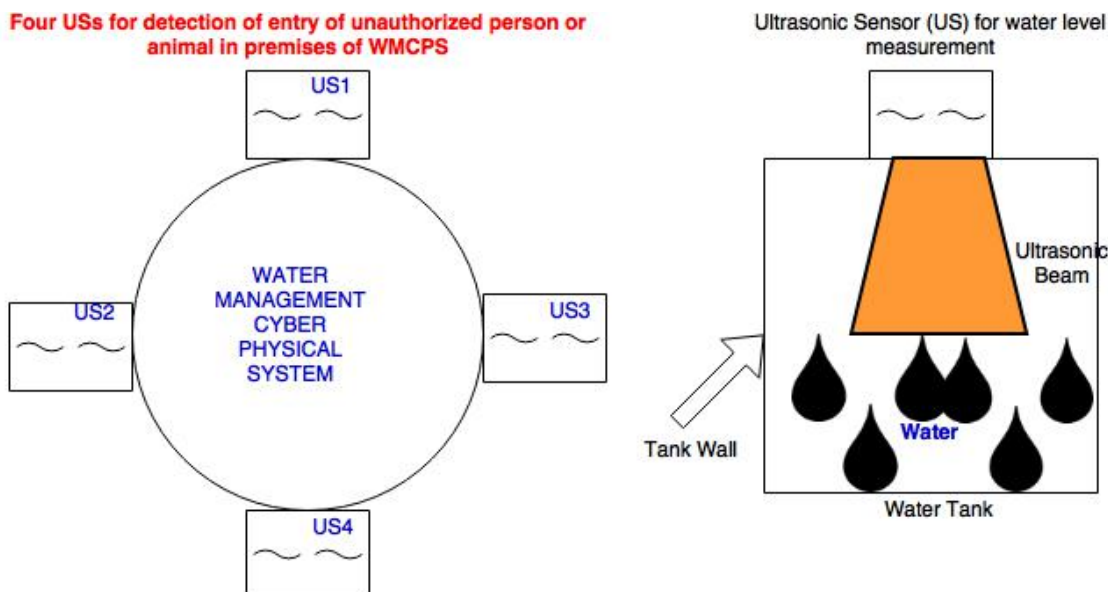


FIGURE 3.3: Ultrasonic Sensor measure level of Water in Tank

TABLE 3.1: Evolution of Water Pump [[67]

Time	Invention	Inventor
287–212 BC	Water-screw	Archimedes
285–222 BC	Force pump	Ktesivius
260–180 BC	Chain Pump	Philon Byzantium
200 BC	Waterwheel with pots	
1200 AD	Windmill	Chinese
1698 AD	Vaccum Based Pump	Thomas Avery
1580 AD	Gear Pump	Serviere
1580 AD	Sliding Vane Pump	Ramelli
1650 AD	Piston vacuum pump	Otto van Guericke
1674 AD	Packed plunger pump	Sir Samuel Morland
1687 AD	Centrifugal pump	Denis Papin
1830 AD	Screw pump	Revillion
1840 AD	Steam pump	Henry R. Worthington
1851 AD	First centrifugal pump	patent John Gwynne
1857 AD	Direct-acting steam pumps for boiler feed.	Henry R. Worthington
1860 AD	Reciprocating stream pump	A.S. Cameron
1874 AD	Vane pump	Charles Barnes
1897 AD	Deep suitable turbine pump	Preston K. Wood
1899 AD	Rotary vane pump	Robert Blackmer
1901 AD	Deep well vertical turbine pump	Byron Jackson
1905 AD	Multistage centrifugal pumps	—
1908 AD	Electric motor-pump	Hayward Tyler
1916 AD	Motor-driven reciprocating pump	Aldrich
1923 AD	Seal-less vertical pump	Ruthman
1927 AD	Cylinder reciprocating pump	Aldrich
1940 AD	Axial-flow and jet pumps	—
1956 AD	Submersible sewage pump	Flygt

Table 16 represents a chronology of the evolution of water pump from Archimedes water screw in 287-212 BC to Submersible sewage pump of 1956 AD. The Archimedes screw is a mechanical instrument. It is used to move water upwards. The timing of development of Archimedes screw is still an open question. Archaeological evidence during the reign (705–681 BC) of Sennacherib (king of Assyria) and ancient literature recommend that the water screw was in use in the civilisation of Mesopotamia before the era of Archimedes [78]. Cylinder based force pump use a vacuum to draw water from valves. Motor-driven centrifugal pumps create suction to pull the water. Axial-flow pumps after the invention in 1940 applied as compressors of jet pumps are also mentioned in Table 16. Jet pumps are suitable for wells of more than 200 feet in depth. The journey which starts from Archimedes water screw has reached to Electromagnetic pumps. Electromagnetic pumps can move conductive fluids and handle extreme temperatures. That's why it uses in nuclear reactors. Whereas HVAC water-distribution systems use rotating and stationary component based centrifugal pumps. Impeller and shaft are part of rotating division. Casing, the housing cover, and bearings are part of the stationary section as shown in Fig.3.4.

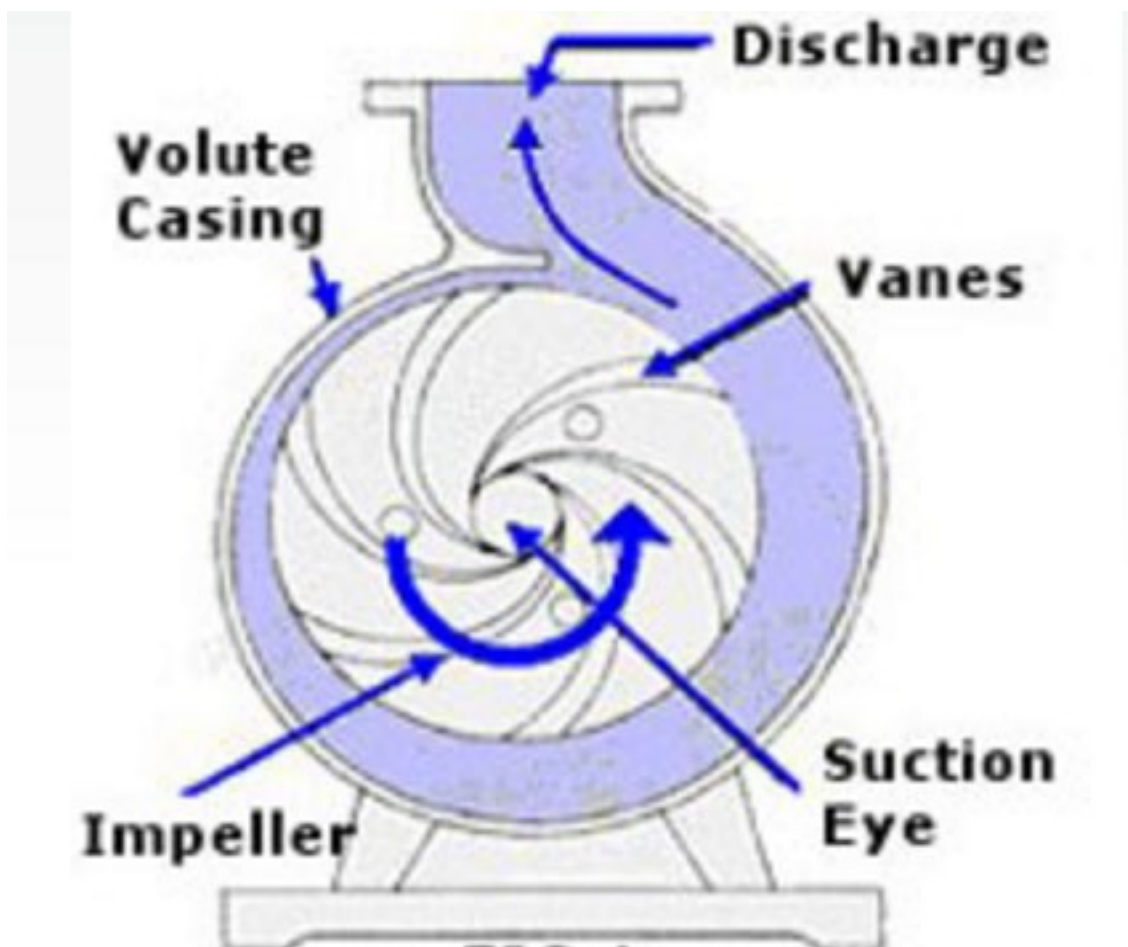


FIGURE 3.4: Design of Centrifugal pump [38]

Fluid flows into the suction eye. The suction eye is the centre of the rotating impeller. With the rotation of the impeller in an anti-clockwise direction, impeller creates centrifugal acceleration with thrusts of the fluid outward radially. The energy produced by this type of acceleration is directly proportional to the speed of the impeller. The faster the impeller turns, the quicker the fluid movement and eventually, its force become stronger [38]. Pump power requirement is related to three parameters: flow in the pump, head of the pump, and wire-to-water efficiency. Hydraulic simulator (i.e., EPANet) is used to calculate these parameters for any schedule of the pump [30]. Along with hydraulic and water quality simulation, EPANet also models and simulates dynamic reaction between chemical and biological species. In

EPANet, pump supplies a constant amount of energy (horsepower or kilowatts) to the fluid. We have to define a Pump curve in EPANet. The pump curve is a combination of flow and head. The pump never operates outside the range of its pump curve. In modern pumps, motors are used for pumping the water.

High and fine literature is wine, and mine is only water; but everybody likes water.

Mark Twain

4

A systematic review and Design of Different Attacks Strategies for Emulator of Water Management Cyber Physical System (WM-CPS)

4.1 Introduction

Cybersecurity is a multidisciplinary branch of science that deals with computer science, philosophy, electronics, sociology (psychology of hacker), politics (patronage behind hacking) and other subjects. Threats to Cybersecurity is initiated by malware, hacker, and Insider and we call this as an act of Cyberterrorism. Tools for war dialling, scanning, traffic sniffing and password cracking, are available to assist in these type of activities. Diane VanDe Hei is an executive director of the Association of Metropolitan Water Agencies and spokesman of Information Sharing and Analysis Center (ISAC) for the water utilities. He said, "If we had so many dollars to spend on a water system, most of it would go to physical security," That shows the lack of concern for the cybersecurity of water management system (WMS) [31]. WMS uses Ions sensor to detect chemical contamination in water and ensure proper dosage of chlorine, and other mineral nutrients. WMS used the physical sensor to measure temperature, pressure, turbidity, pH and Oxidation-Reduction Potential (ORP). Threats to the infrastructure of the water management system are on not only sensors but also actuators like pump, valve, chemical dosing devices, gate, an emitter, and so on. Attacks are also possible in WMS in following ways: barge into operations, make unauthorised changes in program/instruction, an obstacle to the regular progress of data, transmit incorrect information, manipulate alarm thresholds, and block access to account information. To protect the water system (one of the critical infrastructure) needs the association of a risk assessment, enlisting of vulnerabilities and mitigation of hazardous [31]. Hackers have almost become demonic. There is the possibility of the presence of roguish persons of wicked mind as Insider. There is no useful mode of remedying the hacker, purification of the spirit of an intruder, but we can develop a countermeasure to avoid these types of cyber-attacks on water management CPS. The countermeasure is a

panacea evolved out of the experience of all cybersecurity experts, heightening cybersecurity and conducive to end user. Every time cyber security expert take a countermeasure to secure their system after listing all possible way of an attack on that time. But the destructive forces are at work and the process of insertion, modification and abridgement going on and cybersecurity expert again discovered that his countermeasure is fell short of a fact. Countermeasures end in dissolutions which in turn give place for the re-creation of countermeasures. The crusade against cyber terrorism will go on until hacker stops. According to Charlie Chaplin, nothing is permanent in this wicked world, not even our troubles. Similarly, neither threats nor its countermeasure is continuous in this world. Both risks and its countermeasures have to update itself incrementally to survive. Either threats or countermeasures will gradually die out if that will not fit to survive. First cybersecurity attack on water system was in 1994. Sub-Section 4.2 analyses seven attacks on water supply before 1994. In Sub-Section 4.3 studies seven cybersecurity attacks on integrated sensor-actuators water management system after 1994. In Sub-Section 4.4, we have developed an emulated WMS consists of Sensors and Actuators. Information disclosure, forced browsing, SQL injection and denial of service are four different current attack strategies taken into consideration in our work. We perform these four attacks on the emulated system. We find four threats and also discuss its solution. Comparison of attack strategies used in past attacks and attack strategies applied to the emulated system is shown in Table 4.1. Sub-Section 4.5 presents recommended cybersecurity practices for the water sector. Sub-Section 6 describes the conclusion and its future scope.

TABLE 4.1: Attacks Strategies Used in Past Attacks and Attacks Strategies on Emulated System

Attacks on	Attack Strategies
Arizona's DAM	Information Disclosure
Maroochy Water Services	Denial of Service (DOS)
Harrisburg Water Plant	Information Disclosure, Forced Browsing and DOS
Tehama Colusa Canal Authority	Denial of Service (DOS)
South Houston Water Plant	Informed Disclosure
Bowman Dam	Forced Browsing
Kemuri Water Company	Information Disclosure, Forced Browsing

4.2 Past Attacks on Water Supply before 1994

The first cybersecurity attack on water supply took place in 1994. Before 1994, world history includes many instances of attacks on water resources. Although those were not cybersecurity attacks on the water system, they also affect human life in significant ways, as shown in Fig. 4.1. Nero used cherry laurel water, which contains cyanide, to poison the wells of ancient Rome during the first century CE [9]. In 1503, Leonardo da Vinci and Machiavelli planned to divert the Arno River away from Pisa during the conflict between Pisa and Florence [9]. In 1841, a reservoir in Ops Township, Upper Canada (now Ontario) was destroyed [29]. In 1863, General U.S. Grant cut water supply during the Civil War siege of Vicksburg [9]. In 1973, a biologist in Germany threatened to contaminate water with bacilli of anthrax and botulinum unless he was paid US\$8.5 million [29]. In 1983, Israel claimed to uncover a plot by Israeli Arabs to poison the water in Galilee, Israel with "an unidentified powder." In 1992, lethal concentrations of potassium cyanide were reported in the water tanks of a Turkish Air Force compound in Istanbul [29].

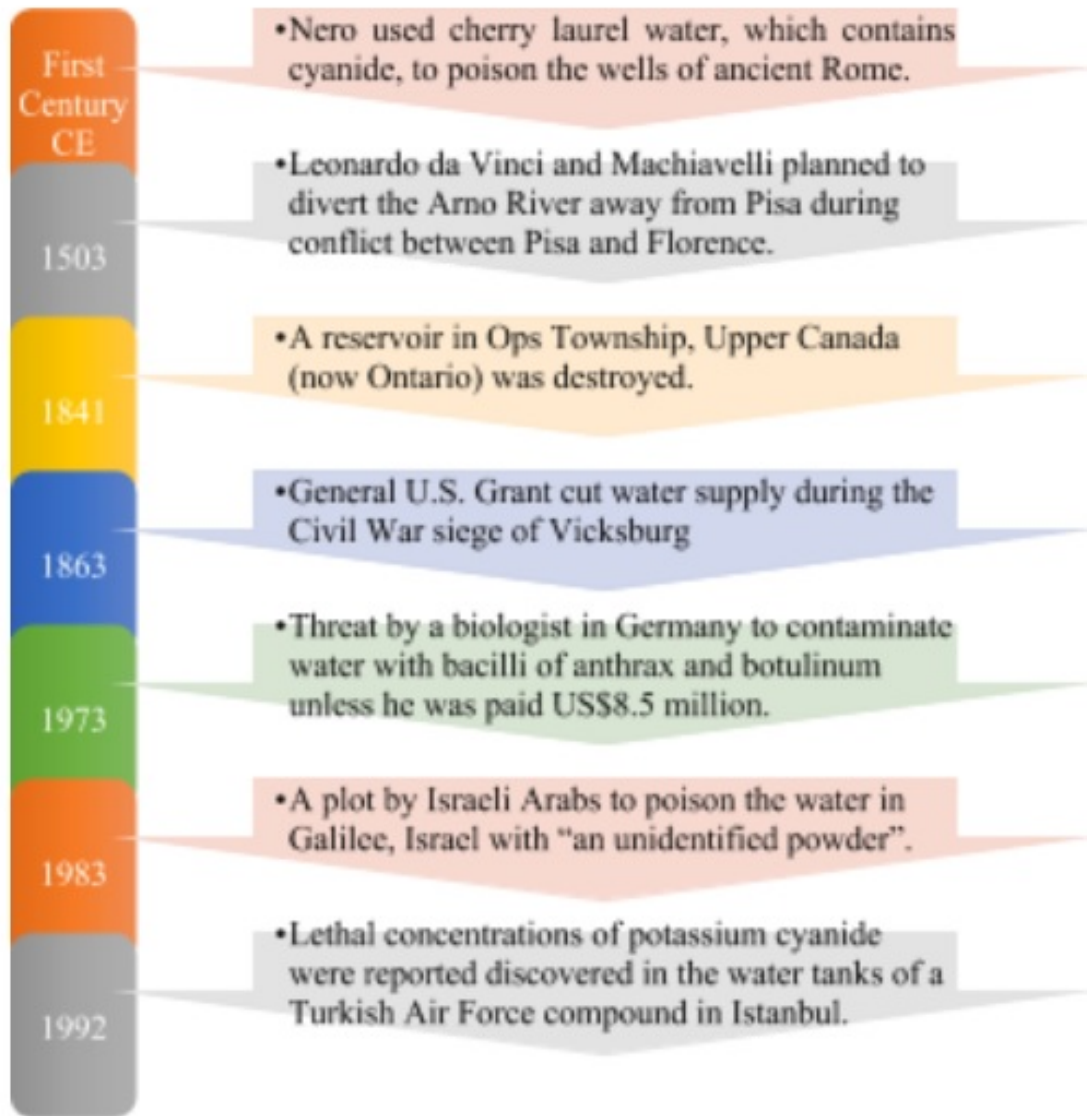


FIGURE 4.1: Water Terrorism before Cyber Security Attacks on Water System

4.3 Past Cyber Security Attacks between 1994 and 2016

Criminal hackers take advantage of both technical vulnerabilities and human failings to penetrate insecure systems [37]. There are seven reported incidents of cyber terrorism on water management CPS, as shown in Fig. 4.2. In 1994, the first incident was published in the USA. In 2000, the second incident was at Maroochy Water Service, Australia. In 2006, there were cybersecurity attacks on Harrisburg water treatment plant, USA. In 2007, USA again faced cybersecurity attacks on Tehama Colusa Canal Authority, USA. After that, there were no attacks on the water system for the next years. In 2011, Water Plant in South Houston, USA again faced cyber terrorism. The last reported incidence of the attack on the water system in the USA was in 2013 when hacking of Bowman Dam; the USA took place. There was a report of cyber terrorism in 2016, but the location is not known, and the report also use the name of the non-existing company called Kemuri Water Company (KWC). Fig. 4.2 organises incidents of cyber terrorism in chronological order. The real greatness of countermeasures lies in expounding the philosophical background of both cyber security and cyber terrorism.

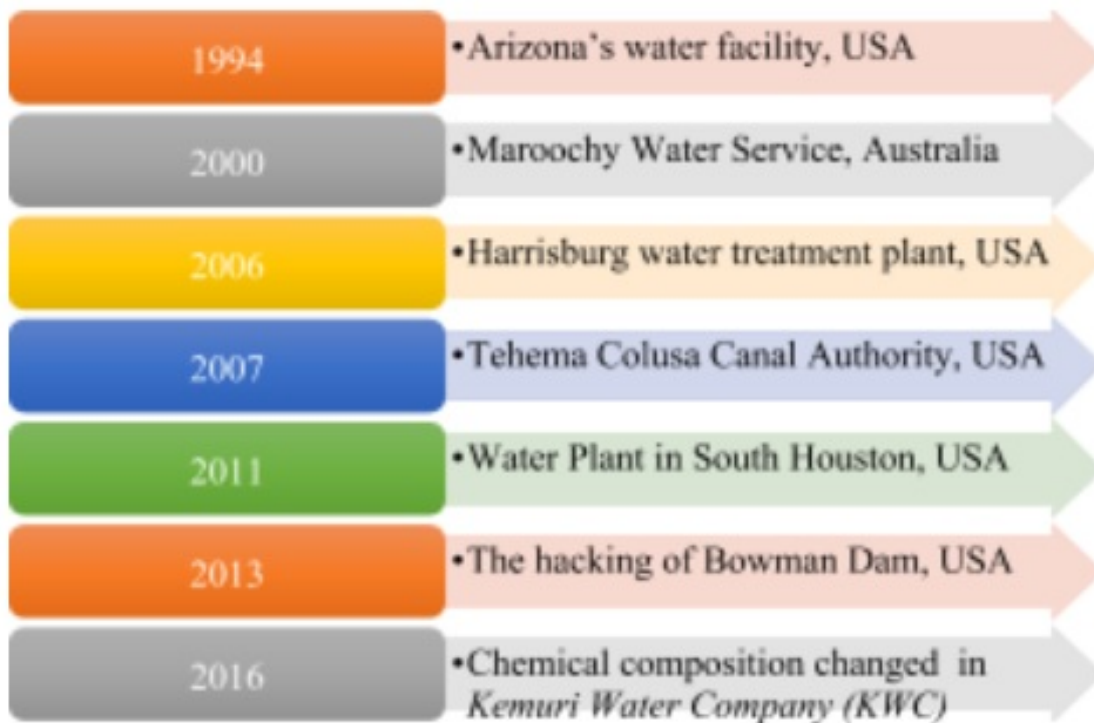


FIGURE 4.2: Reported Incident of Cyber Terrorism related to Water

4.3.1 Attacks on Arizona's Theodore Roosevelt Dam



FIGURE 4.3: Theodore Roosevelt Dam [77]

Theodore Roosevelt Dam was the world's largest "cyclopean-masonry" dam, as shown in Fig. 4.3. It was constructed between 1905 and 1911 [77]. Theodore Roosevelt Dam forms Roosevelt Lake. Length, Shoreline, Capacity, Surface

acreage when full, and Maximum depth of Lake are 22.4 miles, 128 miles, 1,653,043 acre-feet, 21,493 acres and 188 feet respectively [77]. In 1994, reports came about the vicious activities of a computer hacker. In that report, the hacker gets access to the SCADA computer system of Arizona’s Roosevelt Dam on the Salt River located northeast of Phoenix Arizona. It was one of the first cases of cyber-attack on water systems. Hacker never gains total control of irrigation water supply, floodgates and its 36 MW hydroelectric generating capacity. The researcher concludes that no lives or property were ever attacked. Otherwise, there was the possibility of flood in cities like Mesa, Tempe, and Phoenix, and Arizona [45].

4.3.2 Attacks on Maroochy Water Services (MWS)

The most famous water system hacked at MWS, Australia [75]. MWS sense difficulty in their wastewater system in March 2000. Wastewater pumping stations were unable to receive proper radio message. That’s why pumps were not functioning accurately. Alarms were not doing their works to alert staff to faults. It was initially thought these are a minor problem which any system face in the beginning. Finally, during monitoring of each signal in system, one engineer discovered that it is not a teething problem, but it is an attack on the system and someone intentionally making the problems. Then they track the Vitek Boden, who was behind this crime. Mr Vitek was also a former contractor in MWS. Finally, he was taken into custody and jailed. Mr Boden takes help of the laptop and portable radio transmitter to manipulate of 150 sewage-pumps, as shown in Fig. 4.4. In three months, he has discharged 106 litres of unprocessed sewage into local waterways. The prime motivation behind the attack was revenge because Mr Boden failed to get a job in MSC. This incident is illustrated as an example of the damage that may happen if we don’t think about the security of SCADA systems. The case was cited in a report made by U.S.A. president’s advisory of IT security [75].

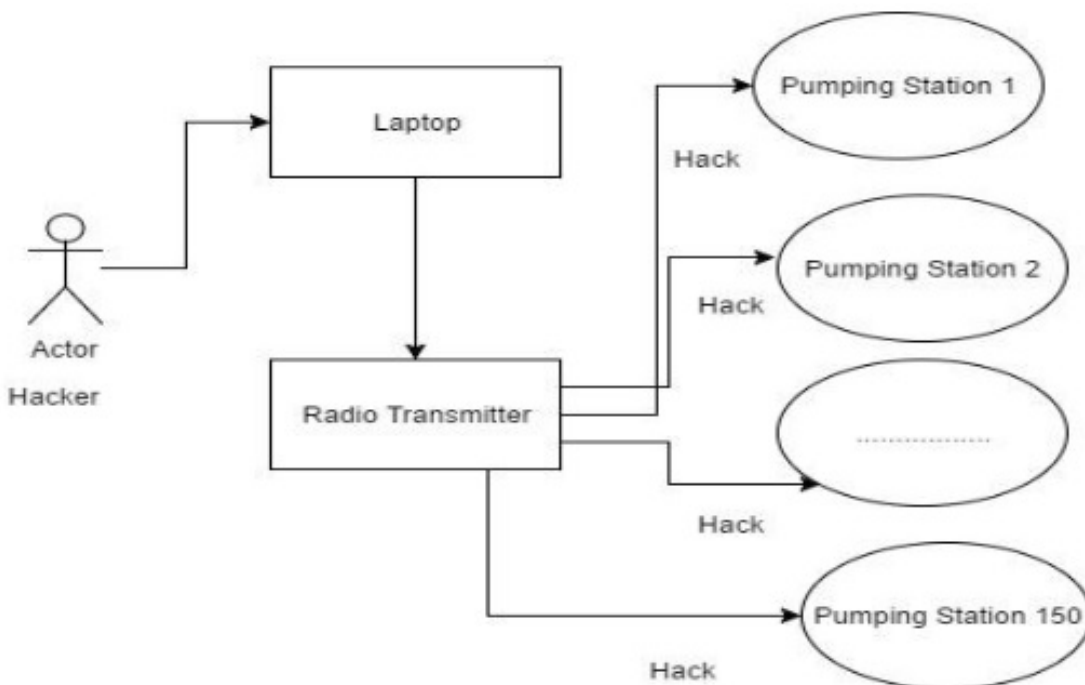


FIGURE 4.4: Attacks on Maroochy Water Services



FIGURE 4.5: Attacks on Harrisburg Water Treatment Plant

4.3.3 Attacks on Harrisburg water treatment plant

Harrisburg water filtering plant incident is one of the most famous cyber-attacks on water SCADA system [6]. According to Snyder, laptops used in water treatment systems because these treatment systems have to monitor multiple locations. In 2006, a hacker compromised remote access of employee's laptop and used compromised laptop as an entry point into treatment plant of water at Harrisburg and also install a computer virus and spyware on the plant's computer system [49]. After the incident of the intrusion, the plant changed all system passwords and also eliminated home access to the network [25]. The hacker also tried to covertly use the computer system as its distribution network for e-mails or pirated software, as shown in Fig. 4.5 [25].

4.3.4 Attacks on Tehama Colusa Canal Authority (TCCA)

Michael Keehn was the main perpetrator behind this attack. He was an old employee at TCCA. He deliberately enters and destructs computer system used to divert water from the Sacramento River for irrigation of local farms, as shown in Fig. 4.6. The crime he commits was the destruction of a secured computer with no authorisation to do it. He installed illegitimate software in SCADA system of TCAA. The United States Bureau of Reclamation owned this SCADA system. Keehn approached the system on August 15, 2007. He was arrested following a grand jury indictment on November 15, 2007. He got a sentence of 10 years imprisonment [64]. The intrusion cost the TCCA more than \$5,000 in damages [64].

4.3.5 Attacks on Water Plant in South Houston

A hacker broke into a water plant in South Houston in 2011. After the attack, he tweeted on 5 November that he has identified PLC configurations of a waste-water treatment plant. He gets SCADA data from an HMI (human-machine interface) box of a generator. But he claims to get files of the water metering control system from Spain or Portugal. IP addresses used by attackers belonged to Russia [51].

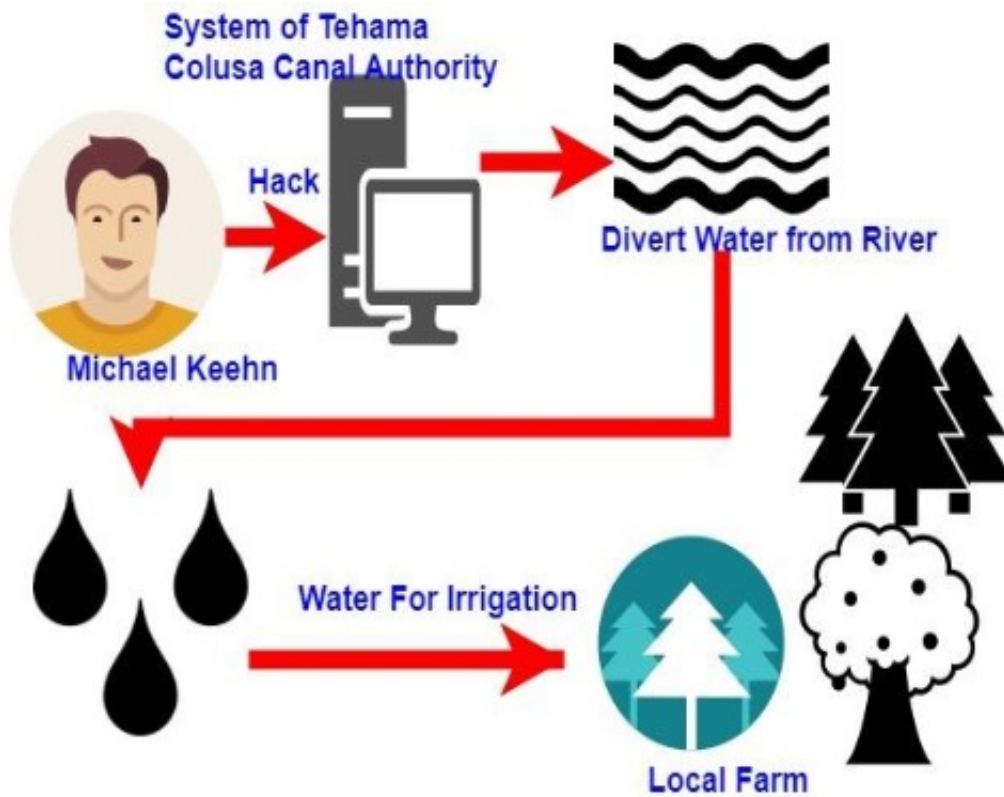


FIGURE 4.6: Attacks on Tehama Colusa Canal Authority

4.3.6 Attacks on Bowman Dam

In 2013 Hamid Firoozi, an Iranian computer hacker and employee of ITSecTeam, hacked into the control system of Bowman Dam, New York. Justice Department indictment in federal court on March 24, 2016, has detail description of this incident. ITSecTeam is an Iran based computer security company sponsored by Islamic Revolutionary Guard Corps and spy agencies of Iran. By accessing the SCADA system, he was capable of learning water levels, temperatures, and status of the sluice gate, which manage the flow of water, as shown in Fig. 4.7. Fortunately, he was not in a position to maintain the barrier from Iran because that control system was in maintenance [84].

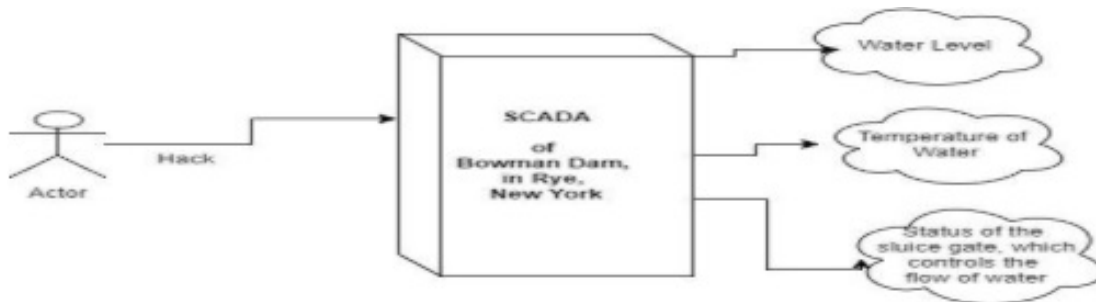


FIGURE 4.7: Three Information Accessible to Hackers of Bowman Dam

4.3.7 Attacks on Kemuri Water Company (KWC)

Syria based hackers enter into the computer of KWC by exploiting web vulnerabilities and manipulate the levels of chemicals to treat drinking water in 2016. Login information of KWC's AS/400-based control system was saved on the front-end web server. This control system managed programmable logic controllers (PLCs) to control valves and ducts for proper management of water flow and chemicals discharge used to treat water, as shown in Fig. 4.2 [46]. Attack was on Top Level, and then hackers were able to control each component either PLC or Valve or Duct.

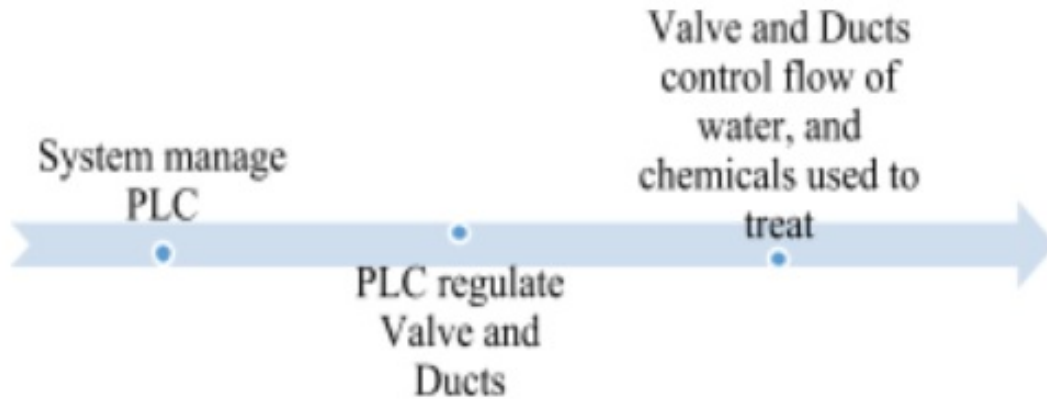


FIGURE 4.8: Working of Kemuri Water Company's (KWC's) Water System

4.4 Present Cyber Security Attacks on Emulated Water Management System

First, we develop an emulation of real Water Management System (WMS) used in the water supply, as shown in Fig. 4.9.

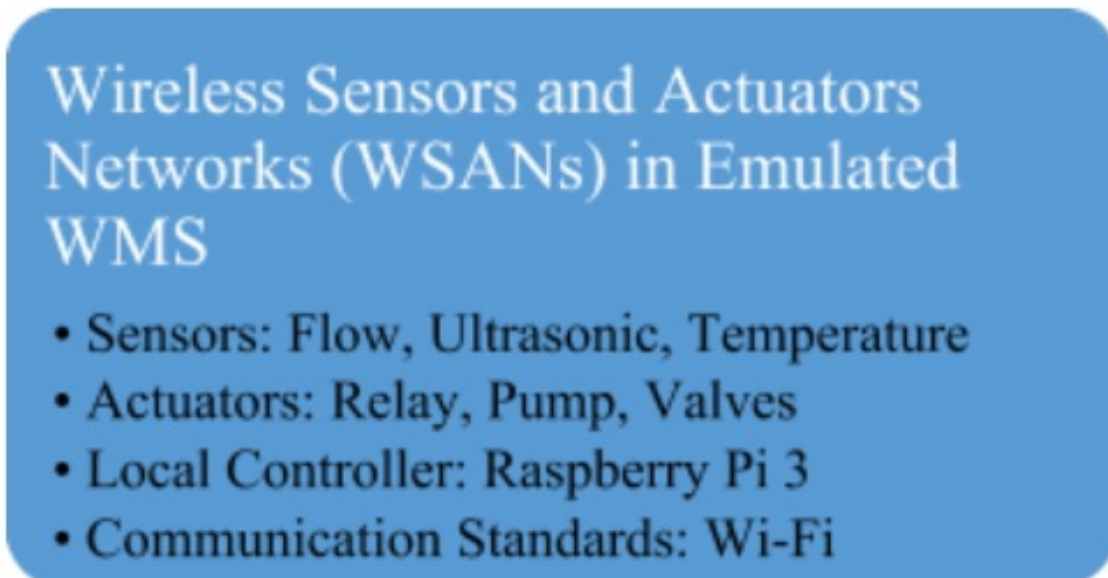


FIGURE 4.9: Wireless Sensors and Actuators Networks (WSANs) Based Water Management System

Real WMS consists of sensors, actuators, and controller. This emulated system also consists of actuators, sensors, and controllers. In our emulated system, relay, pump, and valves are actuators. There are three sensors in our emulated system. A flow sensor measures the rate of water flow. Level, Temperature of water is measured by ultrasonic sensor and temperature sensor, respectively. Raspberry Pi 3 is a controller. Then we host, this emulated system on a website called <https://cps4wm.info>. On the click of the website, we are in a position to on/off the pump. Then, we have performed an attack on that using four tools, as shown in Table 4.2. We are using the Common Vulnerability Scoring System (CVSS) to deals with four threats, we have found during attacks on water management system (WMS). Then finally, we proposed a solution for every loophole in the security of WMS.

TABLE 4.2: Security Attacks on Emulated Water Management Cyber-Physical System

Security Attacks	Tools Used
Information Disclosure	Firefox Analyzer, Wappalyzer
Scanning: Accessible Folder	DirBuster
Exploitation: SQL Injection	SQLMAP
Exploitation: DOS	SQLMAP

4.4.1 Information Disclosure

Service was found to be running that provides detailed version information. This information is used to determine what vulnerabilities may exist in the service, assisting malicious users in launching more targeted attacks. It reports the code of the directly browsed PHP scripts: `/updatevalue.php` and `/getdata.php`. The script `/updatevalue.php` can be used to upload arbitrary value in the waterflow table, and the script `/getdata.php` can be used to show this value.

TABLE 4.3: Common Vulnerability Scoring System (CVSS) with Severity 1

Severity	CVSS	Published	Modified
1	(AV:N/AC:L/Au:N/C:N/I:N/A:N)	April 01, 2006	December 03, 2013

Common Vulnerability Scoring System (CVSS) with Severity 1 is shown in Table 4.3. The solution is Remove critical information from the web page.

4.4.2 Forced Browsing - Accessible Folder

Forced browsing is an attack where the aim is to enumerate and access resources that are not referenced by the application but are still accessible. These resources may store sensitive information about web applications and operational systems, such as source code, credentials, internal network addressing, and so on, thus being considered a valuable resource for intruders. This attack is performed manually when the application index directories and pages are based on number generation or predictable values or using automated tools for common files and directory names. In our work, we have used DirBuster tools. `/images/` and `/cgi-bin/` are the most discovered folders. The directory `/images` are directly accessible. We conclude that it is not directly possible to exploit it for obtaining access to the target, but it could be used as the point of access in the case a backdoor will be upload in the server. CVSS version 3.0 score is 5.3. Metric and value associated with forced browsing, as shown in Table 4.4. The solution is to make images folder no more reachable.

TABLE 4.4: Forced Browsing: Accessible Folder

Metric	Value
Attack Vector	Network
Attack Complexity	Low
Privileges Required	None
User Interaction	None
Scope	Unchanged
Confidentiality Impact	Low
Integrity Impact	Low
Availability Impact	Low

4.4.3 Blind SQL Injection - SQL-Injection

Web applications usually store information in a SQL server to show them to other users. When the application developer uses unvalidated user-controlled variables as part of a SQL query; a SQL injection or Blind SQL injection vulnerability is being introduced into the application. When an attacker executes SQL Injection attacks, sometimes the server responds with error messages from the database server complaining that the SQL Query’s syntax is incorrect. The blind SQL injection is similar to standard SQL Injections except that when an attacker attempts to exploit an application, rather than getting a potentially useful error message, they get a generic page specified by the developer instead. This makes exploiting a potential Blind SQL Injection attack more difficult but not impossible. An attacker can still retrieve valuable information and potentially execute operating system commands by issuing a set of True and False questions through SQL statements. Sqlmap is the tool used for SQL injection. `Sqlmap -u "http://*.*.**/updatevalue.php?val=1" -dump -level 1` is a command. The entry point for SQL-injection is the script: `/updatevalue.php?val=x`. Following is an example of attack vector: `Web-application-ip/updatevalue.php?val=3+where+1=1+-IF((SELECT+1+from+ information_schema.tables WHERE table_name REGEXP '[n-z]'limit 0, 1) =1, +SLEEP(10),+0)`. With blind SQL injection, it has been possible to discover the following information: table_schema called `cpswminf.project` and a user called `cpswminf.pandey`. Furthermore, the user `cpswminf.pandey` has privileges `USAGE`. Common Vulnerability Scoring System (CVSS) with Severity 9 is shown in Table 4.5. The solution is Input validation, parameterised queries and use of the stored procedure.

TABLE 4.5: Common Vulnerability Scoring System (CVSS) with Severity 9

Severity	CVSS	Published	Modified
9	(AV:N/AC:L/Au:N/C:C/I:P/A:P)	December 31, 2004	June 19, 2013

4.4.4 Delayed SQL Execution (MySQL): DOS

SQL injection techniques analyze the application’s response to parameter values that are designed to be interpreted and executed by a database. These requests contain arguments that are not affected by input validation filters. The application submits the original payload to the database, where the database interprets the payload as a valid SQL query. This implies that arbitrary SQL commands may be executed through this parameter value. These tests do not

generate database errors. No database errors should appear in the HTML response. Vulnerabilities identified by this module highlight problem with input validation routines and the creation of SQL queries. Common Vulnerability Scoring System (CVSS) with Severity 10 is shown in Table 4.6.

TABLE 4.6: Common Vulnerability Scoring System (CVSS) with Severity 10

Severity	CVSS	Published	Modified
10	(AV:N/AC:L/Au:N/C:C/I:C/A:C)	December 31, 2010	Sept.30,2016

The DOS is realized with the previous SQL-injection inner blind attack vectors setting a large SLEEP time. The SLEEP () function sets in waiting status the connection established with MySQL server for the specified time. For large values, the RAM, cache and other resources used by MySQL became unbearable for the system. The DOS caused full-service interruption, and the web page has out of reach. To restore normal behaviour of the web application it has needed to restart all system. Solutions are input validation; parameterized queries use of stored procedures.

4.5 Future Cybersecurity Practices for the Water Sector

Project initiated by Water Utility Council of American Water Works Association (AWWA) delivers step-by-step guidance to save WMS during cyber-attacks. A panel of industry and academia was consulted to identify the most common cybersecurity issues in front of the water system. To deal with these issues, a list of recommended cybersecurity practices was developed, as shown in Fig. 4.10. This list is the framework to deal with risk in the WMS [12].

4.6 Conclusion and Future Scope

This work analyses, compares and discusses seven real examples of cyber threats on WMS and also seven real examples of physical threats on WMS. To create risks for human life, all chemical and physical characteristics of drinking water always manipulated by a hacker in WMS. Therefore, Ions and the physical sensor will help us to detect cyber-attack by monitoring of both chemical and physical characteristics features of drinking water. We have also emulated a small version of WMS with small sensors, actuators, and controller of real WMS of large sensors, actuators, and controller. We have also performed attacks on our emulated system to understand the threats on real WMS. We have also proposed the solution to fill the security loopholes in real WMS. We have also analysed the compatibility of our proposed solution with recommended cybersecurity practices developed by the American Water Works Association (AWWA). Therefore, work in the area of cybersecurity of water management CPS will have to update in continuous manners as per line of English poet Robert Frost "But I have promised to keep and miles to go before I sleep." Hence every endeavour of a cybersecurity expert to develop countermeasure of cyber terrorism is desirable in the long run of the future. In future, we shall also do attack and fault modelling of every sensor and actuators used in WMS to develop secure sensor network for WMS.



FIGURE 4.10: List of Recommended Cybersecurity Practices

If there is magic on this planet, it is contained in water.

Loren Eiseley

5

Serial Communication among Controllers, Sensors and Actuators in Emulator of Water management Cyber Physical System (WM-CPS)

5.1 INTRODUCTION

Raspberry Pi and Arduino are the two most famous open source boards among electronics researcher and scientist due to their easiness and simplicity. Water Management Cyber-Physical System (WM-CPS) has a sensor and actuator network (SAN) made of two sensors and two actuators. We are using an open source board as a controller, sensors for reading information and actuators triggered by the controller in SAN. Pump and relay are two actuators. Flow sensor and Level sensor are two sensors. In SAN, Raspberry Pi 3 (RPi-3) and Arduino Mega2560 are two controllers taken into consideration, as shown in Fig. 5.1. Raspberry is a master controller, and Arduino is slave controller. I2C, SPI, and Serial are three options to connect RPi-3 and Arduino. Problem with I2C and SPI is that we need a logic level converter because Pi uses 3.3V and Arduino 5V. Connecting 5V to the Pi would destroy it immediately. Therefore, the simplest way is Serial via USB [54]. We have established a serial communication between Raspberry Pi3 and Arduino Mega2560 by connecting Mega2560 to the RPi-3 USB port using the USB cable. All sensors and actuators are directly connected with Arduino. Raspberry can communicate with both sensor and actuators via Arduino. We have written the program in the Arduino IDE software and upload the program to the Mega2560. After that, we have also written a program for RPi, to read the data already sent by a sensor on Arduino Mega2560. RPi program also write command for Actuator on Arduino Mega2560. The base station is made of two sensors and two actuators. Sensor and actuators are controlled by Arduino and Raspberry communicate with Arduino, as shown in Fig. 5.1. Three Arduino will see as ttyACM0, ttyACM1 and ttyACM2 in Raspberry. ACM0 is Arduino associated with the level sensor. ACM1 is Arduino associated with the flow sensor. ACM2 is Arduino directly deals with relay and indirectly deals with the pump.

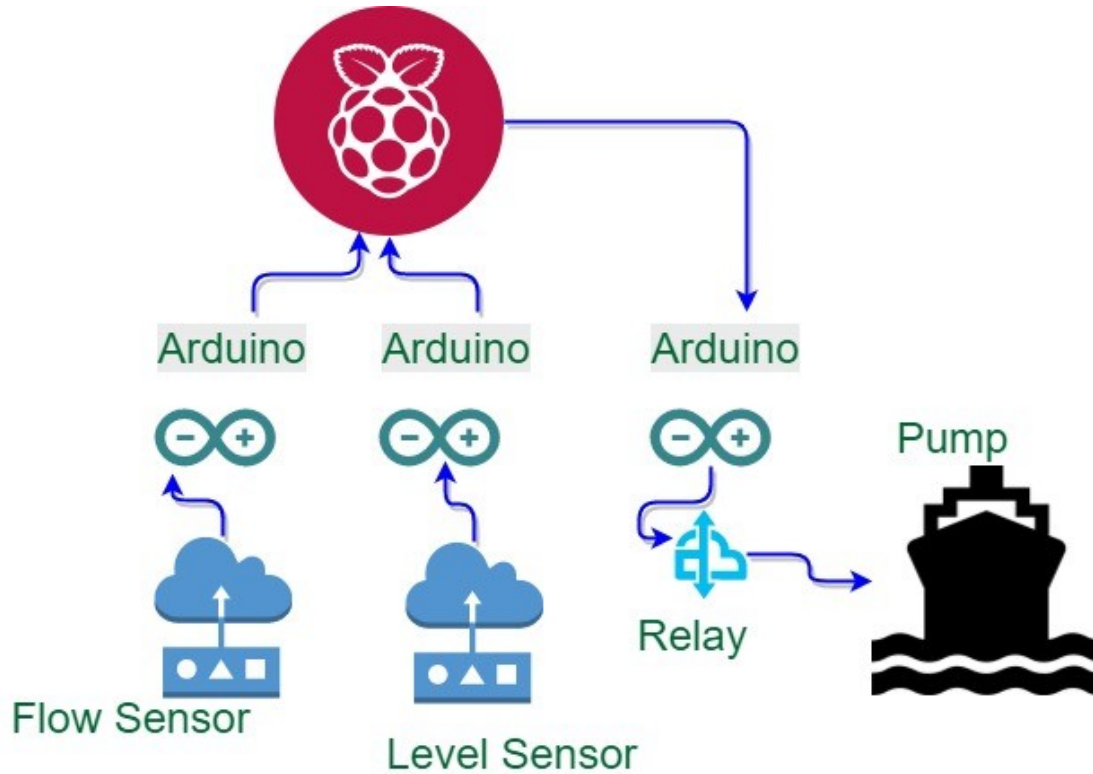


FIGURE 5.1: Serial Communication in Sensor and Actuator Network

We have created multiple tables in MYSQL database to store both readings of sensors and commands for actuators. We have also designed web-interface using PHP language and hosted our website as VPS. The website is accessible in <http://cps4wm.info> domain.

5.2 OVERALL SYSTEM ARCHITECTURE

Building a sensor actuators network system requires the development and integration of many hardware component and software program. Fig. 5.2 shows Arduino and Fig. 5.3 shows Raspberry used in the system architecture of a water management cyber-physical system that we have developed.

MEGA2560 R3 Black connect echo ofBoard has ATmega16U2 chip. It can connect to the computer via the USB cable or power it with an AC/DC adapter or a battery.

In addition to 1.2 GHz Quad Core CPU, Raspberry PI 3 Model B has one GB RAM, one HDMI port, and four USB port.

Channel relay shield module, as shown in Fig. 5.4 conform international safety standard, the isolation groove between control areas and load area. We connect Pump with the relay. If we power on the relay, then pump will on and if we power off relay then pump will off.

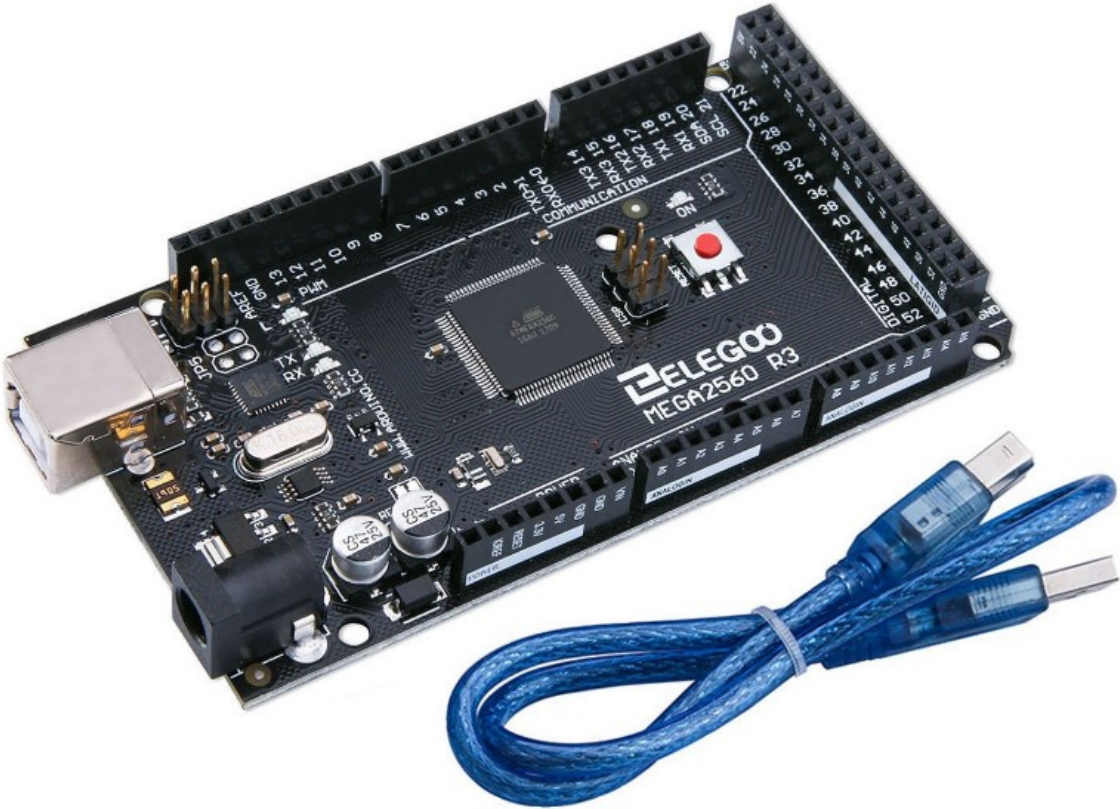


FIGURE 5.2: MEGA2560 R3 Arduino Board with ATmega16U2 chip [11]



FIGURE 5.3: Raspberry PI 3 Model B with 1.2 GHz Quad Core CPU [66]



FIGURE 5.4: XCSOURCE 5V/9V/12V/24V Channel Relay Shield Module

5.3 DESIGN OF HARDWARE AND SOFTWARE COMPONENTS

5.3.1 Design of Sensor Node

In this research, we developed networked sensor nodes using Arduino and HC-SR04 ultrasonic sensor. Arduino is the most used open-source single-board microcontroller development platform. Arduino Mega2560 is based on Atmel Atmega16U microcontroller. Atmega16U is 8-bit RISC-based microcontroller combines 16KB program memory, 512-Byte EEPROM, 512-Byte SRAM, 22 general purposes I/O lines, and 32 general purpose working registers. We connect the echo of HC-SR04 at Pin No. 7 and trigger of HC-SR04 at Pin No. 8 of Maega2560 Arduino. In this chapter, we developed networked sensor nodes using Arduino and YF-S201 flow sensor.

5.3.2 Design of Actuator Node

In this research, we developed networked actuator nodes using Arduino, relay, and Pump as shown in Fig. 5.5.

5.4 SERIAL COMMUNICATION Among CONTROLLER, Sensor Node and Actuator Node

We connect Arduino and Raspberry Pi (RPi) using the USB cable and check the connection between Arduino and RPi by typing `$ ls /dev/tty*` in RPi terminal. Result is `/dev/ttyACM0`. We write code for both Python and Arduino, as shown in Table 5.1 and 5.2.

5.4.1 Programming the Arduino from a Raspberry Pi

Arduino IDE is installed in RPi to allow us to use it to program the Arduino by the following command: `$ sudo apt-get install arduino` After installation, Arduino IDE found on the application menu under programming and electronics. We

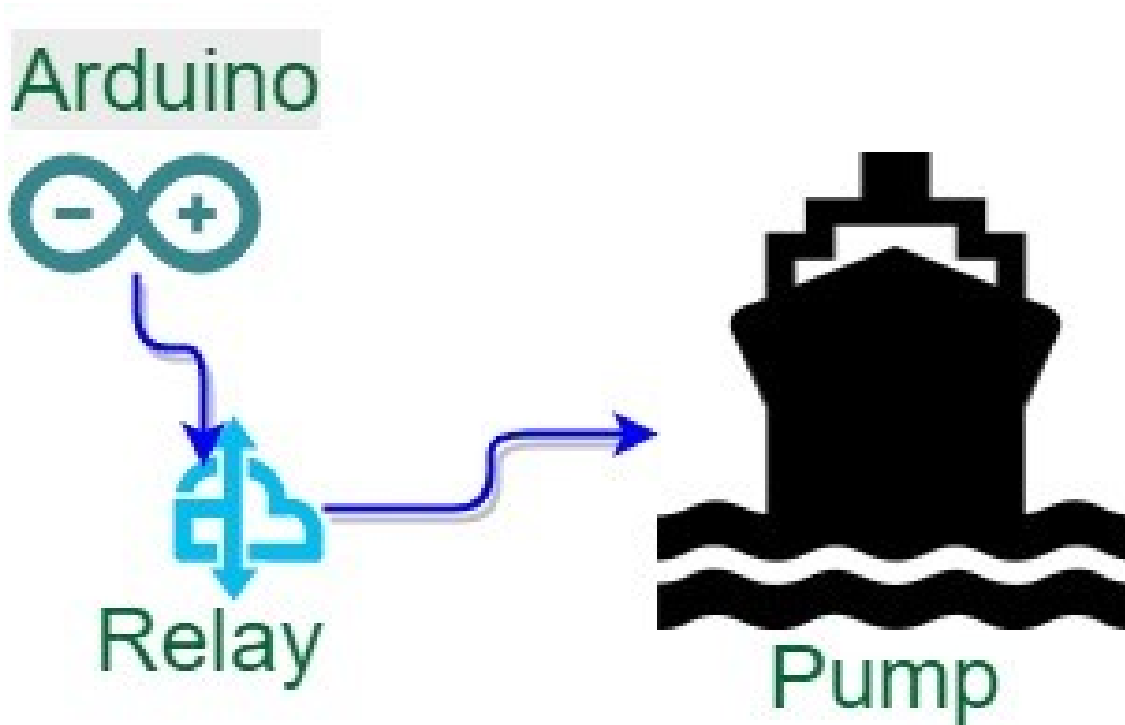


FIGURE 5.5: Actuator Node: Relay for Start and Stop of Pump

TABLE 5.1: Code of Level Sensor for Arduino written in Python

```
#define echoPin 7
#define trigPin 8
int duration, distance;

void setup()
{
  Serial.begin(9600);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}
void loop()
{
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration=pulseIn(echoPin, HIGH);
  distance=duration/58.2;
  Serial.println(distance);
}
```

TABLE 5.2: Code of Level Sensor for Raspberry Pi written in Python

```
import serial
ser = serial.Serial('/dev/ttyACM0', 9600)
ser.readline()
```

compile the Arduino code, as shown in Table 1. Then we select the appropriate model of Arduino from Tools -> Board. We also select the appropriate Serial Port: /dev/ttyACM0. Eventually, we load compiled code on Arduino.

5.4.2 Serial communications to the Raspberry Pi for Sensor

Python includes a library for communicating with serial devices called serial. We have created code in Python (as shown in Table 1). Python code enables the serial communication of 9600 bps (bits per second) on ACM0 port, look for a signal from the Arduino, and print the distance received.

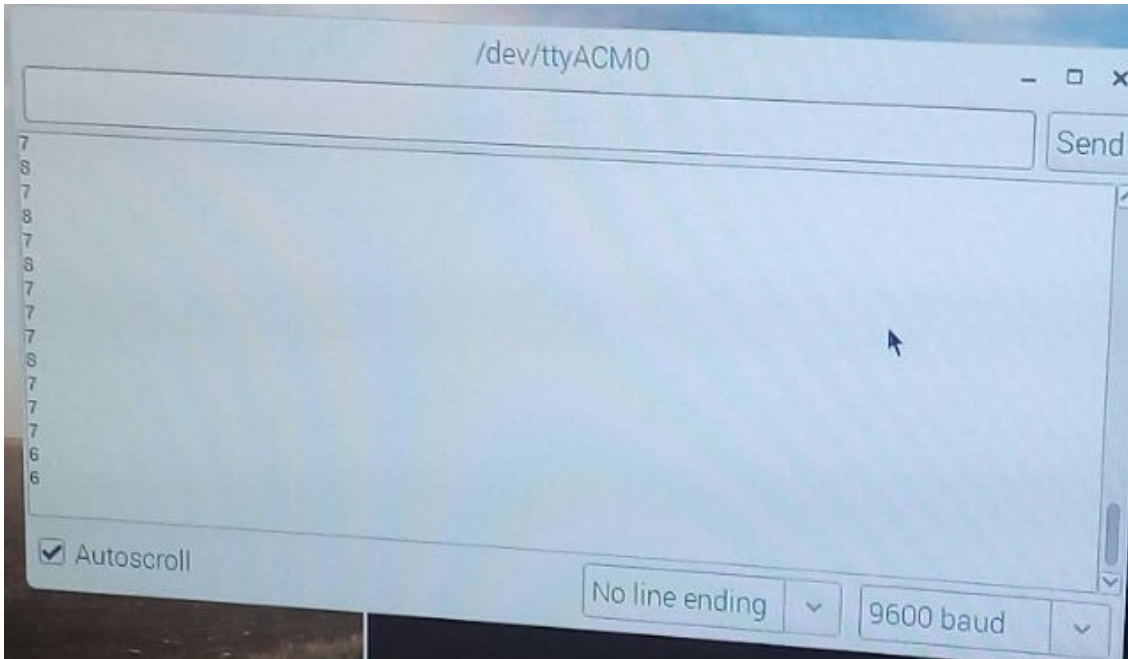


FIGURE 5.6: Level Sensor Reading on Serial Monitor

At one point in time, distance is eight also observed on RPi terminal and Serial Monitor as shown in Fig 5.6- 5.7.

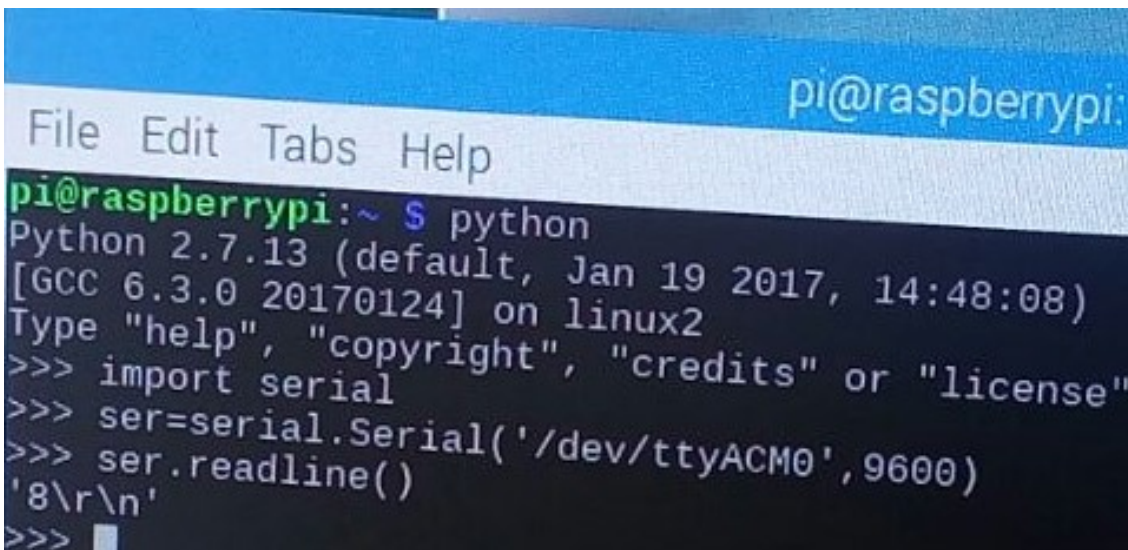


FIGURE 5.7: Level Sensor Reading on RPi Terminal

TABLE 5.3: Serial Communication between Arduino with Flow Sensor and RPi

<pre> Arduino Code for Flow Sensor volatile int flow_frequency unsigned int l_hour; unsigned char flowsensor = 2; unsigned long currentTime; unsigned long cloopTime; void flow () { flow frequency++; } void setup() { pinMode(flowsensor, INPUT); digitalWrite(flowsensor, HIGH); Serial.begin(9600); attachInterrupt(0,flow, RISING); sei(); currentTime = millis(); cloopTime = currentTime; } void loop () { currentTime = millis(); if(currentTime>=(cloopTime + 1000)) { cloopTime = currentTime; l_hour = (flow_frequency * 60/7.5); flow_frequency = 0; Serial.print(l_hour, DEC); Serial.println(" L/hour"); } </pre>
<pre> Raspberry Pi Code written in Python import serial ser = serial.Serial('/dev/ttyACM1', 9600) ser.readline() </pre>

The sensor data pin is connected with Pin 2 of Arduino. The program is shown in Table 5.3 measure and print litre/hour every second, as shown in Fig. 5.8. Pulse frequency is 7.5Q, where Q is the flow rate in L/min. We define a variable called flow_frequency to store the measured value of flow sensor pulses.

On raspberry Pi terminal, reading is '24 L/hour\r, \n', '16 L/hour \r, \n', '8 L/hour\r, \n', '16 L/hour\r, \n', '0 L/hour\r, \n', and so on as shown in Fig. 5.8.

```

pi@raspberrypi:$ python
»>import serial
»>ser=serial.Serial ('/dev/ttyACM1', 9600)

```

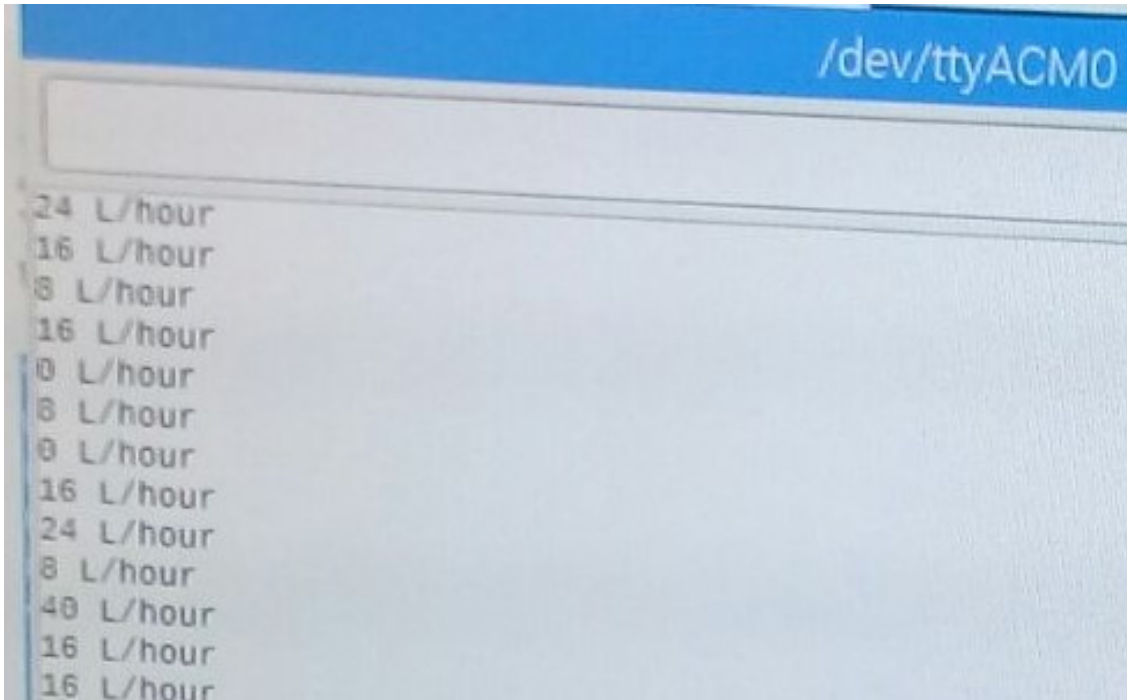


FIGURE 5.8: Flow Sensor Reading on Serial Monitor of Arduino

```
»>ser.readline ()
```

```
'24 L/hors\r, \n'
```

5.4.3 Serial communications from the Raspberry Pi for Actuator

The relay input pin is connected with Pin 5 of Arduino. When Pin 5 is HIGH then the relay will start the PUMP. PUMP will off when Pin 5 is LOW.

Code of getData.php return value stored in waterflow MySQL table. This MySQL table store 1 with the click of Pump ON button and store 0 on click of Pump OFF button. Both Pump ON and Pump OFF button is available on <http://www.cps4wm.info>. Python code reads the value return by getData.php as shown in Table 5.4 and based on value, Raspberry send a command to on/off relay associated with Arduino via serial communication at 9600 rate.

5.5 CONCLUSION AND FUTURE SCOPE

Serial communication is possible between controllers in water management cyber physical system. Raspberry is a master controller, and Arduino is slave controller. Both flow sensor and level sensor are connected with Arduino microcontroller and send sensor reading to Raspberry Pi. Raspberry read sensor reading associated with Arduino by using serial communication. Pump and relay are actuators connected to Arduino. Raspberry execute a command on Arduino for

TABLE 5.4: Serial Communication between Arduino with Actuator and RPi Arduino Code for Relay-PUMP

<pre>const int pump=5; char command; void setup() { Serial.begin(9600); pinMode(pump, OUTPUT); } void loop() { if (Serial.available()>0) command=Serial.read(); int relay=command-'0'; if(relay==1) digitalWrite(pump, HIGH); else digitalWrite(pump, LOW); }</pre>
<pre>Raspberry Pi Code written in Python import serial, requests, json ser=serial.Serial('/dev/ttyACM0',9600) while True: response=requests.get(http://www.cpps4wm.info/getData.php) json_data=json.loads(response.text) if (json_data=='1'): ser.write('1') else: ser.write('2')</pre>

starting and stopping of a pump using serial communication. Command passed from Raspberry is based on the click of ON/OFF Pump button on <http://www.cps4wm.info>.

Anyone who can solve the problems of water will be worthy of two Nobel prizes- one for peace, and one for science.

John F Kennedy

6

Implementation of RS485 Modbus and TCPI/IP in Emulator of Water Management Cyber Physical System

6.1 INTRODUCTION

MODBUS is one of the most famous application layer protocols used in industrial communication due to its simple request-response communication session and its readiness to be implemented on top of several lower level communication protocols and various physical media such as RS232/RS485 serial lines and Ethernet TCP/IP [59]. In this work, Modbus is used for communication among flow sensor, level sensor, and temperature sensor along with pump-relay actuators of our proposed Water Management Cyber-Physical System (WM-CPS). RS485 interface is used in combination with Universal Serial Bus (USB) and Universal Asynchronous Receiver Transmitter (UART) interface for communication between Arduino and Raspberry Pi. Arduino is connected to UART to RS485 Modbus converter. Whereas, Raspberry Pi (RPi) is connected to USB 2.0 to Modbus RS485 converter. When we connect USB-Modbus converter to RPi, the rasbian operating system recognises it as a `/dev/ttyUSB0`. Python library `serial` is used to connect to `/dev/ttyUSB0` as `ser=serial.Serial('/dev/ttyUSB0', 9600)`. For a reading of the sensor, we use `ser.read()` command on RPi. To send the command to an actuator, we use `ser.write('')`. USB 2.0 to RS485 serial converter converts USB 2.0 to RS485, as shown in Fig. 6.1. Its transmission distance is up to 1000 meters. This adapter is made of CP2102 and MAX13487 industrial chip. The CP2102 chip is USB to UART Bridge. The MAX13487 is RS-485 compatible transceivers. This USB-RS485 converter is a transceiver. It sends a command to Actuator connected to Arduino, and it also receives sensor reading from the sensor connected to Arduino. Arduino is acting as a slave controller. Raspberry is a master controller. When Raspberry transmits then Arduino receive and vice versa. The flow sensor, Temperature sensor, and Level sensor are three sensors taken into consideration in our design work of WM-CPS. Pump-Relay is actuators in this WM-CPS.

Formulation of actuator command is based on a click on On/OFF PUMP button on <http://cps4wm.info>. The flow of actuator command is shown as a blue line. The flow of sensor reading is shown as a green line as shown in Fig. 6.2.



FIGURE 6.1: USB 2.0 to Modbus RS485 Converter

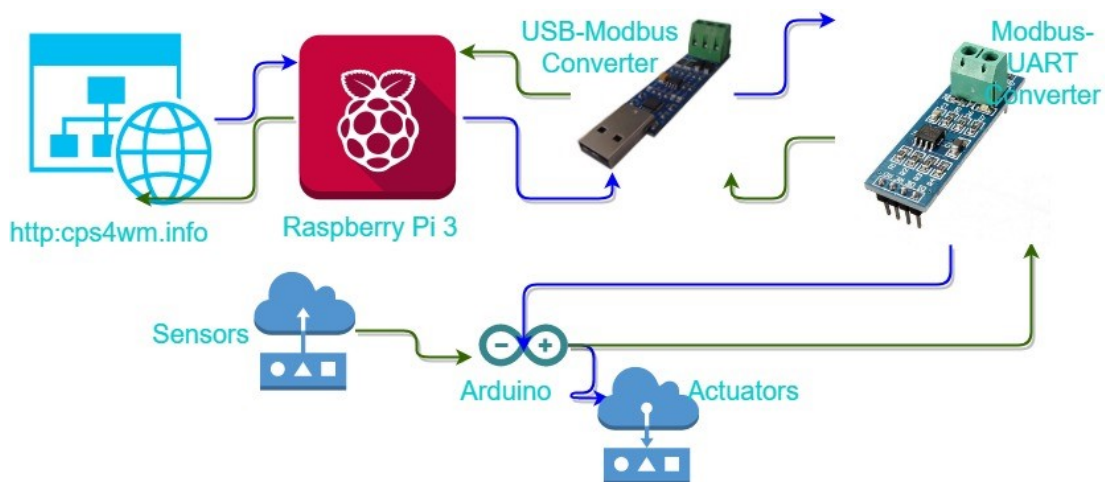


FIGURE 6.2: Communication via USB-Modbus and Modbus-UART

RS485, UART, and USB are three different interfaces used in our Water Management Cyber-Physical System (WM-CPS). Range, number of the receiver, mode, ICs of the different interface are shown in Table 6.1.

TABLE 6.1: Different Variety of Interface

Type	Range	Receiver	Mode	ICs
RS232	10m	10	Single end	MAX232
RS422	1000m	10	Differential	MAX490
RS485	1000m	32	Differential	MAX485
UART	-	-	-	-
USB	5m			

RS485 is 2-wires based half duplex communication standard over longer distances than RS232 or TTL. RS232 is full duplex. RS422 is 4-wires based full duplex communication standard. It supports 32 units on the same bus (Multi Drop). We can simulate full-duplex communication also if we merge two RS485 modules: one for the transmitter and other for the receiver.

Fig. 6.3 - 6.4 are the same Modbus to TTL converter. These boards have an onboard MAX485 chip. Their working voltage is 5 V. These are the tool for RS-485 communications, low-power, and Slew-Rate Transceiver. Its board size is 44 (mm) x14 (mm) x20 (mm).

The two ends of the module have 8 Pins: R0, DE, RE, DI, VCC, B, A, and GND, as shown in Table 6.2. The TX/RX lines UART's available in MCU are connected to the DI/RO pins on the interface board.

TABLE 6.2: Pin of TTL to RS485 Converter

Rear Side	Front Side
R0: Receive Output	VCC
DE	B
RE	A
DI:	Data Input GND

When we connect TX_EN to VCC and DI to transmission pin (TX-OUT) of microcontroller unit (MCU), then RS485 module act as a transmitter as shown in Table 6.3. To use this module as the receiver, we have to connect TX_EN to GND and R0 to receiver pin (RX-IN) of MCU, as shown in Table 6.3. In our work, Arduino is an MCU connected to TTL-Modbus RS485 converter. B is the negation of A.

The RE pin is active when connecting to GND, and the DE pin is active when connecting to VCC. That's why these pins must wire together. In our work, combined pins are said to be TX_EN. When TX_EN pulled High; the RS485 disables receiving and enables transmitting. When TX_EN set low, RS485 enables receiving and disables transmitting. Modbus TCP/IP mode of transmission is possible in Raspberry Pi with the help of two python Library mainly PyModbus and Twisted and free from integrated circuits like MAX232, MAX490 and MAX485.

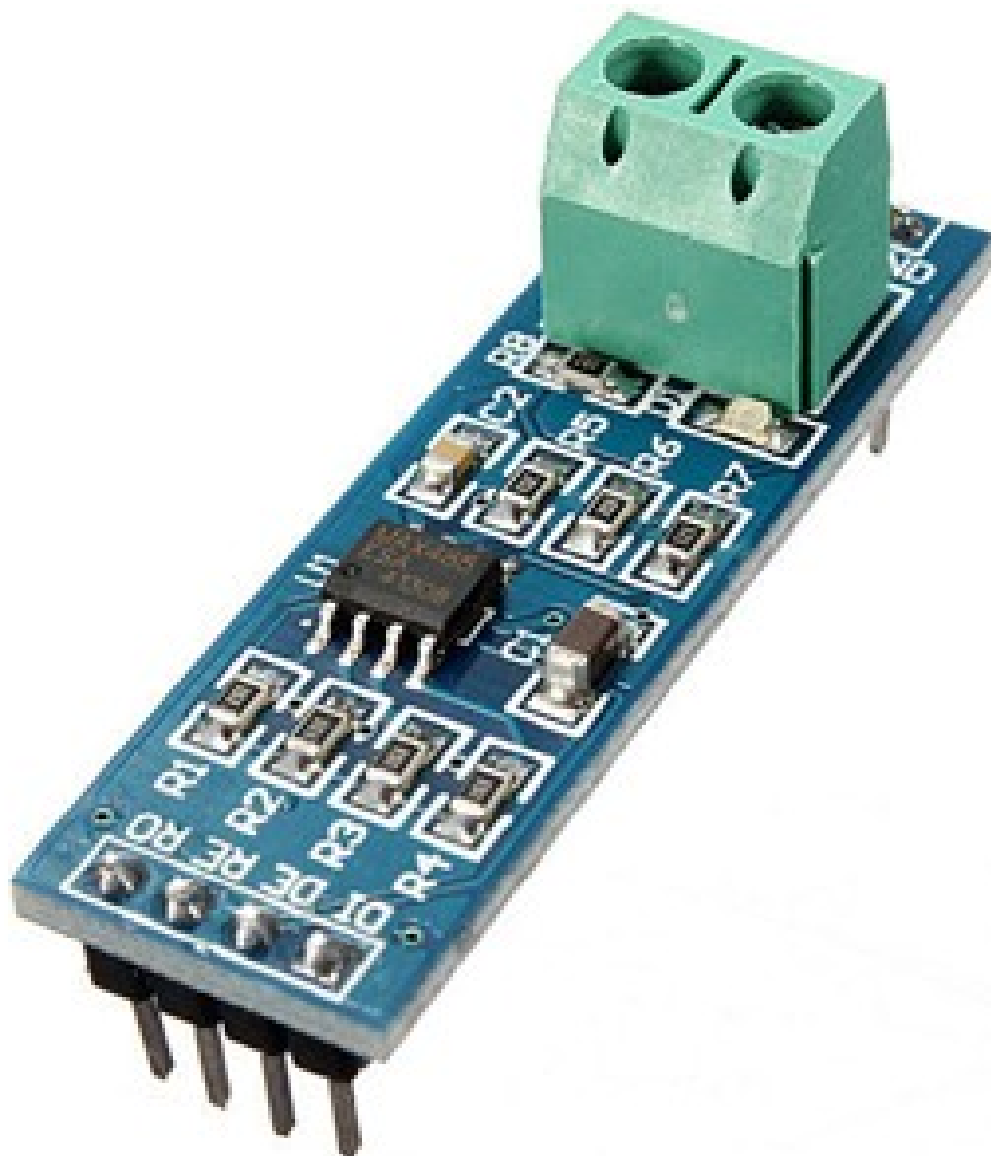


FIGURE 6.3: Rear side of TTL to Modbus RS485 Converter Module

TABLE 6.3: Interconnection of MCU and RS485 module

MCU	RS485
TX-OUT	DI-Data Input
RX-IN	RO-Receive Output
TX_EN	RE-DE pins wired together is known as TX_EN

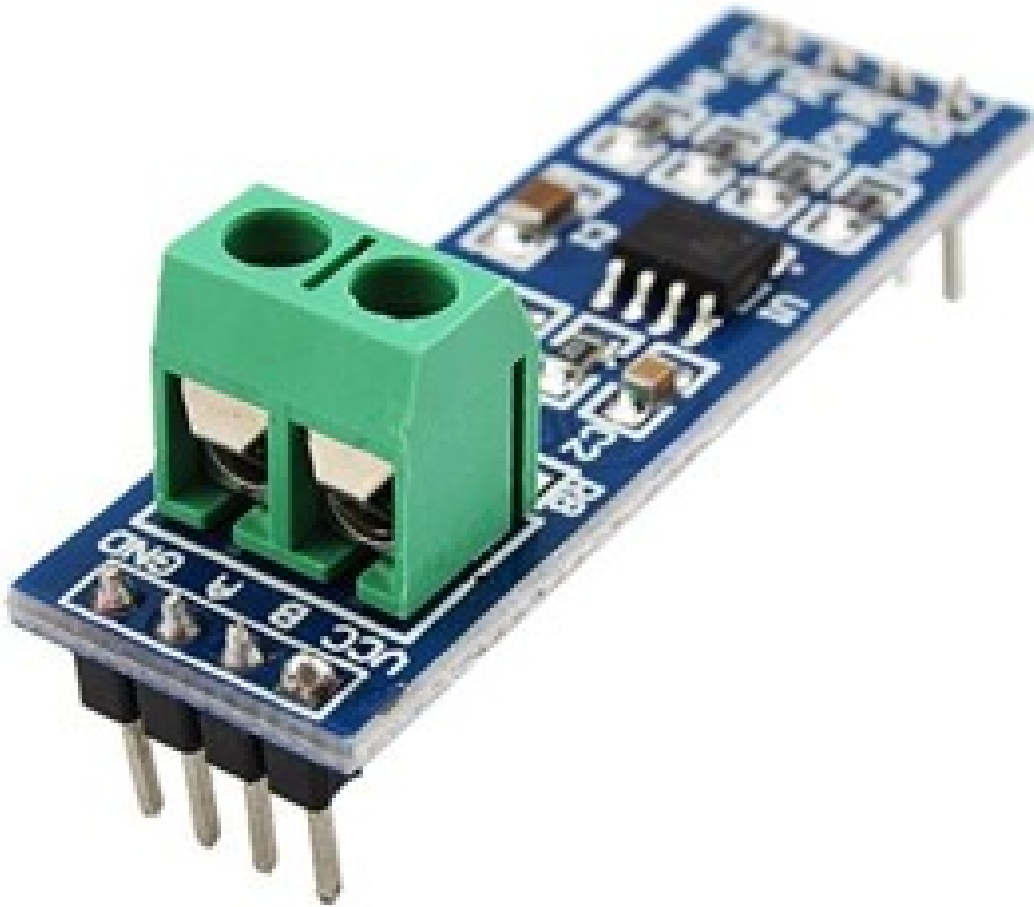


FIGURE 6.4: Front side of TTL to Modbus RS485 Converter Module

6.2 MODBUS COMMUNICATION: HARDWARE BASED

Raspberry Pi-3 has both Bluetooth and Wireless LAN connectivity. It automatically connects with eduroam network available in both the Italian university campus of Gran Sasso Science Institute and Politecnico di Torino. Eduroam is a worldwide roaming facility available for researcher and scientist in more than 72 countries. So, our system can work in any partner universities of eduroam. When we connect USB to RS-485 Modbus converter with Raspberry, the Rasbian operating system installed on Raspberry recognised it as `/dev/ttyUSB0`. To read the signal from RS-485 on Arduino, we pass the signal to the RS485-UART converter. Eventually, Arduino read a signal from Raspberry via USB-Modbus-Modbus-UART and ON/OFF Pump, as shown in Fig. 6.5 and Fig. 6.6 and Table 6.4.

Python use JavaScript Object Notation (JSON) and get reading of click on ON/OFF Pump button on `http://cps4wm.info` as `response=requests.get(http://cps4wm.info/getData.php)` and `json_data=json.loads(response.text)`. If `json_data` is ON, RPi sends 1 to Arduino as `ser=serial.Serial('/dev/ttyUSB0', 9600)` and `ser.write('1')` in Python via USB-Modbus-UART to start Pump otherwise pump will not start.

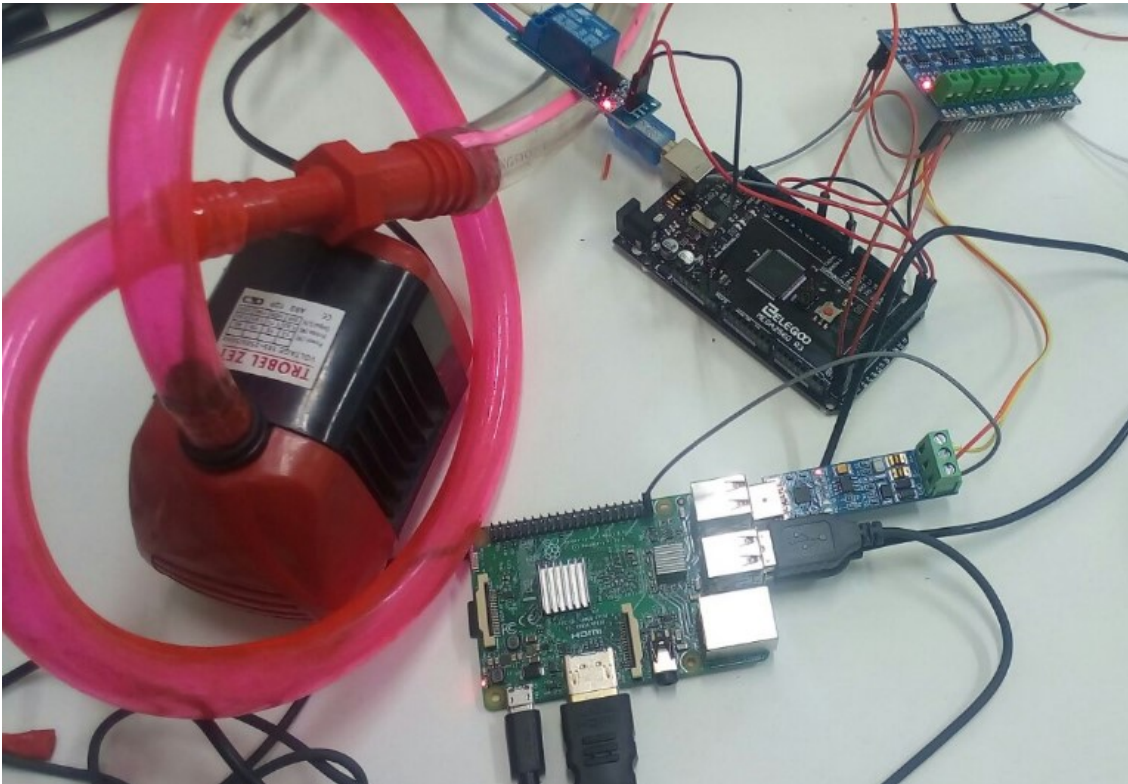


FIGURE 6.5: Modbus in Water Management Cyber-Physical System: Actuator

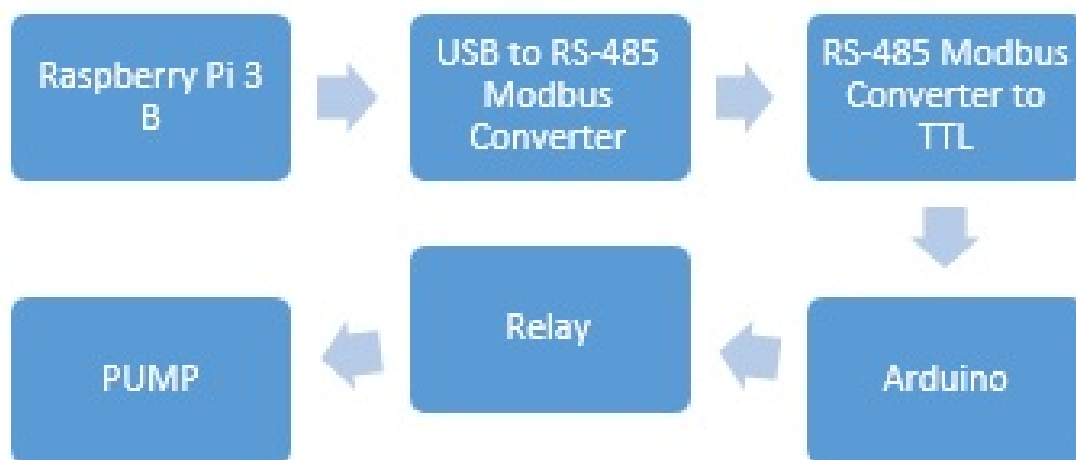


FIGURE 6.6: Work Flow of Water Management Cyber Physical System: Actuator

TABLE 6.4: Pin setting when Raspberry is Transmitter, Arduino is Receiver

Raspberry USB 2.0	Modbus	Modbus	UART	Arduino
		GND		GND
	A	A		
	B	B		
		VCC		VCC
GND Pin 6	GND		DI	
			DE	GND
			RE	GND
			RO	RXO
Message Sent				Message Received
ser.write('1')				Pin 5 High
ser.write('2')				Pin 5 Low

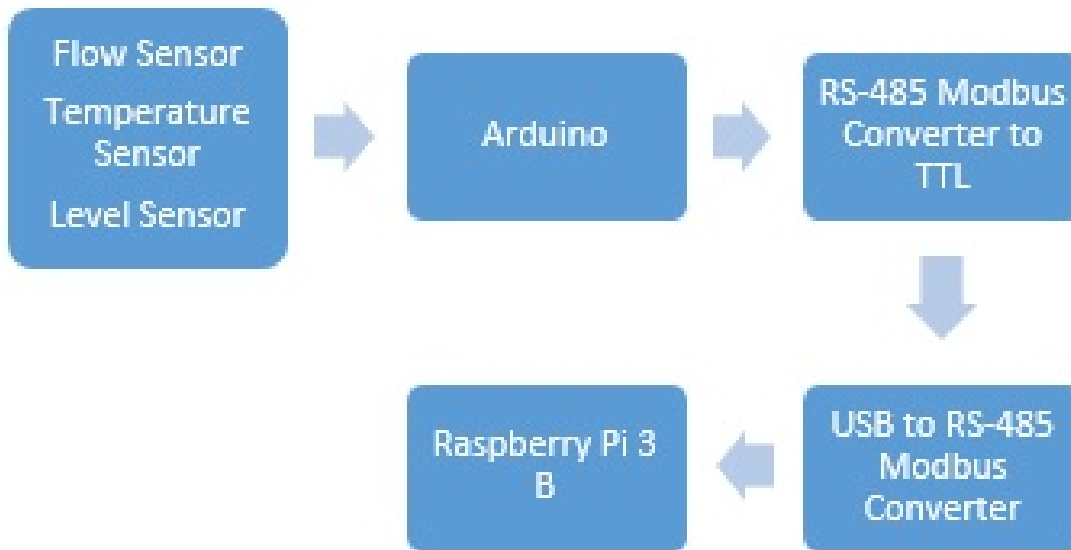


FIGURE 6.7: Work Flow of Water Management Cyber-Physical System: Sensor

Arduino will receive a reading of sensor and transmit reading to UART-Modbus Converter as shown in Fig. 6.7 and Table 6.5. On this stage, the format of reading of sensors will change into Modbus data format. This Modbus data will pass to Modbus-USB converter. Eventually, RPI receives data in serial communication.

6.3 MODBUS COMMUNICATION: SOFTWARE BASED

This system is using pymodbus and twisted library written in python. We call StartTCPServer, ModbusDeviceIdentification, and ModbusSequentialDatablock from server.sync, device and datastore packages respectively. ModbusSlaveContext and ModbusServerContext are called from datastore packages of pymodbus library. Also, ModbusRtuFramer and ModbusASCIIFramer are called from the transaction package of pymodbus library. RaspberryPi: `sudo pip install pymodbusRaspberrypi` : `sudo pip install twisted`

TABLE 6.5: Pin setting when Raspberry is Receiver and Arduino is Transmitter

Raspberry 2.0	USB	Modbus	Modbus	UART	Arduino
			GND		GND
		A	A		
		B	B		
			VCC		VCC
GND Pin 6		GND		DI	TXO
				DE	VCC
				RE	VCC
				RO	
Print 5					Level is 5
ser.readline()					Pin 7 ECHO
ser.read()					Pin 8 Trig
Message Received					Message Sent

First, we need to start the TCP server on every slave. StartTCPServer(context, identity=identity, address= ("192.168.43.103", 5020)) is python statement to start purifier. Fig. 6.8 shows the output of the slave purifier.

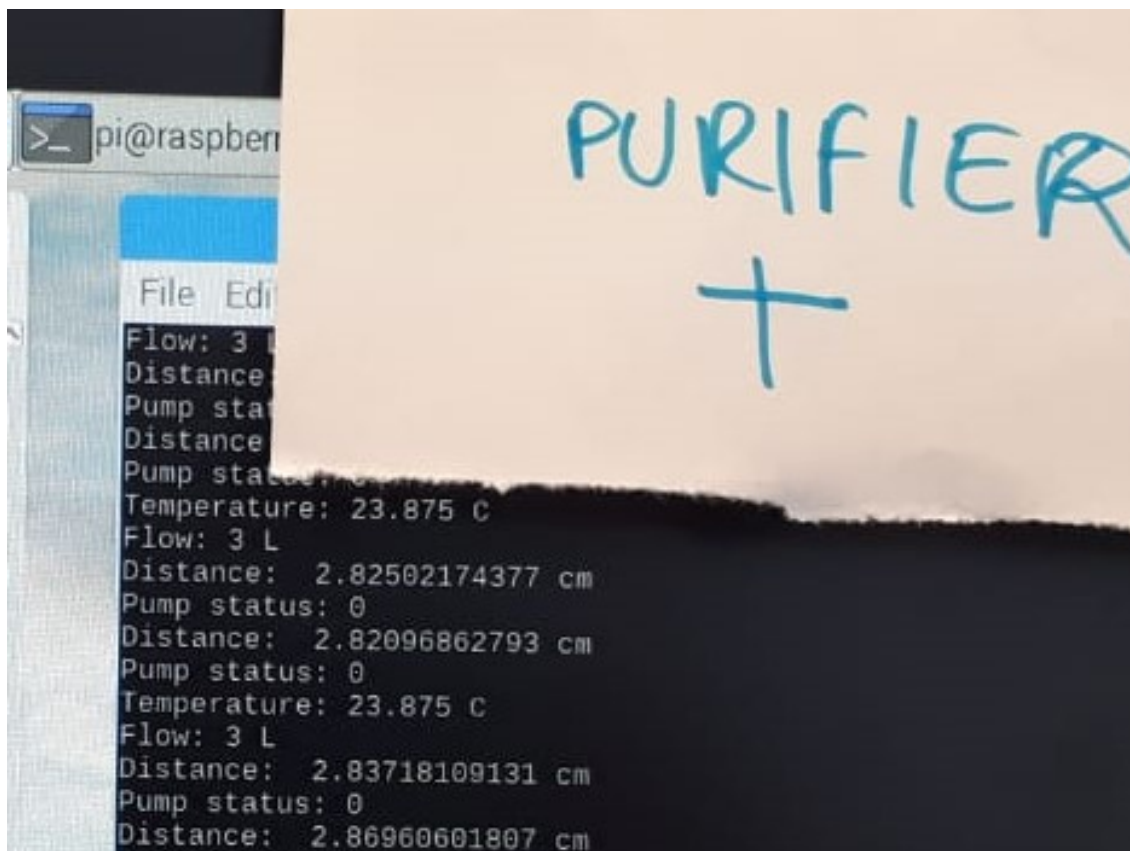


FIGURE 6.8: Output of Slave Purifier

Figure 6.8 shows output of the Lake Purifier.

To start TCP Modbus server we use the following line: StartTCPServer(context, identity=identity, address= ("192.168.43.102", 5020))

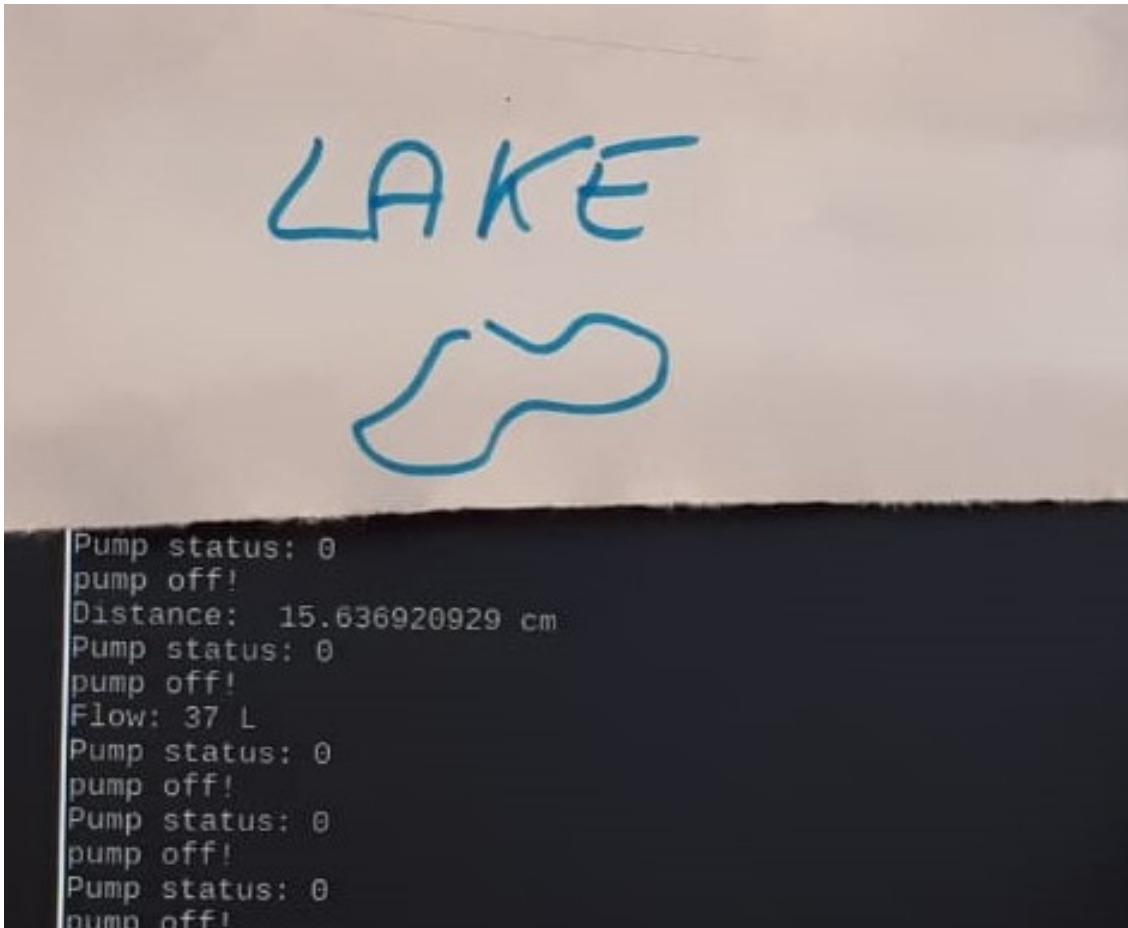


FIGURE 6.9: Output of Slave Lake

We also use 192.168.43.104, 192.168.43.105 to start TCP Modbus server at clean and house respectively. When all slaves are sending data to the master, then we execute master.py that will show all data received from slave using Modbus TPC/IP communication.

We also use 192.168.43.104, 192.168.43.105 to start TCP Modbus server at clean and house respectively. When all slave is sending data to the master, then we execute master.py that will show all data received from slave using Modbus TPC/IP communication.

6.4 CONCLUSION

When MCU wants to transmit, connects TE_EN of RS485 module to VCC, else connects to GND. In a software implementation, we write master.py for master Raspberry Pi to control all four slaves using Modbus library available in Python. We write slave_lake.py, slave_purifier.py, slave_clean.py and slave_house.py for Raspberry attached with lake, purifier, clean and house respectively. We have successfully implemented Modbus communication between sensors and actuators on both hardware and software level.

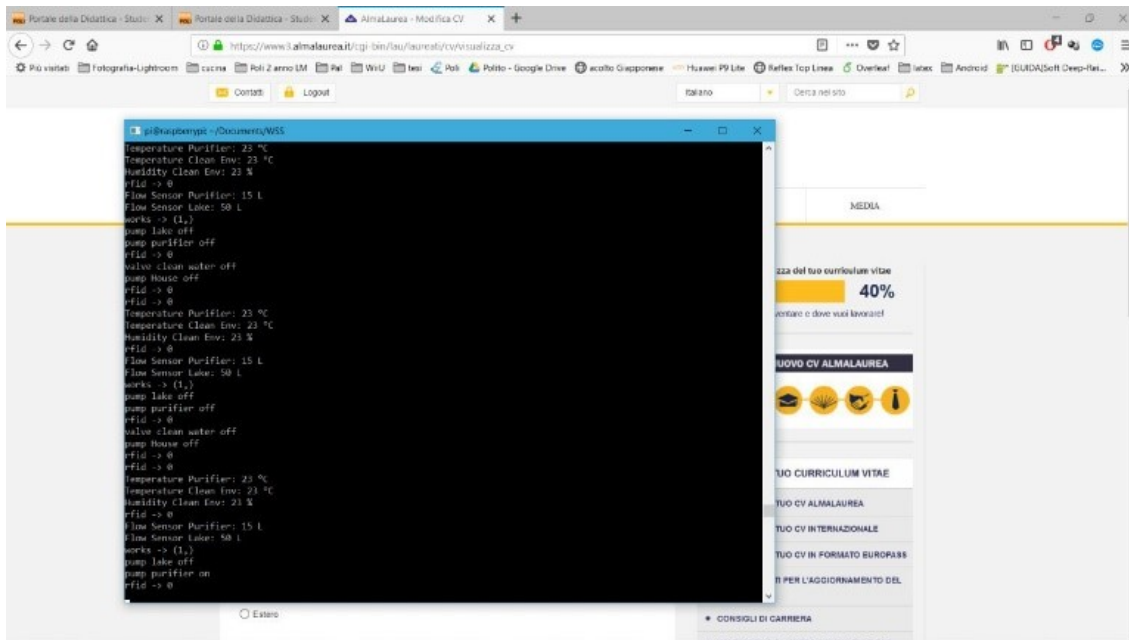


FIGURE 6.10: Output of Master.py

6.5 FUTURE SCOPE

In future, we shall use a vario variety of sensor like PH sensor, ORP Sensor, Pressure Sensor, Chlorine Sensor, and so on. There is also a wide scope to use different actuator Valve, Switch, and Chemical dosing device in our future WM-CPS. Modbus will replace with Bluetooth, Zigbee, GSM, and other communication standards in future.

Thousands have lived without love, not one without water.

W. H. Auden

7

Simulation of Water Management Cyber Physical System using EPANET before Design of Emulator

7.1 EPANET: Water Quality Simulator

EPANET performs simulation of real water management CPS, and it deals with the following nine phenomena:

- Energy and cost of pumping from the reservoir to Storage Tank or Pipe Network;
- The flow of water in the pipe;
- Head losses of friction, bends and fitting;
- Height of water in the tank;
- The Pressure of water at a junction;
- Chemical concentration in a network;
- Age of water in the system at any place;
- Blending water from a different source;
- Growth of disinfection by-products.

First five are related to phenomena of hydraulic simulation and the last four are water quality phenomena. GPM (Gallons Per Minute) is a default unit of flow and demand. One Gallon is equal to 3.78541 litres. Unit of length and elevation is feet. The diameter of the tank and pipe are taken as input in inches. Hazen-Williams is the default headloss formula. EPANET make use of three types of informational objects and six physical components.

7.1.1 Physical Peripherals

In analogy with electronic, links are wires and junction are ports. Out of six physical components, three are nodes, and three are links. For EPANET, the water management system is a collection of nodes connected with links. There are eight nodes and nine links illustrated in: Fig. 7.1.

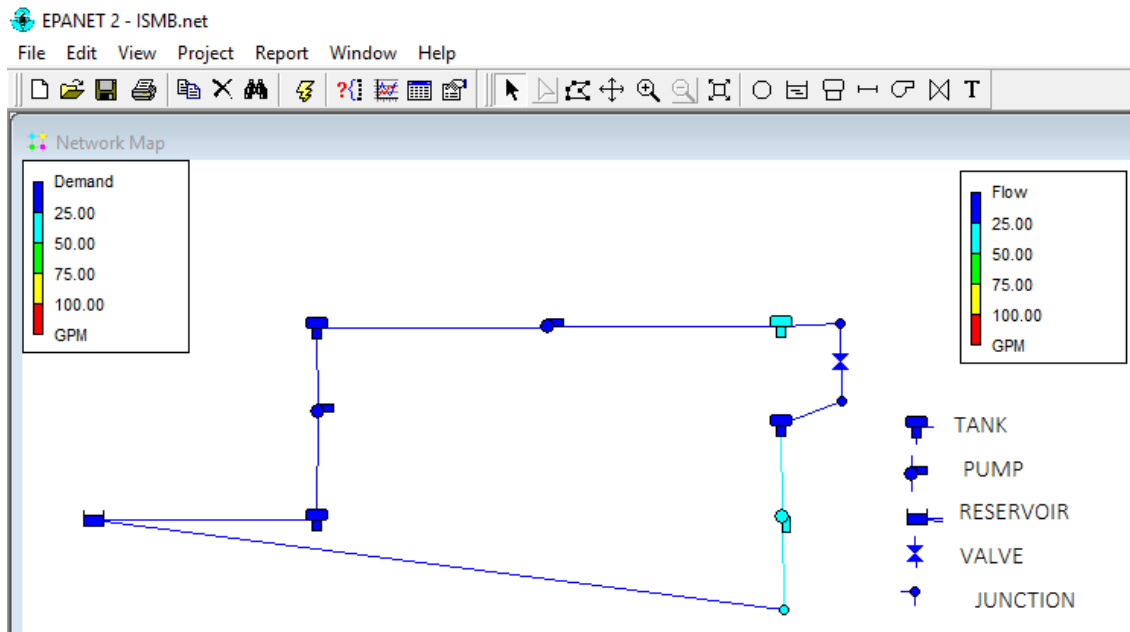


FIGURE 7.1: Physical Peripherals in a Typical Water Management System

Nodes in water management cyber-physical system are reservoirs, junctions, and tank. Initial water level, minimum water level, diameter, and maximum water level are unique properties of Tank and not applicable for either reservoir or junction. The elevation is only common properties of all three nodes. Elevation represents height from sea level. We have measured pressure in psi and demand in GPM at every node.

After simulation of Network Map shown in Fig. 7.1, we observe the following output on nodes of network map in terms of Demand, Head, Pressure and Quality (*ReportMenu* → *Table* → *NetworkNodes* → *OK*) as shown in Fig. 7.2.

Pipes, Pumps and Valves are links. Length and Diameter are unique properties of Pipe. These are not applicable for pump and valve. Similarly, Pump Curve is a unique property of Pump and Fixed Status is a unique property of Valve.

After simulation of Network Map shown in Fig. 7.1, we observe the following output on the link of network map in terms of Flow, Velocity, Unit Headloss and Friction Factors (*ReportMenu* → *Table* → *NetworkLinks* → *OK*) as shown in Fig. 7.3.

7.1.1.1 Junctions

In the other hands, junctions are places where pipes of one zone meet with a pipe of another zone. In other words, a place where water passes through the network is said to be a Junction. Elevation (sea level), demand and initial level are basic input of junctions.

At the end of the simulation, pressure outputs are computed at all time periods of simulation as shown as Fig. 7.4.

Network Table - Nodes at 4:00 Hrs

Node ID	Demand GPM	Head ft	Pressure psi	Quality
Junc Home	40.00	0.00	-0.65	0.00
Junc 2	0.00	11.87	5.14	0.00
Junc 3	0.00	12.50	5.42	0.00
Resvr 1	-4.19	0.00	0.00	0.00
Tank Purifier	-20.23	12.00	4.19	0.00
Tank Clean	30.23	11.87	3.59	0.00
Tank Lake	-0.04	0.00	0.00	0.00
Tank House	-45.77	12.50	4.96	0.00

FIGURE 7.2: Results of Node Outputs

Network Table - Links at 4:00 Hrs

Link ID	Flow GPM	Velocity fps	Unit Headloss ft/Kft	Friction Factor	Reaction Rate mg/L/d	Quality	Status
Pipe ResvToLakeTank	-0.04	0.00	0.00	0.000	0.00	0.00	Open
Pipe HouseToLake	-4.23	0.01	0.00	0.074	0.00	0.00	Open
Pipe CleanTankToValve	10.00	0.03	0.00	0.065	0.00	0.00	Open
Pipe ValveToHouseTank	-10.00	0.03	0.00	0.065	0.00	0.00	Open
Pump LtoP	0.00	0.00	0.00	0.000	0.00	0.00	Closed
Pump PtoC	20.23	0.00	0.13	0.000	0.00	0.00	Open
Pump HtoH	35.77	0.00	12.50	0.000	0.00	0.00	Open
Valve CtoH	10.00	0.03	0.63	0.000	0.00	0.00	Active

FIGURE 7.3: Results of Link Outputs

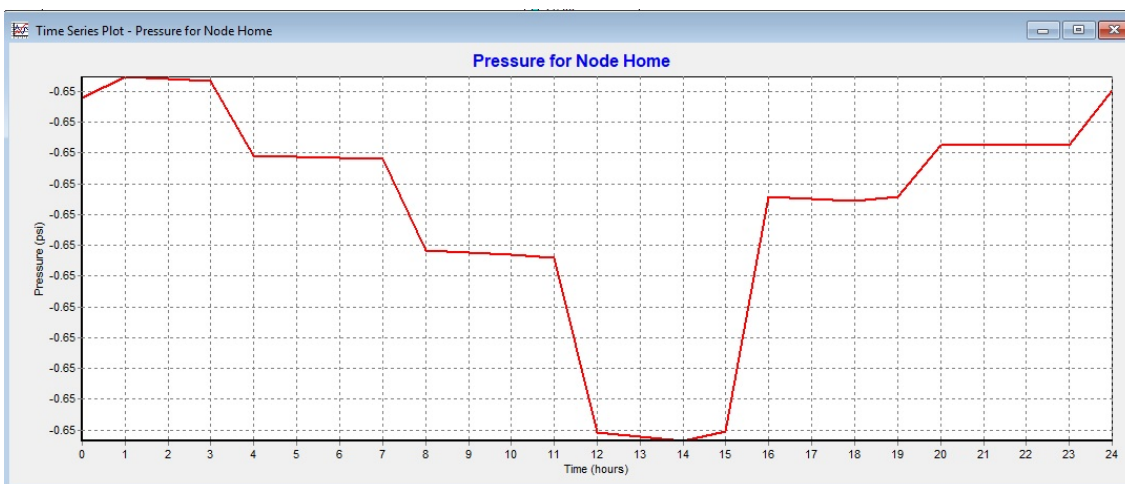


FIGURE 7.4: Plot of Pressure on Junction Home

7.1.1.2 Tanks

Tanks are nodes with storage capacity. The basic properties of Tank are Initial Level, Min Level, Max Level, Diameter, Elevation are shown in Table 7.1.

TABLE 7.1: Tank Properties in Water Distribution System (Fig. 7.1)

Tank	Initial Level	Min Level	Max Level	Diameter	Elevation
Lake	2	1	9.06	12.15	0
Purifier	2	1	3.54	7.795	2.329
Clean	2	1	3.54	7.795	3.576
House	2	1	5.52	12.15	1.5

Hydraulic head (water surface elevation) varies with time during simulation, we have taken 3.576 as elevation and 2 as



FIGURE 7.5: Plot of Head in Clean Tank

initial level, therefore hydraulic head start with 5.576 in Fig. 7.5.

7.1.1.3 Reservoirs

It models lake water, river water, sea water after desalination, groundwater source, lab water after synthesis of hydrogen and oxygen, and sink of rainwater harvesting. Following Fig. 7.6 represents demand of 24 hours during the simulation of our water management system.

7.1.1.4 Emitters

Emitters are a device attached with the junction in order to model the flow through nozzle or orifice that setting free water from the system to the atmosphere. It is also used to simulate leak. EPANET behave towards emitter as a property of a junction not as a water distribution network equipment.

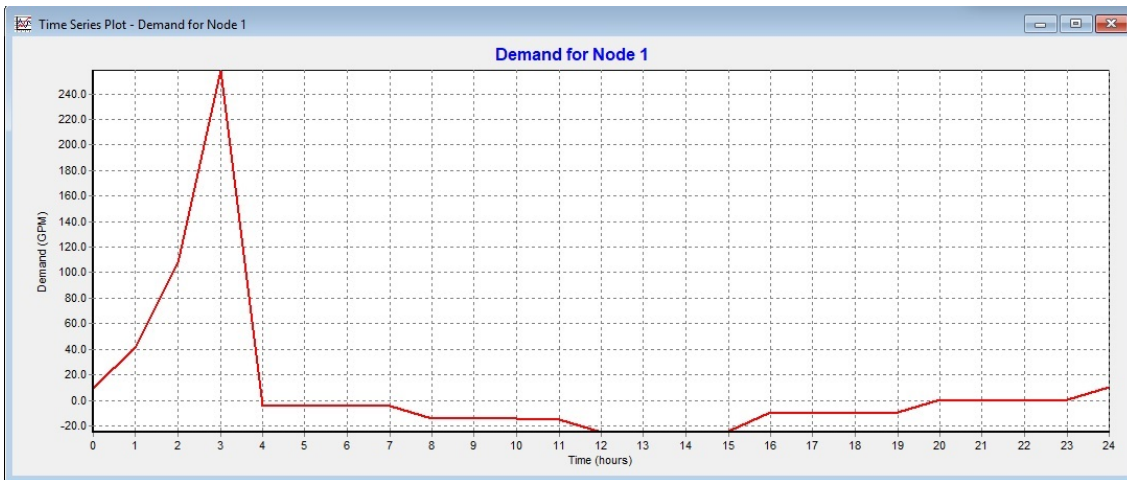


FIGURE 7.6: Plot of Demand in Reservoir: Node 1

7.1.1.5 Pipes

Pipes are linking that is made for moving water from one junction to another in a water distribution system. Simulator assumes that pipes are always full of water. The flow direction of water is from the end with a higher head to end with lower head. Unit Headloss of pipe six is shown in Fig. 7.7.

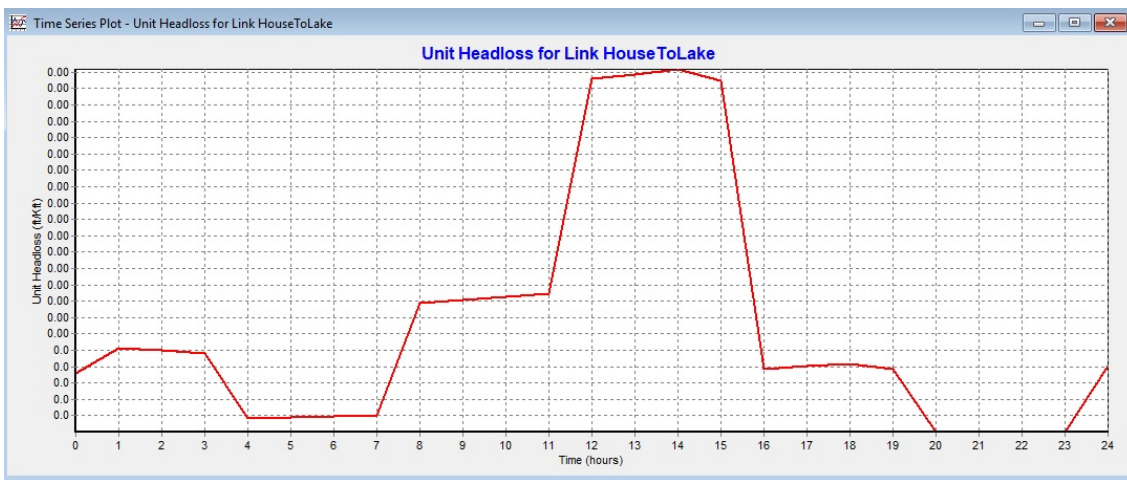


FIGURE 7.7: Unit Headloss in Pipe 6

7.1.1.6 Pumps

The pump gives a significant amount of energy to the fluid. This amount of energy is measured in either horsepower or kilowatts. The pump curve is a combination of flow and head.

The basic properties of the pump are head, Flow, and Pump Curve, as shown in Table 7.2.

The pump operations are limited to the range of its pump curve as shown in Fig. 7.8 and Table 7.2.

TABLE 7.2: Properties of Network Link:Pump

Pump	Flow	Head	Pump Curve
LtoP	3.963	5.41	Lake2Purifier
PtoC	2.642	4.265	Purifier2Clean
HtoR	2.642	4.265	House2Reservoir

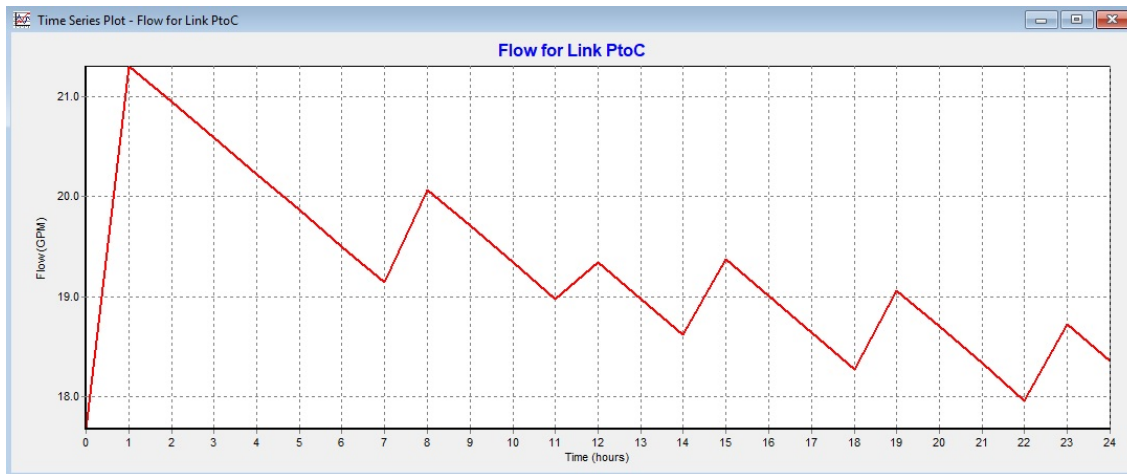


FIGURE 7.8: Flow For Pump in Water Distribution System

7.1.1.7 Minor Losses in Pump

It is also called local losses. It is only due to bends and fittings. Minor loss coefficients for a different type of fittings are shown in Table

TABLE 7.3: Minor Loss Coefficients for selected Fittings

FITTING	LOSS COEFFICIENT
45 degree Elbow	0.4
Completely Open Globe valve	10.0
Completely Open Angle valve	5.0
Completely Open Swing check valve	2.5
Completely Open Gate valve	0.2
Closed return bend	2.2
Exit	1.0
Long Elbow	0.6
Medium Elbow	0.8
Short Elbow	0.9
Standard tee-flow through run	0.6
Standard tee-flow through branch	1.8
Square Entrance	0.5

7.1.1.8 Valves

Properties of each valve containing:

- ID label
- ID of start node
- ID of end node
- Diameter, inches (mm)
- Valve type
- Valve setting
- Minor loss coefficient

Every Valve types have its own settings shown in Table 7.4.

TABLE 7.4: Valve Type & its specific Setting

Valve Type	Setting
PRV (pressure reducing valve)	Pressure, psi (m)
PSV (pressure sustaining valve)	Pressure, psi (m)
PBV (pressure breaker valve)	Pressure, psi (m)
FCV (flow control valve)	Flow (flow units)
TCV (throttle control valve)	Loss Coefficient
GPV (general purpose valve)	ID of head loss curve

Valves are links especially used to manage the pressure or flow at a particular place in the water distribution network. There are three states: partially open, fully open and closed.

Velocity output of fully open valves is shown in Fig. 7.9.

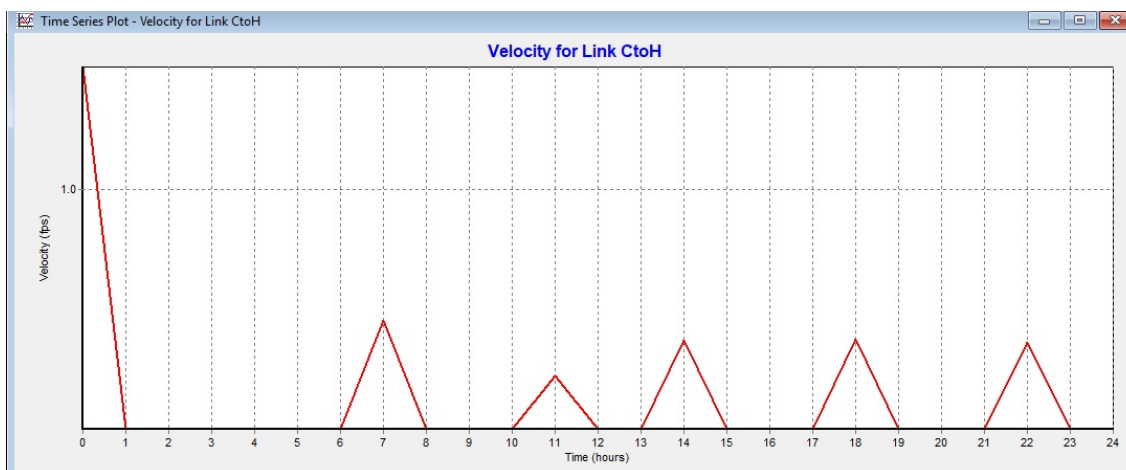


FIGURE 7.9: Velocity through Valve in Water Distribution System

7.1.2 Non-Physical Components

7.1.2.1 Curves

Curves contain data pairs. There are four different varieties of curves.

1. Efficiency Curve
2. Head Loss Curve
3. Pump Curve
4. Volume Curve

A combination of flow and head is known as a pump curve. The pump always operates inside the range of its particular pump curve. We have created three pump curve from the Data page. One Pump curve with the following data: Lake2Purifier as ID, 3.963 as Flow and 5.41 as Head illustrated in Fig. 7.10.

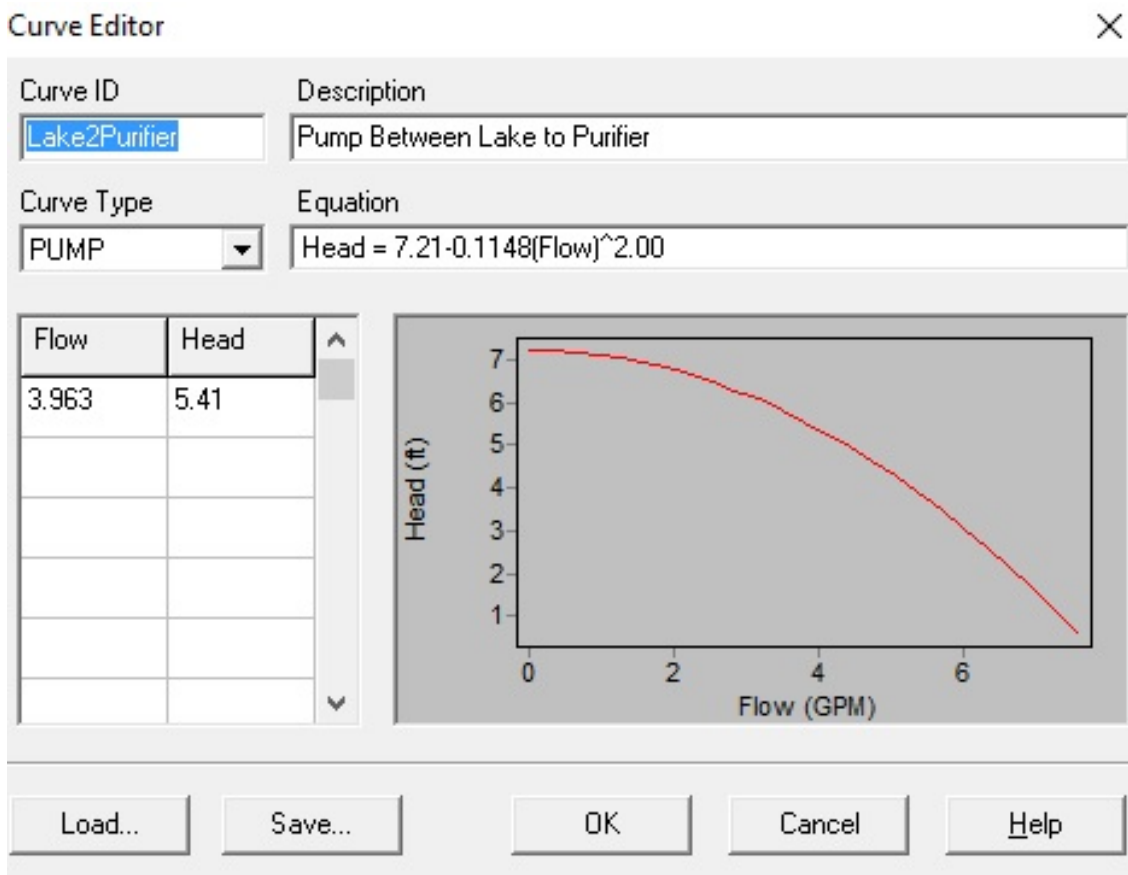


FIGURE 7.10: Curve Editor

7.1.2.2 Patterns

In case of using a pattern time step of 4 hours, demand changes at the six different times. For that, we have to access *Browser* → *Data* → *Options* → *Time* then add 24 Hours in Total Duration and 4 in Pattern Time Steps as shown in 7.11. Then we need to create Patterns using *Browser* → *Data* → *Pattern* → *AddButton*. We set value 0.5 is for 0-6

hours, 1.3 for 6-12 hours, 1.0 for 12-18 and 1.2 for 18-24 hours as shown in Fig. 7.12. This pattern is based on a common pattern of human life. People sleep from midnight to early morning, so the demand for water during this time interval is the lowest in the whole day i.e. 0.5. Since the duration of our simulation is 72 hours, the pattern will repeat after each 24-hour interval of time.

Property	Hrs:Min
Total Duration	24
Hydraulic Time Step	1:00
Quality Time Step	0:05
Pattern Time Step	4:00
Pattern Start Time	0:00
Reporting Time Step	1:00
Report Start Time	0:00
Clock Start Time	12 am
Statistic	None

FIGURE 7.11: Water Demand in Interval of 4 Hours

7.1.2.3 Controls

Controls are commands that decide how the network will work. Some Example of Simple Control is shown in Table 7.5.

TABLE 7.5: Some Example of Simple Control

Control Statement	Meaning
LINK ValveToHouseTank CLOSED IF NODE House ABOVE 5	Close Pipe ValveToHouseTank when the level in Tank (House) exceeds 5 ft.
LINK LtoP 1.5 AT TIME 16	Set the relative speed of the pump (Pump From Lake to Purifier) to 1.5 at 16 hours into the simulation
LINK PtoC CLOSED AT CLOCK-TIME 10 AM	Pump (Pump From Purifier to Clean) is frequently closed at 10 AM
LINK LtoP OPEN AT CLOCKTIME 8 PM	Pump (Pump From Lake to Purifier) is repeatedly opened at 8 PM

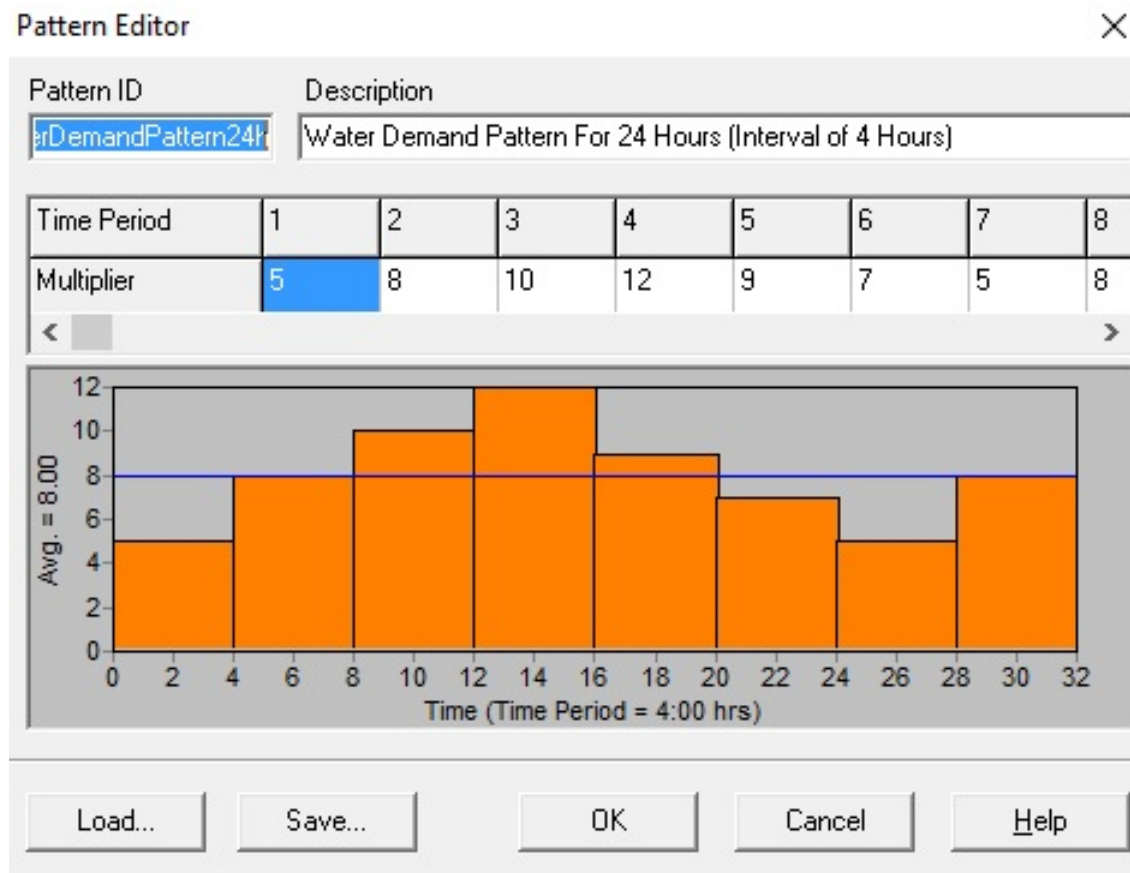


FIGURE 7.12: Pattern of Water Demand in Interval of 6 Hours

7.2 EPANET-MSX: Water Quality Simulator

Along with hydraulic and water quality simulation, it also models and simulates the dynamic reaction between chemical and/or biological species.

- Chemical-Chemical
- Chemical-Biological
- Biological-Biological

MSX refers to multi-species. That's why it is called the multi-species extension of EPANET.

7.3 Merlion: Water Quality Simulator

Merlion is also a water quality modelling framework. It differs with EPANET as it doesn't model bulk or wall reaction. But, it has extra features than EPANET like identify booster station locations, contaminant source injection locations and manual grab sample locations. It is faster for the multi-scenario simulation that's why it is more memory intensive than EPANET.

7.4 WST: Water Security Toolkit

The term WST is an abbreviation of Water Security Toolkit. The WST is an extension of the TEVA-SPOT. WST helps in the assessment of various cause and effects in order to choose the most optimal plan of action. It supports optimization methodologies for hydraulic and water quality modelling. Some following optimization methodologies are:

1. Sensor Placement Optimization: find out locations of sensor that help to detect contamination more accurately;
2. Contaminant Transport Simulation: insertion of contamination at locations in the distribution system and its flow throughout the network;
3. Impact Assessment: assess possible economical and physical loss of water consumer;
4. Hydrant Flushing: use hydrants to eliminate contaminated water from a cyber physical system;
5. Booster Placement: inject decontamination agents on specific junctions in the water network to destroy contaminants;
6. Grab Sampling: taking a grab sample from a random junction in the water network to verify contamination or cleanup;
7. Source Identification: isolating contaminated areas of the network by closing the valve;
8. Visualization.

WST replicates sensor placement optimization, contaminant transport simulation and impact assessment from TEVA-SPOT toolkit and adds several new features like hydrant flushing, booster placement, grab sampling and source identification.

We are using EPANET input (INP) file i.e. Net3.inp that defines the water distribution network model as shown in Fig. 7.13. We first copy the input file into the current working directory and generate both results and log file of Fig.7.13 by following command lines and WST configuration file in the YAML format (*verify_wst.yml*):

```
C : \WST\wst - 1.3\bin > copy C : \WST\wst - 1.3\examples\Net3\Net3.inp
C : \WST\wst - 1.3\bin > wst tevasim verify_wst.yml
```

The sub-command tevasim generate two files:

Results File: *Net3tevasim_output.yml*
Log File: *Net3tevasim_output.log*

We generate corresponding results file, log file, and HTML file of 7.13 by following command lines and WST configuration file in the YAML format (*verify_wst.yml*):

```
C : \WST\wst - 1.3\bin > wst visualisation verify_wst.yml
```

The sub-command visualisation generate three files:

Results File: *Net3visualization_output.yml*
Log File: *Net3visualization_output.log*
Visualization File: *Net3visualization.html*

In order to specify sensor location and booster location in a water distribution network, we are using *booster_mip_ex1.yml* configuration file as following:

```
C : \WST\wst - 1.3\bin > wst booster_mip C : \WST\wst - 1.3\examples\booster_mip_ex1.yml
```

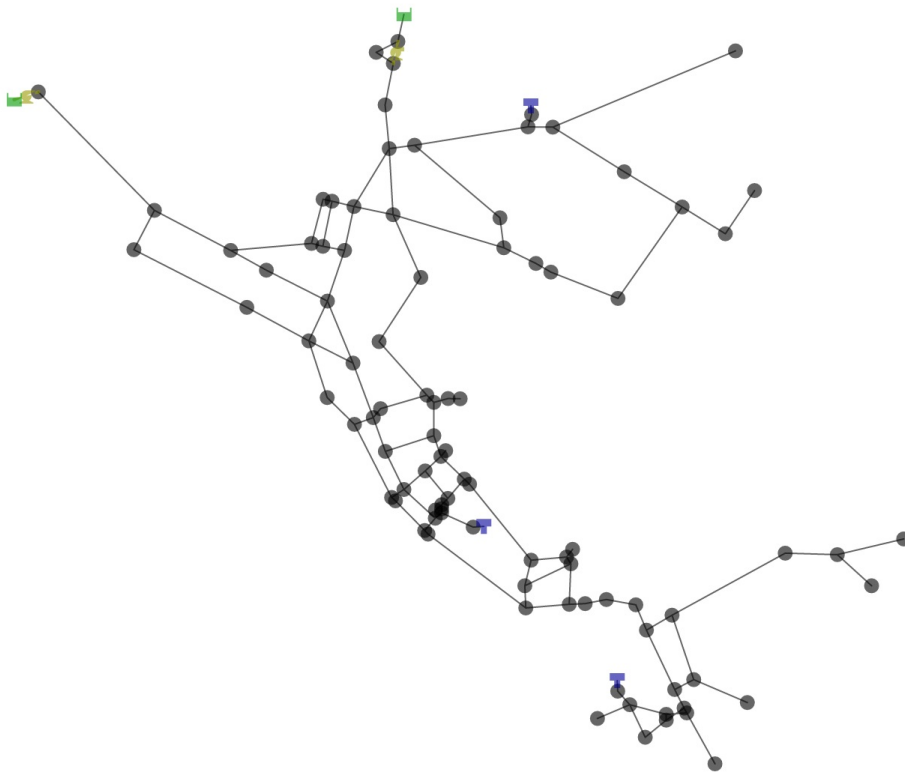


FIGURE 7.13: Layout of Sample Water Distribution Network

7.4.1 Configuration File

The *booster_mip* template configuration file is as following:


```

# booster_mip configuration template
network:
epanet file: Net3.inp          # EPANET network file name
scenario:
location: null                 # Injection location: ALL, NZD or EPANET ID
type: null                     # Injection type: MASS, CONCEN, FLOWPACED or SET-POINT
strength: null                 # Injection strength [mg/min or mg/L based on type]
species: null                  # Injection species, required for EPANET-MSX
start time: null               # Injection start time [min]
end time: null                 # Injection end time [min]
tsg file: Net3.tsg             # TSG file name, overrides injection parameters above
tsi file: null                 # TSI file name, override TSG file
msx file: null                 # Multi-Species extension file name
msx species: null              # MSZ species to save
merlion: false                 # Use Merlion as WQ simulator, true or false
booster mip
detection: [111, 127, 179]     # Sensor locations to detect contamination scenarios
model type: NEUTRAL            # Booster model type: NEUTRAL or LIMIT
model format: PYOMO            # Booster Optimization model:APML or PYOMO
stoichiometric ratio: [0.0]    # Stoichiometric ratio [decon/toxin], in LIMIT model
objective: MC                   # Objective to Minimize
toxin decay coefficient: 0      # Toxin decay coefficient: None, INP or number
decon decay coefficient: 0      # Decontaminant decay coefficient: None, INP or number
feasible modes: ALL            # Feasible booster nodes
infeasible modes: NONE         # Infeasible booster nodes
max boosters: [2]              # Maximum number of booster stations
type: FLOWPACED                # Booster source type: MASS or FLOWPACED
strength: 4.0                  # Booster source strength [mg/min or mg/L depending on type]
response time:0.0              # Time [min] between detection and booster injection
duration: 600.0                # Time [min] for booster injection
evaluate: false                 # Evaluate booster placement: true or false, default=false
solver:
type: glpk                      # Solver type
options:                          # A dictionary of solver options
threads: 1                       # Number of concurrent threads or function evaluations
logfile: null                     # Redirects solver output to a logfile
verbose: 0                        # Solver Verbosity level
initial points: []
configure:
output prefix: Net3             # Output file prefix
debug: 0                         # Debugging level, default = 0
boostersim:
eventDetection:
boosterimpact:

```

A stoichiometric ratio of 0.01 mg/CFU specifies that 0.01 mg of a disinfectant is needed to activate 1 CFU of contaminant.

The sub-command *booster_mip* generates visualisation configuration file along with results and log file. Results File: *Net3booster_mip_output_5.yml*

Log File: *Net3booster_mip_output.log*

Visualization File: *Net3booster_mip_output_5_vis.yml*

Then we generate all five possible visualisation using sub-command *visualization* as following:

```
C : \WST\wst-1.3\bin >wst visualization C : \WST\wst-1.3\bin\booster_mip_ex1\Net3booster_mip_output_5_vis.yml
```

Initially, WST places two boosters in a water distribution network in order to neutralise contaminant, as shown in Fig. 7.14.

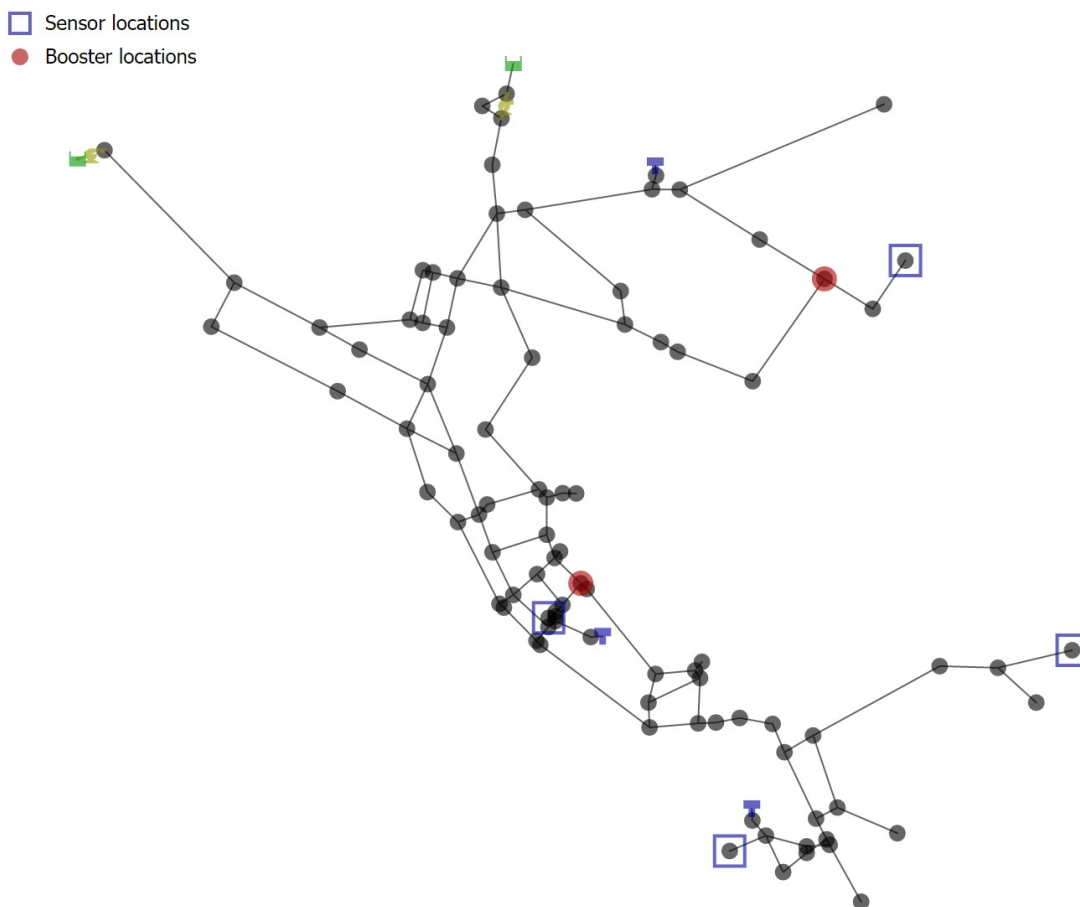


FIGURE 7.14: Location of Two Boosters in Water Distribution Network

Later, WST increase number of boosters from two to four in water distribution network in order to neutralize contaminant as shown in Fig. 7.14, Fig. 7.15, Fig. 7.16, Fig. 7.17 and Fig. 7.18.

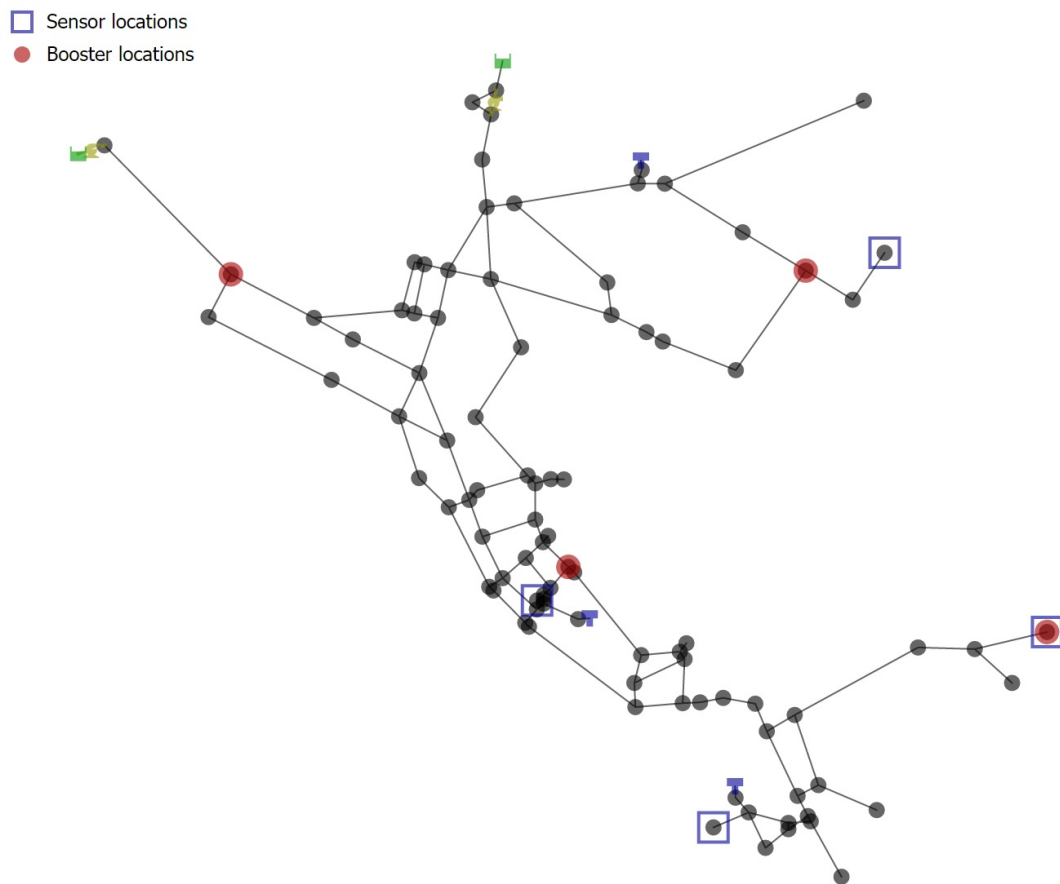


FIGURE 7.15: Location of Four Boosters in Water Distribution Network

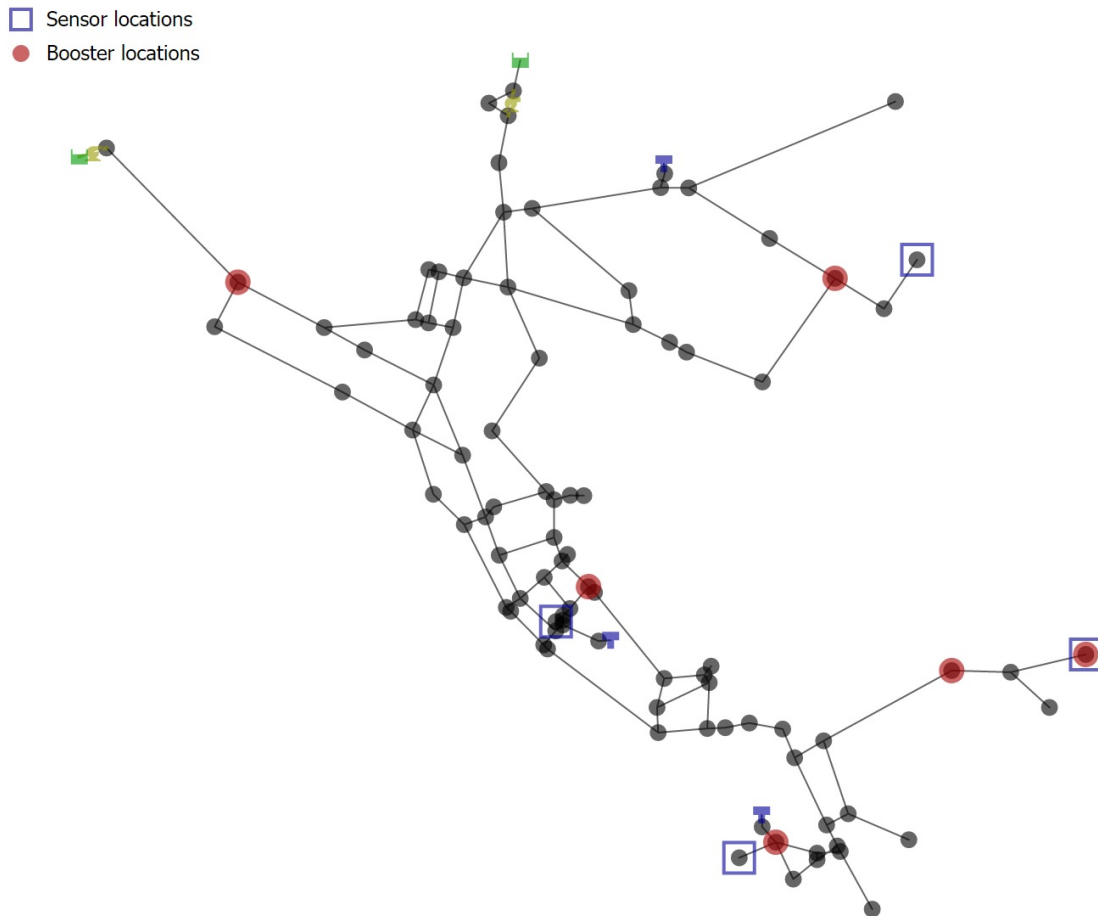


FIGURE 7.16: Location of Six Boosters in Water Distribution Network

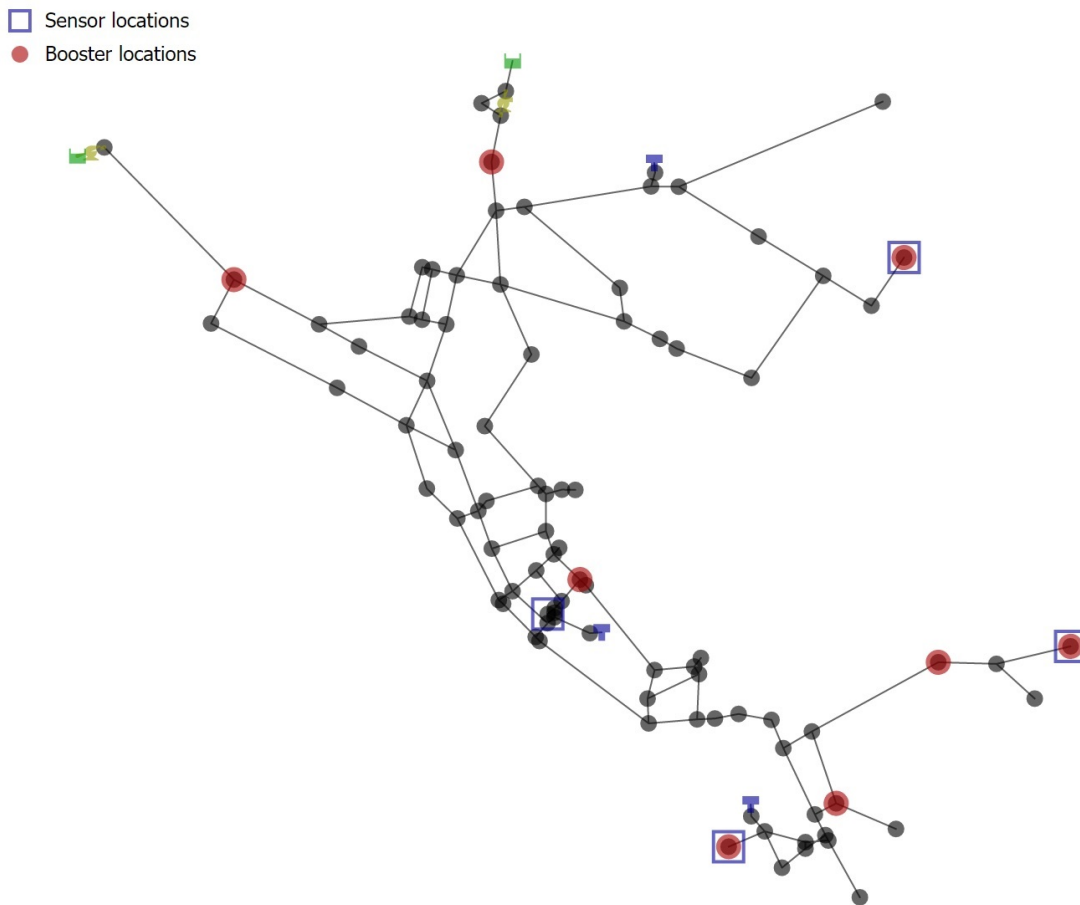


FIGURE 7.17: Location of Eight Boosters in Water Distribution Network

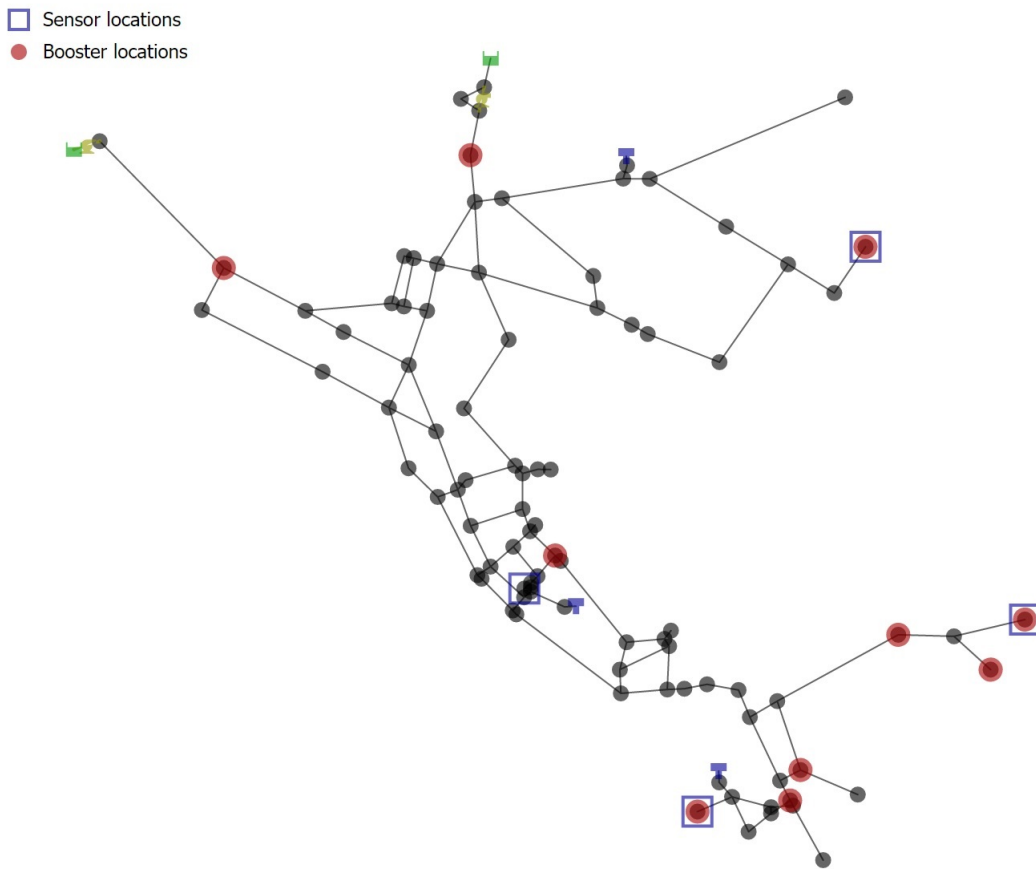


FIGURE 7.18: Location of Ten Boosters in Water Distribution Network

We forget that the water cycle and the life cycle are one.

Jacques Cousteau

8

Design of Emulator for Water Management Cyber Physical System: using Raspberry Pi without SECUBE

In the water management cyber physical system, four tanks and five Raspberry pi 3 controllers are used, as shown in Fig. 8.1. The first tank is in Lake. The second tank is in Water Purifier station. The third tank is called Clean Water Tank. The fourth tank is known as the house tank. One Raspberry Pi is acting as a master controller and rest four Raspberry Pi are slave controllers. Each tank has its controller called a slave controller. Master is controlling all four slave controller. In order to control the pump of 220 V and also valve of 220V, 5 V relay is used between pump and Raspberry Pi 3 controller. 5V relay is also used between Valve and Raspberry Pi 3. Always on Relay work perfectly fine with 5 volt VCC pin Number 2 and 4 of Raspberry Pi 3. Always off Relay work perfectly fine with 0 volt GND pin Number 6 and 9 of Raspberry Pi 3 (RPi3). There are one temperature sensor, two Flow sensors, and four ultrasonic sensors used to measure temperature, the flow of water and the level of water in Tank, as shown in Fig. 8.1. Ultrasonic sensors usually installed on top of every tank in order to measure the level of water in that tank.

The temperature sensor is only attached with Purifier tank that measures the temperature of water. Flow sensors are associated with Pump of Lake and Purifier. Flow sensors measure how many litres of water passes from Lake and Purifier tank. This flow sensor is typically open drain. It needs to pull to the ground to signal a pulse and float high to a voltage.

The final implementation of the architecture discussed in Fig. 8.1 is shown in Fig. 8.2.

8.1 Connect with Eduroam/Wi-Fi of ISMB

We add following line of codes to `/etc/wpa_supplicant/wpa_supplicant.conf`

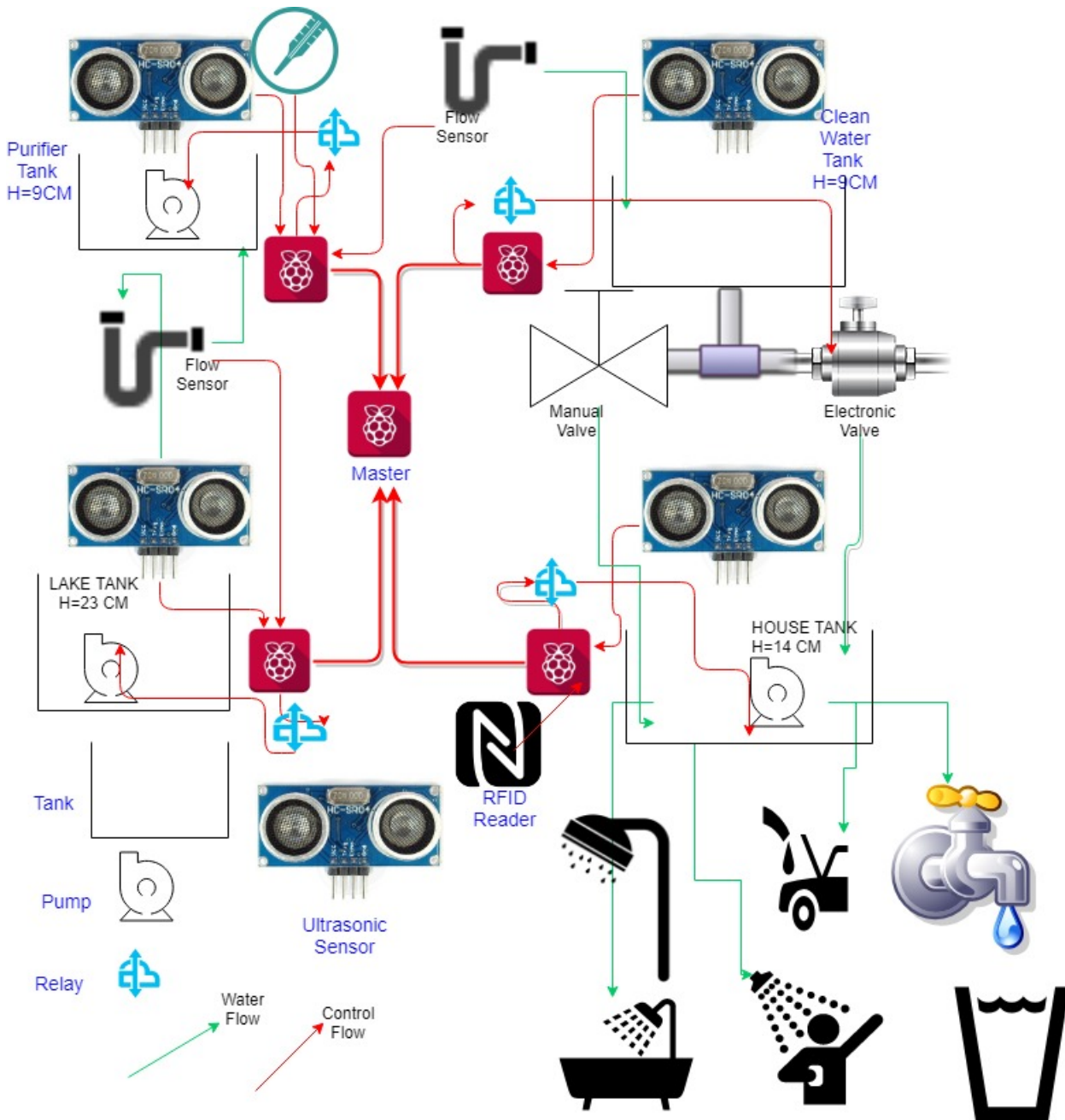


FIGURE 8.1: Architecture of Emulator of Water Management Cyber Physical System

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
country=GB

network={
ssid="eduroam"
psk="bogio61"
key_mgmt=WPA-PSK
}
```

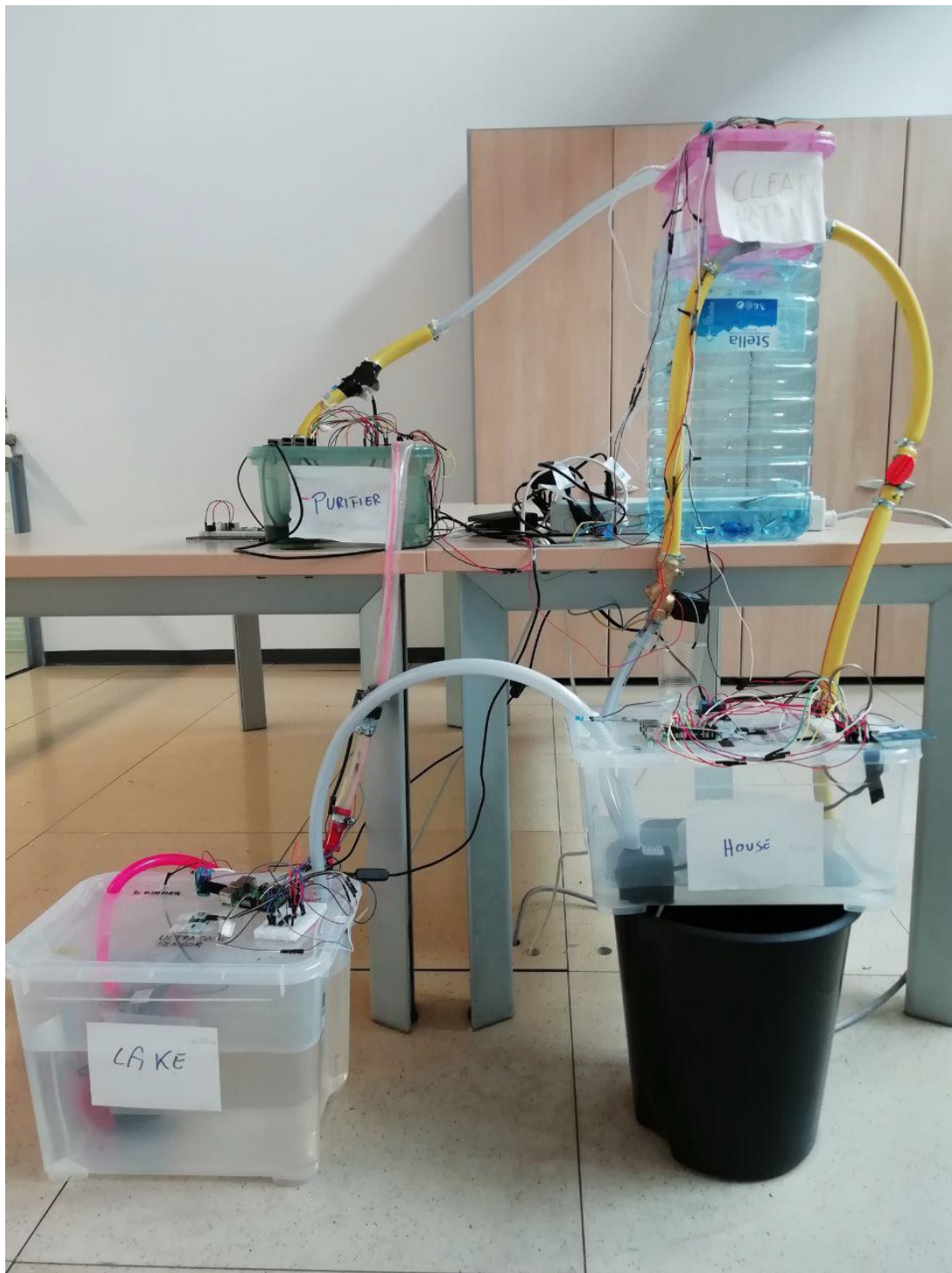



FIGURE 8.2: Implemented Emulator of Water Management Cyber Physical System

OR

```
network={  
ssid="eduroam"  
scan_ssid = 1  
key-mgmt=WPA-EAP  
eap=PEAP
```

```
identity="bishwajeet.pandey@gssi.infn.it"
password="*****"
phase1="peaplabel=0"
phase2="auth=MSCHAPV2"
}
```

wpa_supplicant is a text file that is composed of all accepted networks, securing policies, and Pre Shared Keys (PSK). WPA is an abbreviated form of WiFi Protected Access. EAP stands for Extensible Authentication Protocol. SSID is the Service Set Identifier (Network Name). In order to force a new label, we set peaplabel to 1. Phase 1 is outer authentication, i.e. TLS tunnel. Phase 2 is an inner authentication, i.e. TLS tunnel. Whereas TLS is transport layer security. PEAP is also known as Protected Extensible Authentication Protocol.

8.1.1 Assign Static IP to Master and Slaves

sudo nano /etc/dhcpd.conf.

Type the following lines on the top of the file:

```
# interface wlan0
static ip_address=192.168.43.100/24
static routers=192.168.43.1
static domain_name_servers=192.168.43.1 8.8.8.8
```

Type sudo reboot.

For assigning of Static IP to purifier we type the following command

sudo nano /etc/dhcpd.conf.

Type the following lines on the top of the file:

```
# Example static IP configuration:
#interface eth0
static ip_address=192.168.43.103/24
static routers=192.168.43.1
static domain_name_servers=192.168.43.1 8.8.8.8
```

Similarly, we assign 192.168.43.102, 192.168.43.104 and 192.168.43.105 to Lake, Clean and House.

8.1.2 Install PyMySQL

\$sudo apt-get install pymysql command is used to install pymysql.

8.2 Flow Sensor

Water flow sensor consists of a plastic valve body, a water rotor, and a hall-effect sensor as shown in Fig. 8.3. This flow sensor calculates the rate of flow of liquid (here water) in a pipe of our WMCPS. This flow sensor situated inside the water pipe, and take help of a pinwheel sensor to compute how much water has passed through it. A little magnet is inbuilt into the pinwheel sensor. There is a Hall Effect magnetic sensor on the other side of the plastic tube. This Hall Effect



FIGURE 8.3: YFS201 Water Flow Sensor

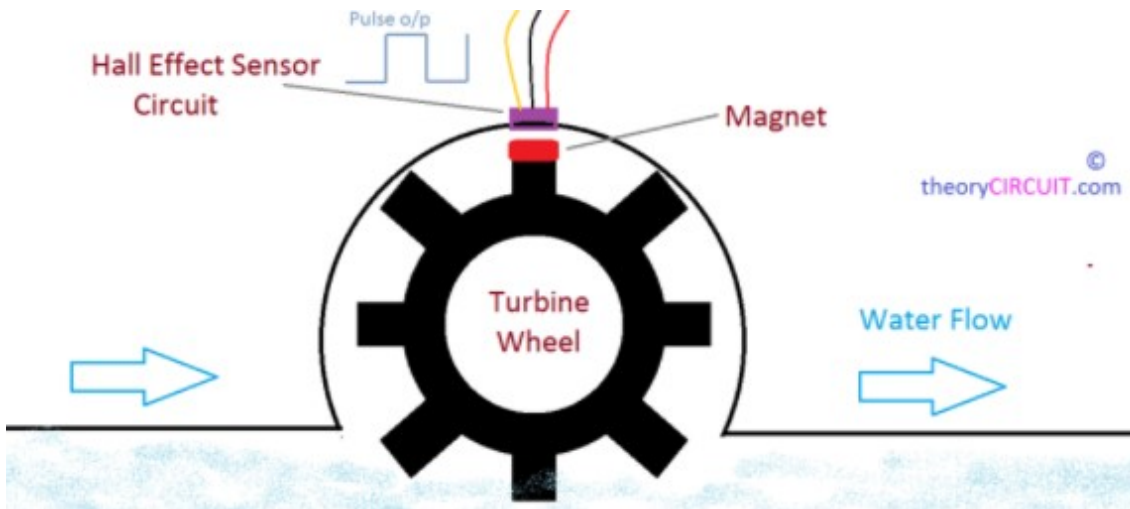


FIGURE 8.4: Working of YF-S201 sensor [79]

magnetic sensor measures the number of spins the pinwheel has made through the plastic wall. By this way, a sensor is always dry. Fluid movement is directly proportional to the number of pulses. Pulses are the output of the sensor. Each pulse is approximately 2.25 millilitres. Range of Flow Rate is 1-30 Liters/Minute. The range of Temperature is -25 to 80 °C. Operating Humidity Range is 35%-80% RH. Frequency (Hz) is 7.5 Flow rate (L/min). 450 Pulses are one Liter. We have associated Flow Sensor with Lake and Purifier tank. This flow sensor is typically open drain. It needs to pull the ground to signal a pulse and float high to a voltage. The working of YF-S201 sensor is as shown in Fig. 8.4.

If we do not have the input pin connected to anything, it will float. To get round of this, we use a pull-up or a pull-down resistor. In this way, the default value of the input can be set. It is possible to have pull-up/pull-down resistors in hardware and using the software. In hardware, one 10K resistor between the input channel and 3.3V (pull-up) and 0V (pull-down) is commonly used.

```
GPIO.setup(FLOW_SENSOR, GPIO.IN, pull_up_down=GPIO.PUD_UP)
```

Water meter pulse outputs are typically open drain. This means they are pulled to ground to signal a pulse and float high to an external voltage.

```
GPIO.add_event_detect(FLOW_SENSOR, GPIO.FALLING, callback=countPulse)
```

8.3 Pump and Relay

Relay is used to control high voltage electronic devices. In our work, Pump is high voltage electronic devices. Therefore, we are using one relay with each Pump.

5V Relay Terminals and Pins



FIGURE 8.5: Relay Terminals and Pins

In Fig. 8.5, relay has rating of 10A @ 250 and 125 V AC and 10A @ 30 and 28 V DC. The high voltage output connector has 3 pins, the middle one is the common pin and as we can see from the markings one of the two other pins is for normally open connection and the other one for a normally closed connection. On the other side of the relay i.e. low voltage input connector, we have three pins: a Ground, a VCC pin for powering the module and one input pin called Signal. When a relay contact is normally open (NO), there is an open contact when no current is given on Signal pin of the relay. When a relay contact is Normally Closed (NC), there is a closed contact and there is no current on Signal input pin of the relay. In either case, applying an electrical current to the Signal pin will change their state.

In Fig. 8.6, relay module has four relays. We zoom out high voltage output connector of the fourth relay and show L shaped standard lining for Normally open. Common is slash shaped and touched with L shaped standard lining of Normally closed and make a close area. We have connected one relay to pump as shown in Fig. 8.7 and have written a python script that receives value i.e. command from the website. If we click on "Turn ON", 1 will store in the table called water flow on the web server. If we click on "Turn OFF", 0 will store in the table called water flow on the web server. We are able to read the value of command using requests.get command of Python.

```
r = requests.get('http://cps4wm.info/getData.php')
```

If requests.get is "1" then the relay will on because we are using a normally open configuration of a relay as shown in Fig. 8.7 and Pump will start. And, if requests.get is "0" then the relay will remain in a normally open state and the pump will off. The PHP script is written to select current value from MySQL waterflow table using MySQL query of "select * from waterflow". The PHP script is also written to update value field of waterflow table based on click on Turn ON/OFF button using MYSQL query of "update waterflow set value=\$val" [17].

8.4 Waterproof Temperature Sensor

It measures the temperature of water flow in the water distribution system from either tank to users or reservoirs to tanks. DS18B20 sensor as shown in Fig. 8.8 measures the temperature of water flowing in a water distribution system from either tank to users or reservoirs to tanks. Its colour is predominantly black. The material used to build this sensor is durable stainless steel. Cable length is 100 cm. Stainless steel tube size is about 6 * 50mm (diameter * length). Range of power supply is 3.0 V-5.5V. It is an original chip DS18B20, of three wire temperature sensor. There are three Cables: Red (VCC), yellow (information), black (GND) [58].

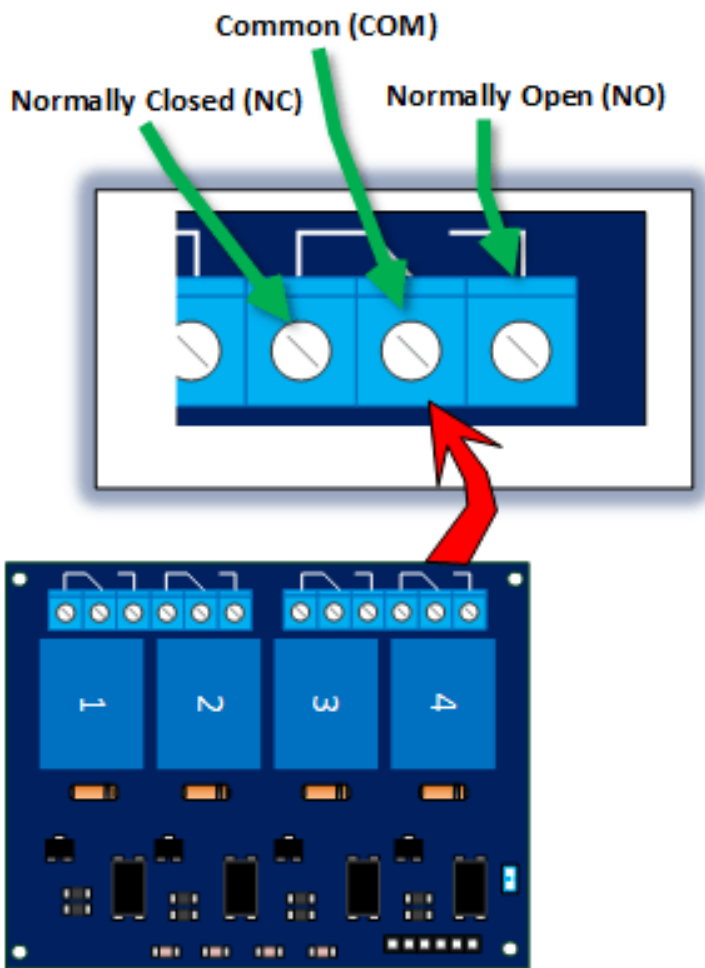


FIGURE 8.6: High Voltage Output Connector of Relay Module

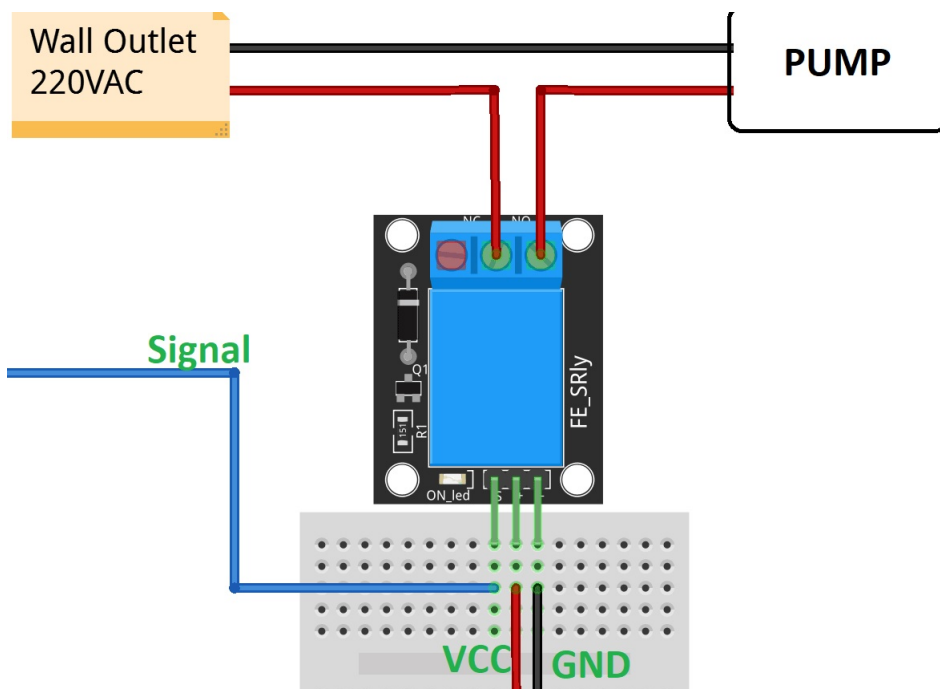


FIGURE 8.7: Relay is used to control High Voltage Pump



FIGURE 8.8: Waterproof Temperature Sensor

If one wire sensor is not working then we edit the config file. `$ sudo nano /boot/config.txt` We add the following line in `config.txt`

```
dtoverlay=w1-gpio, gpiopin=4
```

```
$sudo reboot
```

We connect 7th pin of Raspberry board to the yellow wire of sensor. There is $1\text{ k}\Omega$ resistor between the 7th pin of Board and 3.3V VCC. After that, we load one wire communication device kernel modules:

```
sudo modprobe w1-gpio
```

```
sudo modprobe w1-therm
```

We point to address of the temperature sensor

```
$ cd /sys/bus/w1/devices/28*
```

```
$ cat w1_slave
```

```
a8 01 4b 46 7f ff 0c 10 e5: crc=e5 YES
```

```
a8 01 4b 46 7f ff 0c 10 e5: t=26500
```

The response will either have YES or NO at the end of the first line. If it is YES then the temperature will be at the end of the second line in 1/1000 days.

8.4.1 Python Program to Update Water Temperature on MySQL Server

We save this python program with the name of waterTemp.py available on Chapter A.1, and we call this python file in the code execute on Purifier as following:

```
from waterTemp import read_temp
```

We create a table called Temperature in MySQL.

```
Create TABLE Temperature (timestamp datetime NOT NULL, purifier int(11) NULL);
```

Creation of other tables discussed in details in the next SQL section.

8.5 Environment Temperature and Humidity Sensor



FIGURE 8.9: Environment Temperature Sensor

DHT11 sensor as shown in Fig. 8.9 requires 3.5V to 5.5V DC for operation. It measures outside temperature in the range of 0 to 50°C using thermistor. It measures humidity in the range of 20-95% using a capacitive humidity sensor. It gives result after at least every 2 seconds. With the data received from the sensor, we are able to study the effect of humidity and outside temperature on corrosion of pipe, actuators and sensors [58].

8.6 Level Sensor

The HC-SR04 Ultrasonic sensor as shown in Fig. 8.10 has four pins: ground (GND), Echo Pulse output (ECHO), Trigger Pulse input (TRIG), and 5V supply voltage (VCC). This sensor consists of one or more ultrasonic transmitters (mainly speakers), a receiver, and a control circuit. The emitter emits a high-frequency ultrasonic sound, which bounces off anything that has a higher density than air. The receiver of the sensor detects ultrasonic noise reflected by water in front of the sensor. That return signal is then processed by the control circuit to calculate the time difference between the signal being transmitted and received. This half of time difference can subsequently be multiplied with speed of sound, i.e., 340 m/s, to calculate the distance in m between the sensor and the reflecting object.

Ultrasonic Sensor senses the level of water if installed on top of the water tank as shown in Fig. 8.11. We shall subtract the distance of water from the height of the water tank and find the level of water in the tank.

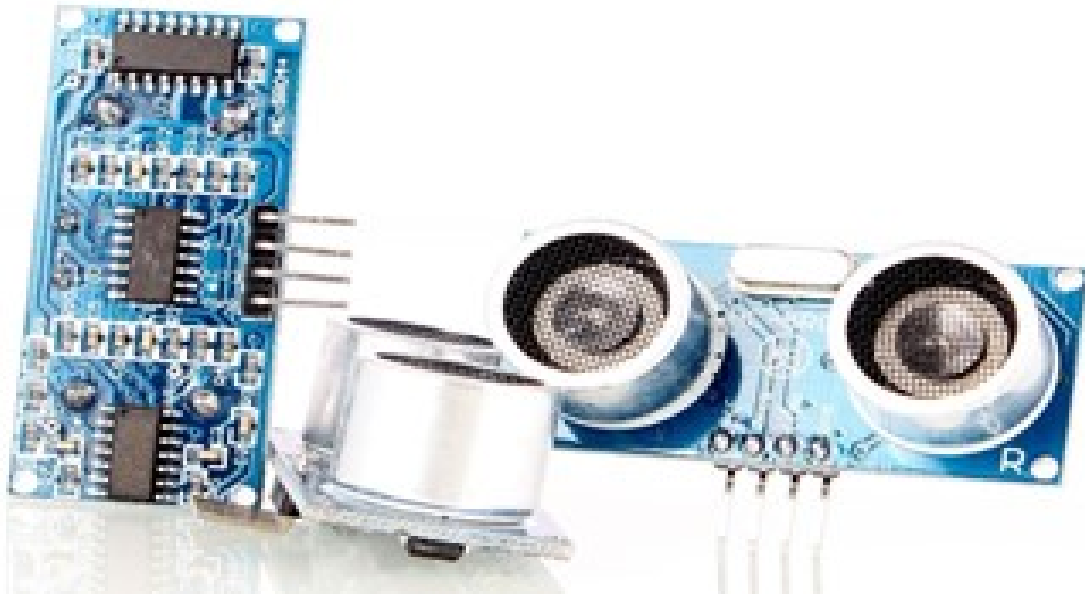


FIGURE 8.10: The HC-SR04 Ultrasonic Sensor

8.7 Slave Controller at Lake Tank

The "lake" tank is the first and the biggest tank of this emulator; in fact, its height is of 23 cm. There is a Raspberry Pi board on top of the tank that receives data from sensors and sends these information to the master.

In Figure 8.12 of lake tank, there is a flow sensor and an ultrasonic sensor and one pump that moves water to the "purifier" tank. The commands to ON are sent to the pump from the master only if the purifier tank has a water level below 4 cm. The flow sensor is programmed to compute the number of litres of water that flow and stores this value and sends it to the master. The counter is reset every time the emulator restarts. The code which test sensors and actuators attached with Lake Tank is available on Chapter A.2.

8.8 Slave Controller at Purifier Tank

The "purifier" is the second tank in this emulator. This tank is smaller than the lake tank, in fact, its height is of only 9 cm.

Raspberry Pi, associated with purifier tank, determine the level of water in the tank using the ultrasonic sensor, compute temperature and flow of water with the help of waterproof temperature sensor and the flow sensor (Fig. 8.13). The controller commands the pump to flows water in the next tank, the "clean water" tank, has at least 4 cm of free space. The python program that executes at Purifier tank is available on Chapter A.3.

8.9 Slave Controller at Clean Tank

The "clean water tank" is the third tank in this emulator. This tank is as small as the purifier tank. Both tanks have a similar height of only 9 cm. But, it is different from the "purifier" tank because this tank is placed over that support that is 40 cm high. The difference in height has been modelled in order to better create a system similar to a real one

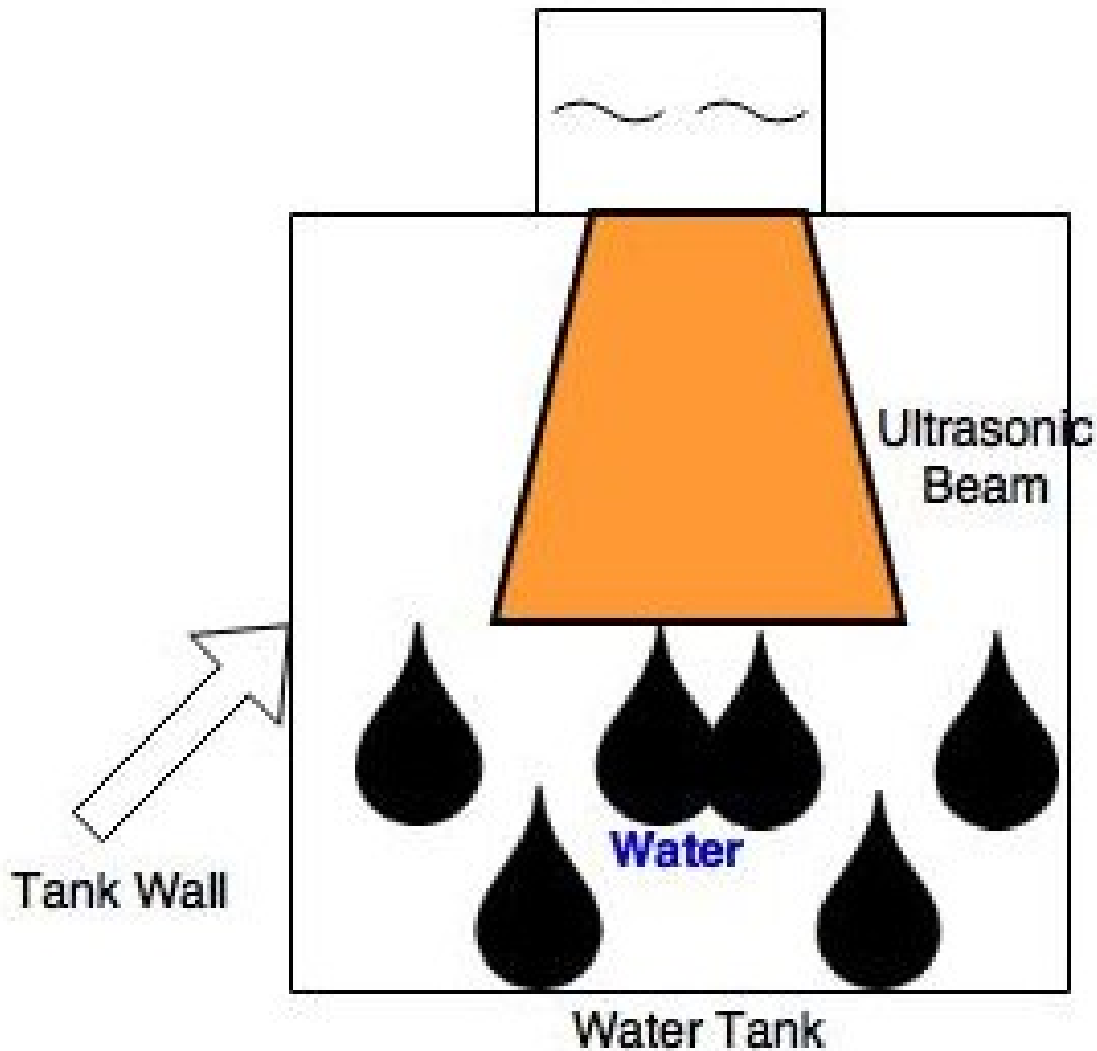


FIGURE 8.11: Ultrasonic Sensor for Water Level Measurement [58]

where there is gravity based water distribution. The Raspberry pi acting as slave controller at clean water tank receive data from the environmental temperature and humidity sensor and the ultrasonic sensor for the water level (Fig 8.14). This tank is without pump; due to the difference of height where this tank and the next one are placed, it is possible to allow the flowing of water only with gravity effect. Two pipes connect this tank to the next tank: the first one is fitted with a manual valve while the second one is fitted with an electronic valve. The python program written to perform these tasks is available on Chapter A.4.

8.10 Slave Controller at House Tank

It is the last tank of the emulator. It is bigger than the "purifier" and "clean water" tanks; in fact, its height is of 14 cm. But, it is smaller than the lake tank. Connected to the Raspberry Pi of this tank, there are also an ultrasonic sensor and a pump (Fig. 8.15). The code which executes on this Raspberry Pi is available on Chapter A.5.

The pump allows the re-usage of water by draining it to the first "lake tank".

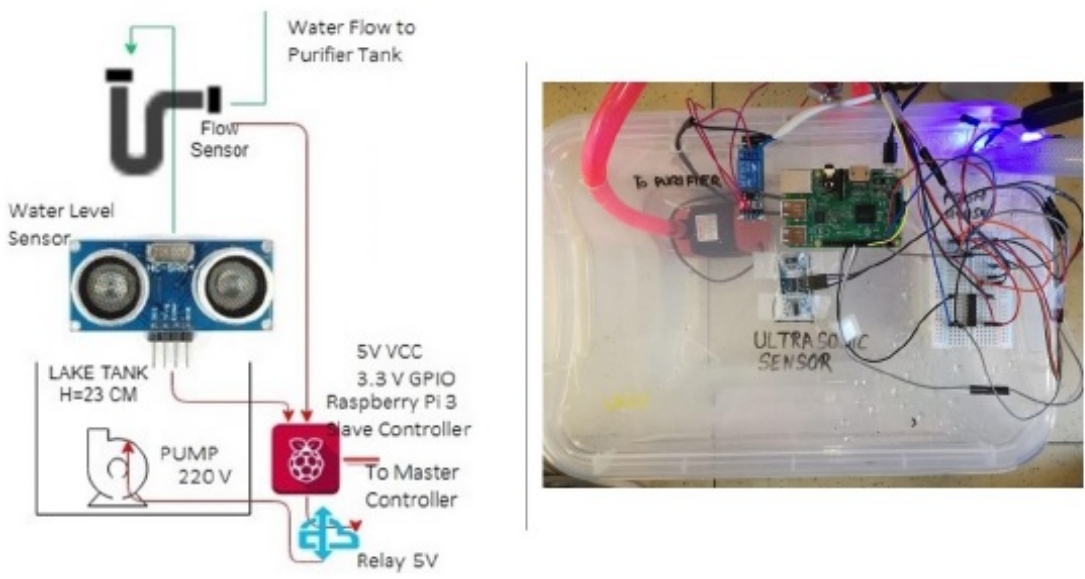


FIGURE 8.12: System at Lake Tank

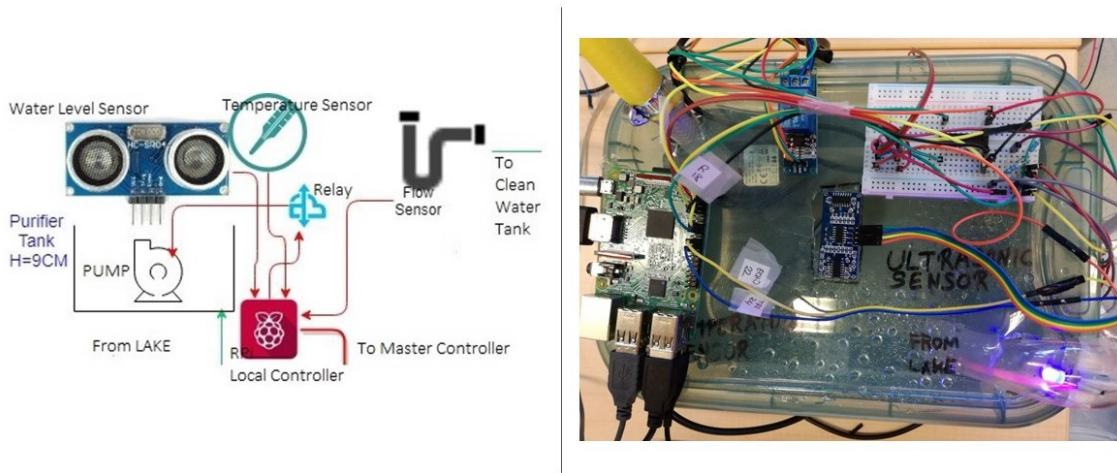


FIGURE 8.13: System at Purifier Tank

8.11 Master Controller

It records the reading of sensors and the status of an actuator of all slave. It also controls actuators associated with all slave, and update sensor reading and actuator status to Cloud. The code to perform all these tasks is available on Chapter A.6

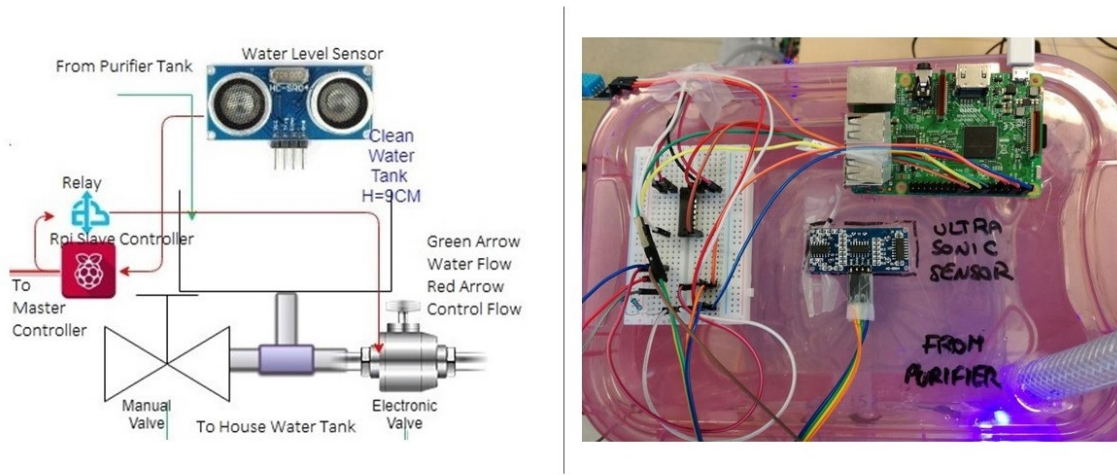


FIGURE 8.14: System at Clean Tank

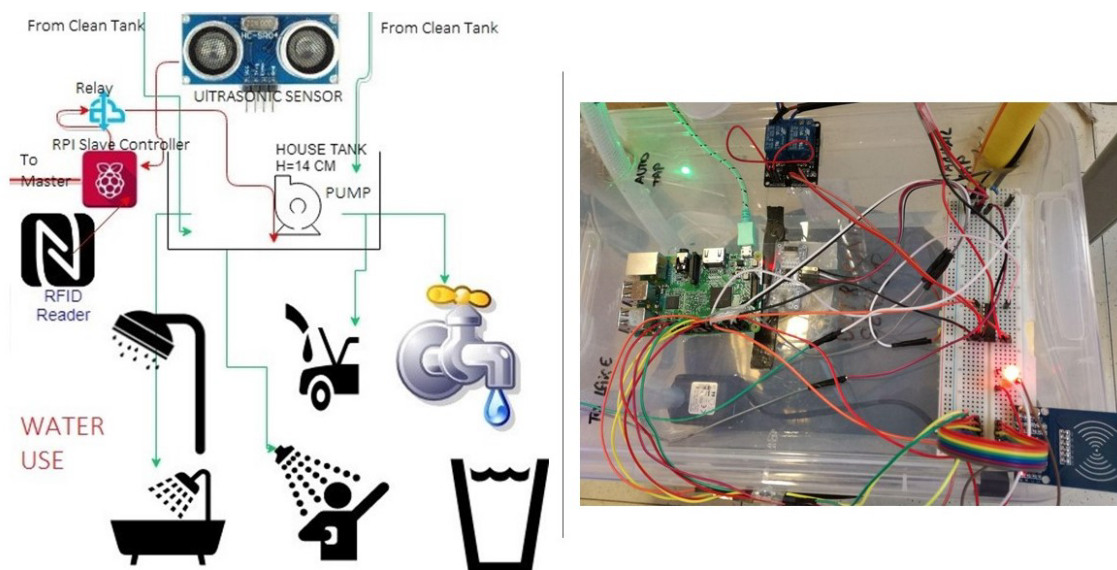


FIGURE 8.15: System at House Tank

Water symbolize purity, clarity and calmness. It is crucial to Buddhists to live in harmony with the environment.

Lord Buddha

9

Design of Emulator for Water Management Cyber Physical System: using Raspberry Pi with SECUBE

9.1 What is Secube?

SEcube is developed by Blu5 Group. Blu5 Group is a global alliance of Blu5 Labs Ltd Malta, CINI Italy, Lero Ireland, LIRMM France and TU Dortmund Germany. The core of SEcube is a 3-D System on Chip (SoC). This SoC consists of the following hardware components also shown in Fig. 9.1:

- STM32F4 Processor
- Lattice MachXO2-7000 FPGA
- EAL5+ Certified Smart Card

9.1.1 STM32F4 Processor

STM32F4 is basically an ARM Cortex M4 Risk CPU. It is manufactured by ST Microelectronics. It delivers high performance and has the following characteristics features:

- 2 MiB of Flash Memory
- 256 KiB of SRAM
- 32-bit parallelism
- Operating Frequency of 180 MHz

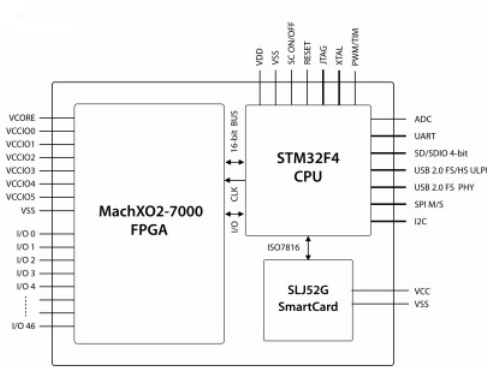


FIGURE 9.1: SECube Block Diagram [81]

- Low Power Consumption
- True Random Noise Generator (TRNG) embedded unit
- Memory Protection Unit (MPU)

9.1.2 Lattice MachXO2-7000 FPGA

Lattice MachXO2-7000 is a Field Programmable Gate Array (FPGA). It has the following characteristics features:

- 7,000 LUTs
- 240 Kib embedded Block RAM (BRAM)
- 256 Kib User Flash Memory
- 47 General purpose I/O: 32-bit external bus able to transfer 3.2 Gib/s. The FPGA is connected to the CPU through a 16-bit internal bus, provide a data transfer rate up to 1.6 Gib/s.

9.1.3 EAL5+ Certified Smart Card

This smart card is developed by Infineon. This smartcard is indeed an EAL5+ certified security controller. It has following characteristics features:

- ISO7816 Interface: Smartcard is connected to the CPU via this interface.
- JavaCard Platform, Global Platform 2.2
- 128 KiB Flash

9.2 About SECube Board

The jumper 3002 selects the connector to be used to power the board. Pin 1-2 selects J5000 otherwise Pin 2-3 selects J1000. The jumper J4003 allows direct control of SECube via the JTAG interface. In addition to J3002, J5000 and J1000, SECube Board has following other hardware components:

- J2000: Ethernet

- J3000: 1V2 power supply line
- J3001: 3V3 power supply line
- J4000: CPU GPIO
- J4001: CPU JTAG
- J4002: MicroSD Card
- J4004: FPGA
- J4005: Related to Smart Card
- LED0-LED7
- SW3000, SW4000, SW4001

9.3 About Software related to SECube Board

Following software need to install on SEcube Board.

- Operating System
- Java Runtime Environment
- Eclipse
- AC6 Tools
- STM32CubeMX
- Lattice Diamond Software for FPGA
- ST-Link/v2 in-circuit programmer
- ST-Link Utility
- Open Source SDK

9.3.1 AC6-Tools

SEcube has inbuilt ARM Cortex family of processors. AC-6 Tools is a suite of tools for C, C++, Assembly language programming for ARM processors, e.g. GNU compiler (GCC).

9.3.2 STM32CubeMX

It is installed as a plugin for Eclipse IDE. Following are installation steps:

- Go to Help menu of Eclipse IDE, and then point to Install New Software and click
- Click «Add»
- Type Name: System Workbench for STM32-Bare Machine Edition
- Type Location: <http://www.ac6tool.com/Eclipse-updates/org.openstm32.system-workbench.site>
- select OpenSTM32 Tools and click Next
- accept license Agreement
- click Finish
- restart Eclipse

9.3.3 ST-Link/v2 in-circuit programmer

Its JTAG interface is used to communicate with STM32 microcontrollers of SEcube Devkit.

9.4 Slave Controller at Lake Tank with SECUBE

There is similar a insertion of Secube function on every slave and master. So we are only adding the code for integration of SECube with Slave Controller on Lake in Appendix. Python Program to test sensors and actuators attached with Lake Tank using SECube is available on Chapter A.7.

We insert SECube in between slave and master Raspberry Pi as shown in Fig. 9.2.

SECube encrypts and decrypts message to/from master/slave using AES.

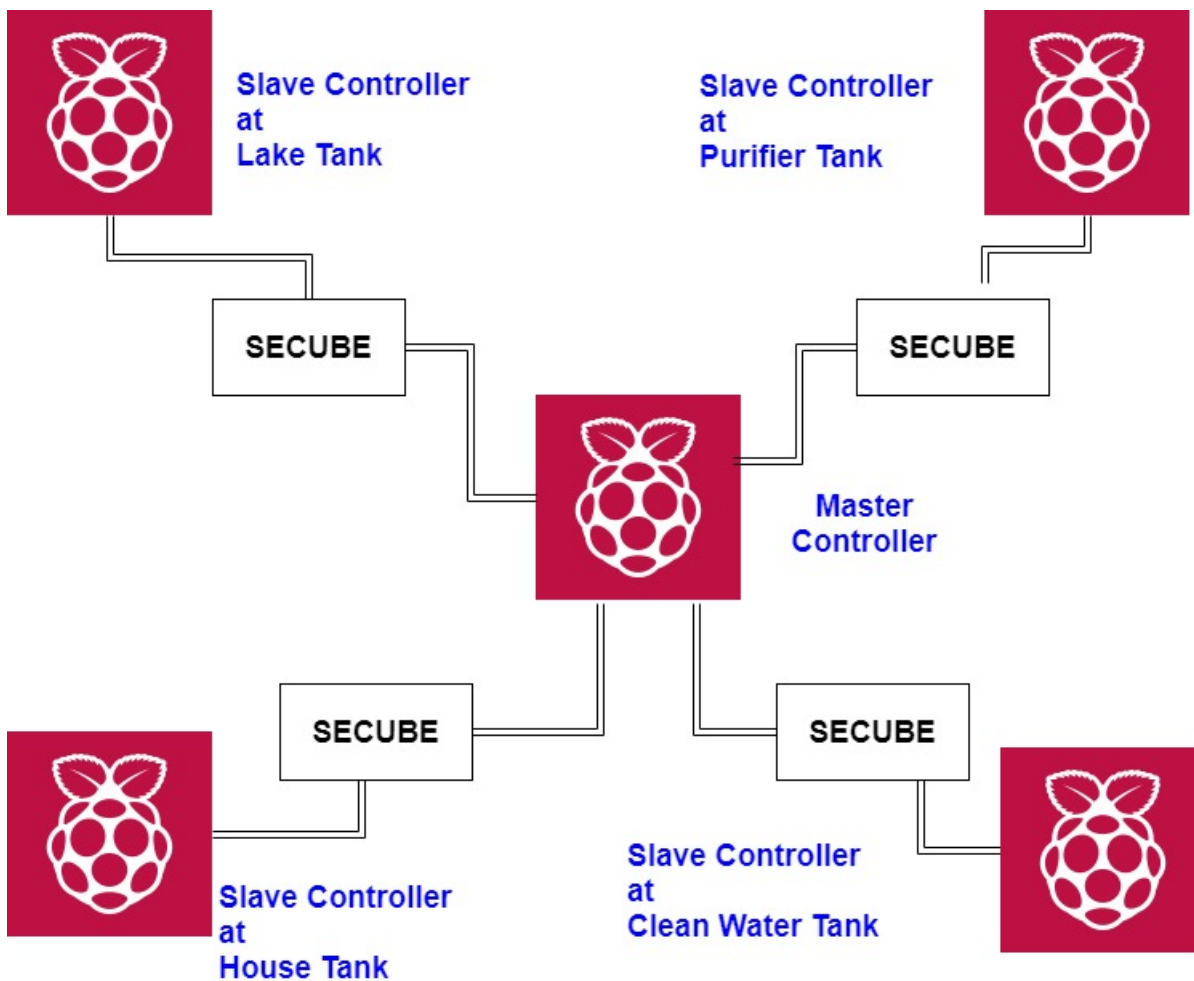


FIGURE 9.2: SECube in between Slave and Master Controller

When the well is dry, we will know the worth of water.

Benjamin Franklin

10

Web Interface and Telepot of Emulator for Water Management Cyber Physical System

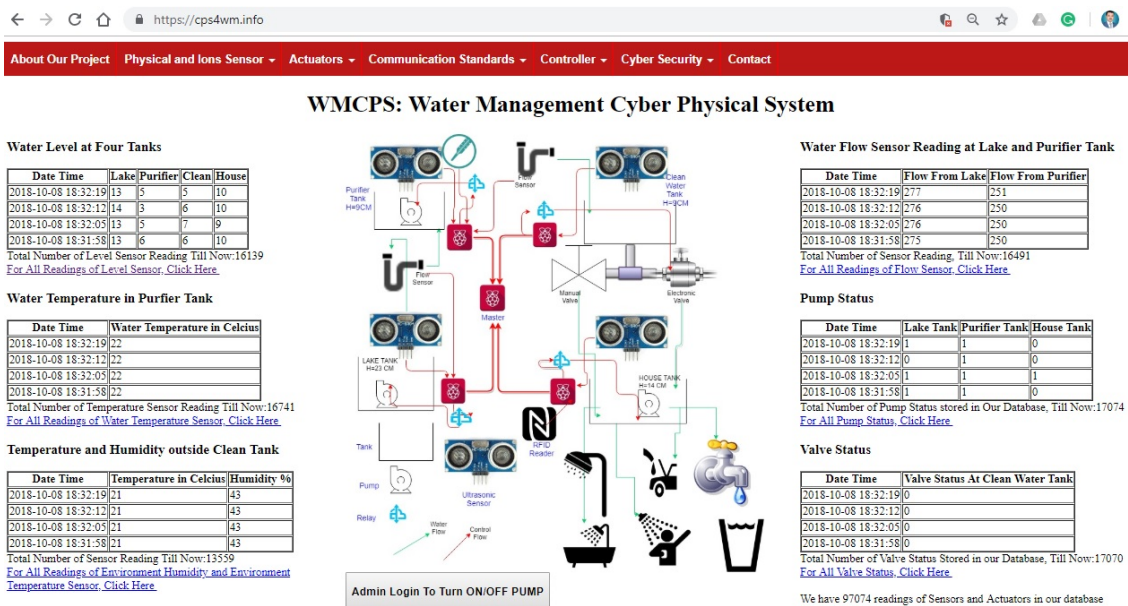
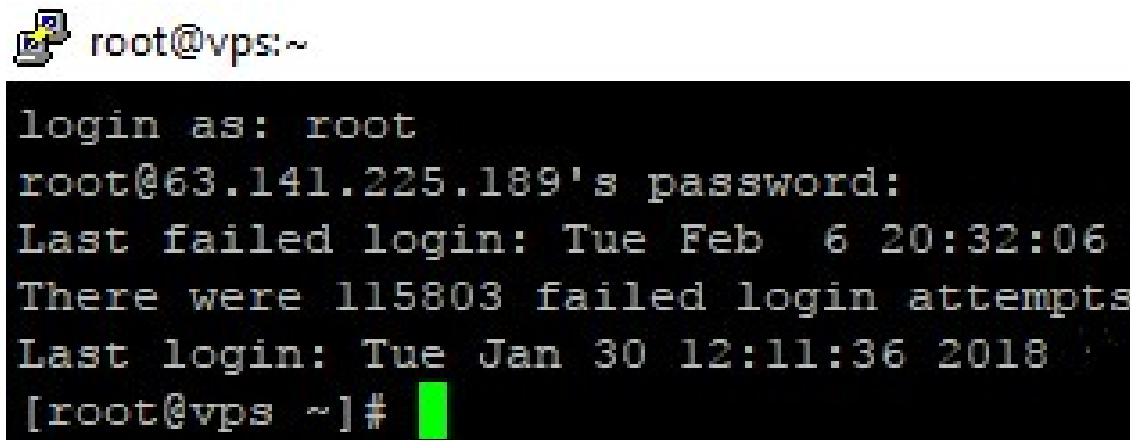


FIGURE 10.1: Home Page of https://cps4wm.info/

In order to render the work more interesting and instructive, several illustrations of sensor reading and actuator status have been displayed on the home page of the website, as shown in Fig. 10.1. This website is hosted on a Virtual Private Server (VPS). We access VPS using Putty. In putty, we enter 63.141.225.189 as an IP address of Hostname.



```

root@vps:~
login as: root
root@63.141.225.189's password:
Last failed login: Tue Feb  6 20:32:06
There were 115803 failed login attempts
Last login: Tue Jan 30 12:11:36 2018
[root@vps ~]#

```

FIGURE 10.2: Login as a root@63.141.225.189 to VPS

After login in VPS (as shown in Fig. 10.2), we have to access `public_html` directory available in `/home/cpswminf` directory because we saved all PHP file, HTML file, CSS file, the image file in the `public_html` directory. Following are the steps to edit the PHP file. In our website, `index.php` is a home page which will display when we open `http://cps4wm.info` in the browser.

```

[root@vps ] # cd /home
[root@vps ] # cd cpswminf
[root@vps ] # cd public_html
[root@vps ] # nano index.php

```

10.1 PHP

PHP (Hypertext Preprocessor) is a server-side scripting language designed for Web development. It is used to force https, establish a connection with database on MySQL server, fetch data from MySQL table and update sensor reading on MySQL table.

10.1.1 PHP Program to force https

`$_SERVER['HTTPS']` set to a non-empty value if the script was queried through the HTTPS protocol. So, if `$_SERVER['HTTPS']` is empty then we redirect to url after appending `https://` before that. `<?php`

```

//force https
if(empty($_SERVER['HTTPS']) || $_SERVER['HTTPS'] != 'on') {
header('Location: https://' . $_SERVER['HTTP_HOST'] .
$_SERVER['REQUEST_URI'], TRUE, 301);
die();
}
?>

```

10.1.2 PHP Program to Establish connection with database on MySQL Server

For our emulator, we have designed a database called cpswminf_project and in order to access we have to provide its credentials i.e. user name and password. 'cpswminf_pandey1', is the user name of database and 'CINI2017' is a password for accessing the database.

```
<?php
ob_start();

$db = new PDO('mysql:host=localhost;dbname=cpswminf_project', 'cpswminf_pandey1', 'CINI2017');
?>
```

10.1.3 PHP Program to Fetch Data from waterflow table

```
<?php
$stmt = $db->query('SELECT * FROM waterflow');

while($row = $stmt->fetch(PDO::FETCH_ASSOC))
{
if($row['value']==0)
$value="On";
$v=1;
}
else
{
$value="Off";
$v=0;
}

}

$num1=0;

?>
```

10.1.4 PHP Program to read the value of Level Sensor of four Tank

Reading of level sensor is a distance measured by ultrasonic sensors subtracted from the height of the tank. Distance table in MySQL has five columns. One is timestamp and others are the level of water in four tanks.

```
<?php
echo "<table border='1'>
<tr>
<th>Date Time</th>
```

```
<th>Lake</th>
<th>Purifier</th>
<th>Clean</th>
<th>House</th>
</tr>";
$result1=$db->query("select * from Distance ORDER BY timestamp DESC LIMIT 0,4");
while($row = $result1->fetch(PDO::FETCH_ASSOC))
{
echo "<tr>";
echo "<td>" . $row['timestamp'] . "</td>";
echo "<td>" . $row['lake'] . "</td>";
echo "<td>" . $row['purifier'] . "</td>";
echo "<td>" . $row['clean'] . "</td>";
echo "<td>" . $row['house'] . "</td>";
echo "</tr>";
}
echo "</table>";

?>
```

10.1.5 PHP Program to read the value of Temperature Sensor in Purifier Tank

The Water Temperature sensor is only attached with purifier tank. MySQL table Temperature stored the reading of temperature. The following query in result3 returns the five newest reading of water temperature sensors and then a dynamic table is created to show 5 different rows (<tr></tr>) of temperature table on the website.

```
<?php
echo "<table border='1'>
<tr>
<th> Date Time </th>
<th>Water Temperature in Celcius</th>
</tr>";
$result3=$db->query("SELECT * from Temperature ORDER BY timestamp DESC LIMIT 0,4");
while($row = $result3->fetch(PDO::FETCH_ASSOC))

echo "<tr>";
echo "<td>" . $row['timestamp'] . "</td>";
echo "<td>" . $row['purifier'] . "</td>";
echo "</tr>";

echo "</table>";

?>
```

10.1.6 PHP Program to Update reading of Environment Temperature and Humidity Sensor on our Project Website

The environment Temperature sensor is only attached with clean water tank. MySQL table TempHumEnv stored the reading of environment temperature and humidity. Following query in result5 returns the five newest reading of Temperature and Humidity sensors and then a dynamic table is created to show 5 different rows (<tr></tr>) of TempHumEnv table on website.

```
<?php

echo "<table border='1'>
<tr>
<th>Date Time </th>
<th>Temperature in Celcius</th>
<th>Humidity % </th>
</tr>";
$result5=$db->query("SELECT * from TempHumEnv ORDER BY timestamp DESC LIMIT 0,4");
while($row = $result5->fetch(PDO::FETCH_ASSOC))
{
echo "<tr>";
echo "<td>" . $row['timestamp'] . "</td>";
echo "<td>" . $row['temperature'] . "</td>";
echo "<td>" . $row['humidity'] . "</td>";
echo "</tr>";
}
echo "</table>";
?>
```

10.1.7 PHP Program to Update reading of Flow Sensor on our Project Website

There are two flow sensors in our emulator. One is with Lake tank and other is with Purifier tank. The MySQL table Flow stores reading of flow sensor associated with both Lake and Purifier tank.

```
<?php

echo "<table border='1'>
<tr>
<th>Date Time</th>
<th>Flow From Lake</th>
<th>Flow From Purifier</th>
</tr>";
$result7=$db->query("Select * From Flow ORDER BY timestamp DESC LIMIT 0, 4");
while($row = $result7->fetch(PDO::FETCH_ASSOC))

echo "<tr>";
```

```
echo "<td>" . $row['timestamp'] . "</td>";
echo "<td>" . $row['lake'] . "</td>";
echo "<td>" . $row['purifier'] . "</td>";

echo "</tr>";

echo "</table>";
?>
```

10.1.8 PHP Program to Print Pump Status on our Project Website

There are three pumps in our emulator with Lake tank, Purifier tank and House tank. The MySQL table PumpStatus stores the status of Pump associated with Lake, Purifier and House tank.

```
<?php

echo "<table border='1'>
<tr>
<th>Date Time</th>
<th>Lake Tank</th>
<th>Purifier Tank</th>
<th>House Tank</th>
</tr>";

$result9=$db->query("select * from PumpStatus ORDER BY timestamp DESC LIMIT 0,4");
while($row = $result9->fetch(PDO::FETCH_ASSOC))
{
echo "<tr>";
echo "<td>" . $row['timestamp'] . "</td>";
echo "<td>" . $row['lake'] . "</td>";
echo "<td>" . $row['purifier'] . "</td>";
echo "<td>" . $row['house'] . "</td>";
echo "</tr>";
}
echo "</table>";
?>
```

10.1.9 PHP Program to Print Valve Status on our Project Website

There is a solo valve in our emulator with Clean Water tank. The MySQL table ValveStatus stores the status of valve associated with Clean tank.

```
<?php
```

```
echo "<table border='1'>
<tr>
<th>Date Time</th>
<th>Valve Status At Clean Water Tank</th>
</tr>";
$result11=$db->query("select * from ValveStatus ORDER BY timestamp DESC LIMIT 0, 4");
while($row = $result11->fetch(PDO::FETCH_ASSOC))
{
echo "<tr>";
echo "<td>" . $row['timestamp'] . "</td>";
echo "<td>" . $row['clean'] . "</td>";
echo "</tr>";
}
echo "</table>";
?>
```

10.2 Javascript

Javascript is a high-level, dynamic, weakly typed, prototype-based, multi-paradigm and interpreted programming language.

10.2.1 Javascript Code to Generate Google Chart from MySQL tables

Google Charts is a Web-based service that generates graphical charts from user-supplied data. The user supplies information and a formatting specification expressed in JavaScript embedded in a Web page; in response, the Google Charts provides an image of the chart.

```
<script type="text/javascript" src="https://www.google.com/jsapi"></script>
<script type="text/javascript">
google.load("visualization", "1", packages:["corechart"]);
google.setOnLoadCallback(drawChart);
function drawChart()
var dataLake = google.visualization.arrayToDataTable([

['Current Time', 'Flow'],
<?php
$result2=$db->query("SELECT * FROM Flow ORDER BY timestamp DESC LIMIT 0, 50");
while( $row = $result2->fetch(PDO::FETCH_ASSOC))
{
echo "[".$row['timestamp'].",".$row['lake']."]";
}
?>
```

```

]);

var options = {
width: 600,
height: 500,
title: 'Water Flow in Liter From Lake
colors: ['#e0440e', '#e6693e', '#ec8f6e', '#f3b49f', '#f6c7b6']
};
var chart = new google.visualization.ColumnChart(document.getElementById("chart_flowLake"));
chart.draw(dataLake, options);

</script>

```

10.3 HTML

Hypertext Markup Language (HTML) is the standard markup language for creating web pages. We have designed HTML based web pages for every sensor, actuators, communication protocols used in emulator.

HTML is used to write a code for the home page of <https://cps4wm.info/>. HTML is also used to write a code for the web page of every sensor and actuator. In place of showing the code of 10+ web page, we have included the code of one web-page of temperature sensor in this thesis i.e.

<https://cps4wm.info/temperaturesensor.php> available on Chapter A.8. This page looks like Fig. 10.3 in a web browser.

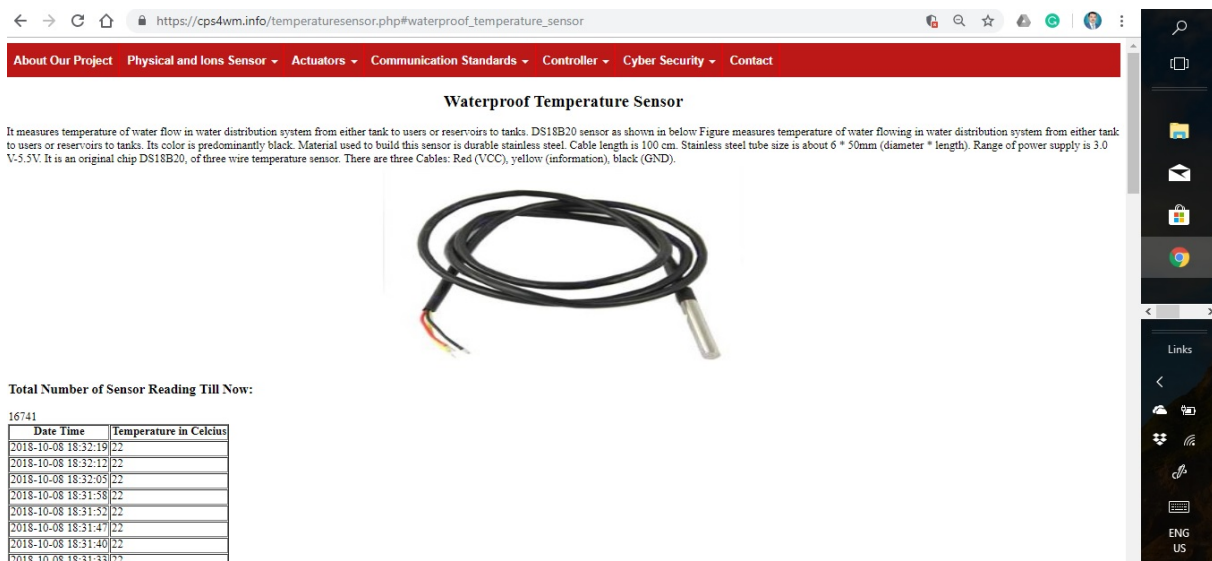


FIGURE 10.3: Web Page Dedicated to Temperature Sensor on Project Website

10.4 MySQL

MySQL is an open source relational database management system. The term MySQL is a combination of "My" and "SQL". "My" was the name of co-founder Michael Widenius's daughter. "SQL" is the abbreviation for Structured Query Language.

```
mysql> show tables;
+-----+
| Tables_in_cpswminf_project |
+-----+
| DistanceLOG                |
| FlowLOG                    |
| Humidity_TemperatureLOG    |
| WaterTemperatureLOG        |
| waterflow                  |
+-----+
5 rows in set (0.00 sec)
```

FIGURE 10.4: List of Tables in MYSQL Database of <http://cps4wm.info>

Reading of level sensor, flow sensor, temperature sensor and humidity sensor is stored in MYSQL database. We have created a database called cpswminf_project. To access the MYSQL database in VPS, we are using the following command.

```

[root@vps ] # mysql$
$mysql> show Databases;$
$mysql> use cpswminf_project;$
```

There are five tables in the database. To see the table name in MYSQL, we are using show tables; command in MySQL as shown in Fig. 10.4. When we operate the level sensor, the distance of water level from the ultrasonic sensor placed on top of the tank stored in DistanceLOG table with the current date and time as shown in Fig 10.5 and Fig. 10.6. Tank height is X. Then the level of water is X-distance.

10.5 Telegram

The Bot class is mostly a wrapper around Telegram Bot API. Many methods are straight mappings to Bot API methods. First, we install Telegram on my Android phone. Then open get_id bot and type

```
mysql> desc DistanceLOG;
+-----+-----+-----+-----+-----+
| Field      | Type      | Null  | Key  | Default |
+-----+-----+-----+-----+-----+
| datetime  | datetime  | YES   |      | NULL    |
| distance  | int(11)   | YES   |      | NULL    |
+-----+-----+-----+-----+-----+
```

FIGURE 10.5: Description of DistanceLOG Table

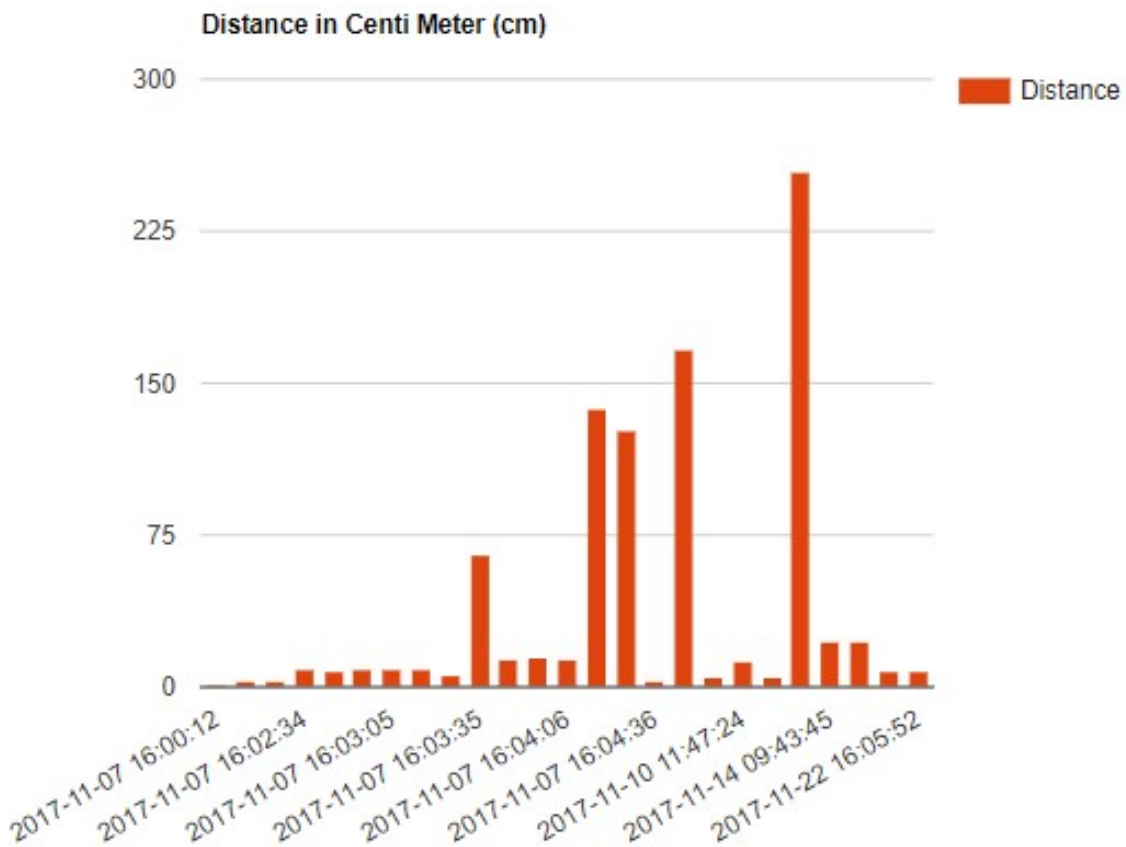


FIGURE 10.6: Distance of water from Level Sensor

my_id returns
 Hello Bishwajeet
 Your Chat ID= 656350789
 Then, we open BotFather bot and type
 newbot returns
 Alright, a new bot. How are we going to call it? Please choose the name of your bot.
 We call it WSSBoella2019 and press enter then it gives a message
 Good. Now let's choose a username for your bot. It must end in 'bot'. WSSBoella2019_bot is a username of our program run on Telegram.

BotFather: gives the following message

"Done! Congratulations on your new bot. You will find it at t.me/WSSBoella2019_bot."

Use this token to access the HTTP API: 743066254:AAHxlTF0rxh-gug0h7tyxgbPt2QOLip0c4I

Then we write a program in python and called it telegrambot_thread.py available on Chapter ??.

10.5.1 actuators.sh

```
#!/bin/bash
```

```
#Select Pump status
```

```
mysql -u cpswminf_pandey -pCINI2017 -D cpswminf_project -h 63.141.225.189 -P 3306 -B -N -e "Select lake from PumpStatus where timestamp = (Select max(timestamp) from PumpStatus);"
```

```
mysql -u cpswminf_pandey -pCINI2017 -D cpswminf_project -h 63.141.225.189 -P 3306 -B -N -e "Select purifier from PumpStatus where timestamp = (Select max(timestamp) from PumpStatus);"
```

```
mysql -u cpswminf_pandey -pCINI2017 -D cpswminf_project -h 63.141.225.189 -P 3306 -B -N -e "Select house from PumpStatus where timestamp = (Select max(timestamp) from PumpStatus);"
```

```
#Select Valve status
```

```
mysql -u cpswminf_pandey -pCINI2017 -D cpswminf_project -h 63.141.225.189 -P 3306 -B -N -e "Select clean from ValveStatus where timestamp = (Select max(timestamp) from ValveStatus);"
```

10.5.2 controlOFF.sh

```
#!/bin/bash
```

```
#Select Pump status
```

```
mysql -u cpswminf_pandey -pCINI2017 -D cpswminf_project -h 63.141.225.189 -P 3306 -B -N -e "Update waterflow set value=0"
```

10.5.3 controlON.sh

```
#!/bin/bash
```

```
#Select Pump status mysql -u cpswminf_pandey -pCINI2017 -D cpswminf_project -h 63.141.225.189 -P 3306 -B -N -e "Update waterflow set value=1"
```

10.5.4 sensors.sh

Fig. 10.7 is displaying the reading of all sensors attached to our system. It also shows the core temperature of raspberry and reading of temperature sensors using command core and temperature.

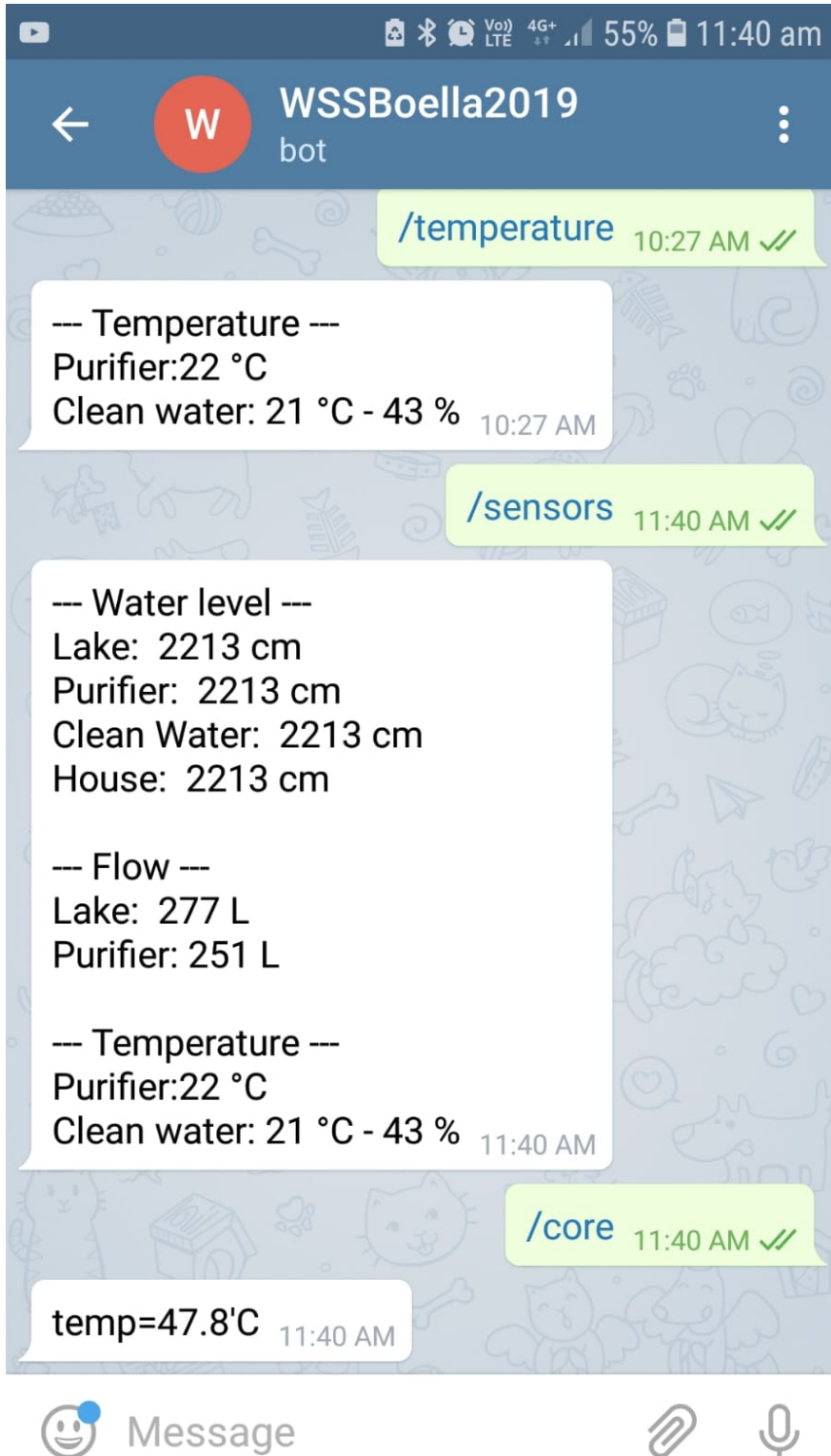


FIGURE 10.7: Telegram Bot of Our System Working on Android Phone: Sensor Reading

```
#!/bin/bash
```

```
#Select Distances
```

```
mysql -u cpswminf_pandey -pCINI2017 -D cpswminf_project -h 63.141.225.189 -P 3306 -B -N -e "Select lake from Distance where timestamp = (Select max(timestamp) from Distance);"
```

```
mysql -u cpswminf_pandey -pCINI2017 -D cpswminf_project -h 63.141.225.189 -P 3306 -B -N -e "Select purifier from Distance where timestamp = (Select max(timestamp) from Distance);"
```

```
mysql -u cpswminf_pandey -pCINI2017 -D cpswminf_project -h 63.141.225.189 -P 3306 -B -N -e "Select clean from Distance where timestamp = (Select max(timestamp) from Distance);"
```

```
mysql -u cpswminf_pandey -pCINI2017 -D cpswminf_project -h 63.141.225.189 -P 3306 -B -N -e "Select house from Distance where timestamp = (Select max(timestamp) from Distance);"
```

```
#Select Flow
```

```
mysql -u cpswminf_pandey -pCINI2017 -D cpswminf_project -h 63.141.225.189 -P 3306 -B -N -e "Select lake from Flow where timestamp = (Select max(timestamp) from Flow);"
```

```
mysql -u cpswminf_pandey -pCINI2017 -D cpswminf_project -h 63.141.225.189 -P 3306 -B -N -e "Select purifier from Flow where timestamp = (Select max(timestamp) from Flow);"
```

```
#Select Temperature
```

```
mysql -u cpswminf_pandey -pCINI2017 -D cpswminf_project -h 63.141.225.189 -P 3306 -B -N -e "Select purifier from Temperature where timestamp = (Select max(timestamp) from Temperature);"
```

```
mysql -u cpswminf_pandey -pCINI2017 -D cpswminf_project -h 63.141.225.189 -P 3306 -B -N -e "Select temperature from TempHumEnv where timestamp = (Select max(timestamp) from TempHumEnv);"
```

```
mysql -u cpswminf_pandey -pCINI2017 -D cpswminf_project -h 63.141.225.189 -P 3306 -B -N -e "Select humidity from TempHumEnv where timestamp = (Select max(timestamp) from TempHumEnv);"
```

10.5.5 sensors_onlyDistance.sh

```
#!/bin/bash
```

```
#Select Distances mysql -u cpswminf_pandey -pCINI2017 -D cpswminf_project -h 63.141.225.189 -P 3306 -B -N -e "Select lake from Distance where timestamp = (Select max(timestamp) from Distance);"
```

```
mysql -u cpswminf_pandey -pCINI2017 -D cpswminf_project -h 63.141.225.189 -P 3306 -B -N -e "Select purifier from Distance where timestamp = (Select max(timestamp) from Distance);"  
mysql -u cpswminf_pandey -pCINI2017 -D cpswminf_project -h 63.141.225.189 -P 3306 -B -N -e "Select clean from Distance where timestamp = (Select max(timestamp) from Distance);"  
mysql -u cpswminf_pandey -pCINI2017 -D cpswminf_project -h 63.141.225.189 -P 3306 -B -N -e "Select house from Distance where timestamp = (Select max(timestamp) from Distance);"
```

10.5.6 sensors_onlyFlow.sh

```
#!/bin/bash  
  
#Select Flow mysql -u cpswminf_pandey -pCINI2017 -D cpswminf_project -h 63.141.225.189 -P 3306 -B -N -e "Select lake from Flow where timestamp = (Select max(timestamp) from Flow);"  
mysql -u cpswminf_pandey -pCINI2017 -D cpswminf_project -h 63.141.225.189 -P 3306 -B -N -e "Select purifier from Flow where timestamp = (Select max(timestamp) from Flow);"
```

10.5.7 sensors_onlyTemperature.sh

```
#!/bin/bash  
  
#Select Temperature mysql -u cpswminf_pandey -pCINI2017 -D cpswminf_project -h 63.141.225.189 -P 3306 -B -N -e "Select purifier from Temperature where timestamp = (Select max(timestamp) from Temperature);"  
mysql -u cpswminf_pandey -pCINI2017 -D cpswminf_project -h 63.141.225.189 -P 3306 -B -N -e "Select temperature from TempHumEnv where timestamp = (Select max(timestamp) from TempHumEnv);"  
mysql -u cpswminf_pandey -pCINI2017 -D cpswminf_project -h 63.141.225.189 -P 3306 -B -N -e "Select humidity from TempHumEnv where timestamp = (Select max(timestamp) from TempHumEnv);"
```

*Cry. Forgive. Learn. Move on. Let your tears water
the seeds of your future happiness.*

Steve Maraboli

11

Future of Our Emulator

There is no absolute version of Emulator. Its definition change with time by time. A system which was taken as perfect on yesterday never treats as perfect today. Similarly, the contemporary definition of the perfect emulator will not pass the test of the future. It is just like the theory of "Fittest to survive." of Darwin, only fit system which can fulfil the requirement of that time always survive. To survive, it has to evolve something new in order to compete with tomorrow needs.

11.1 Make It Publicly Accessible

Now, this is accessible inside Polito. There is possible to integrate public IP address in this system. So, our system will become accessible across the globe.

11.2 Replace with Real Sensor and Actuators

We are using the smallest possible sensors and actuators only for demonstration purpose. In future, our project may acquire the real sensors and actuators used by Torino Water Supply System for the supply of water in Turin City.

11.3 Increase Number of Sensors and Actuators

Our published survey paper [16] analyzes on the usage of 17 different sensors and 18 different actuators in WM-CPS. In the current emulator, we are using only three sensors and two actuators. So, there is an open possibility to use 14 more sensors and 16 more actuators.

11.4 Explore Other Communication Standards

Our published paper [17] surveyed 22 communication standards for our future WM-CPS. In our thesis, we are working on mainly three: WiFi, Ethernet, Modbus. There is a possibility to use other 19 communication standards like Zigbee, Bluetooth, BACnet, and so on.

11.5 Integrate Niagara JACE Router

The JACE 8000 is a compact, embedded IoT (Internet of Things) controller and server platform for connecting multiple and diverse devices and sub-systems. Therefore, there is a possibility to replace our Raspberry Pi controller with Niagara JACE 800 router. With Internet connectivity and web serving capability, the JACE 8000 controller provides integrated control, supervision, data logging, alarming, scheduling and network management. So, we can replace our python script with the inner logic of Niagara. Niagara streams data, so we can replace MySQL server with internal data streaming of Niagara. Niagara has rich graphical displays to a standard Web browser via an Ethernet or wireless LAN, or Internet. Therefore, we may replace our web-interface with inbuilt rich graphical displays to a standard Web browser in Niagara.

11.6 Design Other Environment for Turin Node of Cyber Ranges

In addition to design an emulator for the water supply system, we may also design emulator for smart city project and robotics because Turin Node of Cyber Ranges is a collection of three nodes on water supply system, robotics and smart city.

Bibliography

- [1] A. G. Marcellana J. M. E. Morante A. C. D. Bonganay, J. C. Magno and N. G. Perez. Automated electric meter reading and monitoring system using zigbee-integrated raspberry pi single board computer via modbus. In *IEEE Students' Conference on, Electrical, Electronics and Computer Science (SCEECS)*, pages 1–6. IEEE, 2014.
- [2] V. P. Fernão D. Foito A. Cordeiro, P. Costa. Modbus protocol as gateway between different fieldbus devices - a didactic approach. *Advances in Intelligent Systems and Computing*, 715, 2018.
- [3] Kaiyu WanVangalur Alagar. Context-aware security solutions for cyber-physical systems. *Mobile Networks and Applications*, 19(2):212–226., 2014.
- [4] Pascal TROTTA Tiziana MARGARIA Alberto CARELLI, Giorgio DI NATALE. Towards model driven design of crypto primitives and processes. In *International Conference on Security and Management (SAM'16)*, pages 152–158. ISBN: 1-60132-445-6, CSREA Press, 2016.
- [5] Richard D. Alexander and Srinivas Panguluri. Cybersecurity terminology and frameworks. In *Cyber-Physical Security*, pages 19–47, 2017.
- [6] Litrico X. Sastry S. S. Bayen A. M. Amin, S. Stealthy deception attacks on water scada systems. *13th ACM international conference on Hybrid systems: computation and control*, pages 161–170, 2010.
- [7] Haider Abbas Anam Sajid and Kashif Saleem. Cloud-assisted iot-based scada systems security: A review of the state of the art and future challenges. *IEEE Access*, pages 1375–1384, 2016.
- [8] Brahim Hamid Anas Motii, Agns Lanusse and Jean-Michel Brue. Real-time evaluation of security patterns: A scada system case study. In *Proceedings of International Conference on Computer Safety, Reliability, and Security*, pages 375–389. Springer International Publishing, 2016., 2016.
- [9] Faye Anderson. Security and water. *Water Encyclopedia*, pages <http://www.waterencyclopedia.com/Re-St/Security-and-Water.html>, 2018.
- [10] Carole Ann. History of Water Pumps. http://www.ehow.com/facts_5031932_history-water-pumps.html, 2017. [Online; accessed on 15th February 2017].
- [11] Arduino. Arduino Mega 2560 R3, Last Accessed 7th February 2018,. <https://store.arduino.cc/usa/arduino-mega-2560-rev3>, 2018. [Online; accessed 19-July-2018].
- [12] American Water Work Association. Process control system security guidance for the water sector. *Industry Guidelines*, page <http://awwa.org/Portals/0/files/legreg/documents/AWWACybersecurityguide.pdf>, 2017.
- [13] Prasad R. V. Niemegeers I. G. Aust, Stefan. Outdoor long-range wlans: a lesson for iee 802.11 ah. *IEEE Communications Surveys Tutorials*, 17(3):1761–1775, 2015.
- [14] Ahmad Tbaileh Avik Dayal, Yi Deng and Sandeep Shukla. Vscada: A reconfigurable virtual scada test-bed for simulating power utility control center operations. *IEEE Power Energy Society General Meeting*, pages 1–5, 2015.
- [15] Yi Deng Avik Dayal, Ahmad Tbaileh and Sandeep Shukla. Distributed vscada: An integrated heterogeneous framework for power system utility security modeling and simulation. In *2015 Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES)*,, pages 1–6. IEEE, 2015.

- [16] Marco Indaco Bishwajeet Pandey, Giuseppe Airò Farulla and Paolo Prinetto. Survey of computational ability of sensors-actuators in water management cyber-physical system. *Journal of Advanced Research in Dynamical and Control Systems*, 10(04), 2018.
- [17] Marco Indaco Ludovico Iovino Bishwajeet Pandey, Giuseppe Airò Farulla and Paolo Prinetto. Design and review of water management system using ethernet, wi-fi 802.11n, modbus, and other communication standards. *Wireless Personal Communication*, 2018.
- [18] C. Boffa. Report on “tecnologie impiegate nella gestione dei sistemi idrici” or called eva system. *Report*, 2017.
- [19] Zita Vale Carlos Ramos and Luiz Faria. Cyber-physical intelligence in the context of power systems. In *In International Conference on Future Generation Information Technology*, pages 19–29. Springer Berlin Heidelberg, 2011.
- [20] Chen-Ching Liu Chee-Wooi Ten and Govindarasu Manimaran. Vulnerability assessment of cybersecurity for scada systems. *IEEE Transactions on Power Systems*, 23(4):1836–1846, 2008.
- [21] Pizzo A. D. Giugni M. Fontana N. Marini G. Proto D. Danner, A. Efficiency evaluation of a micro-generation system for energy recovery in water distribution networks. In *International conference on clean electrical power (ICCEP)*, pages 689–694. IEEE, 2015.
- [22] Andrew K. Dennis. Raspberry pi home automation with arduino. *2nd Edition*, 2015.
- [23] Antonio Pecchia Domenico Cotroneo and Stefano Russo. Towards secure monitoring and control systems: Diversify! In *In 2013 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 1–2. IEEE, 2013.
- [24] N. Parashiv F. Zamfir E. Pricop, J. Fattahi and E. Ghayoula. Method for authentication of sensors connected on modbus tcp. In *International Conference on Control, Decision and Information Technologies (CoDIT)*, Barcelona, 4, pages 0679–0683, 2017.
- [25] RICHARD ESPOSITO. Hackers penetrate water system computers. *ABC News Blog*, 2(2):http://blogs.abcnews.com/theblotter/2006/10/hackers_penetra.html, 2006.
- [26] S. Ferdoush and X. Li. Wireless sensor network system design using raspberry pi and arduino for environmental monitoring applications. *Procedia Computer Science*, 34:103–110, 2014.
- [27] F. Fredericka and K. R. Diller. Calculating the optimum temperature for serving hot beverage. *Burns*, 34(5):648–654, 2008.
- [28] Bla Genge and Christos Siaterlis. Developing cyber-physical experimental capabilities for the security analysis of the future smart grid. In *2nd IEEE PES International Conference and Exhibition on Innovative Smart Grid Technologies (ISGT Europe)*, pages 1–7. IEEE, 2011.
- [29] P. H. Gleick. Water and terrorism. *Water policy*, 8(6):481–503, 2006.
- [30] GlobalSpec. Water Valves. http://www.globaspec.com/learnmore/flow_control_flow_transfer/valves/water_valves, 2017. [Online; accessed on 15th February 2017].
- [31] Agrawal D.P. Yamaguchi S. et al. Gupta, B.B. Editorial security, privacy, and forensics in the critical infrastructure: advances and future directions. *Annals of Telecommunication*, 72:513–515, 2017.
- [32] Cogent Real-Time Systems Inc. What is opc? *OPC Data Hub*, 2010.
- [33] Peeyush Jain and Paritosh Tripathi. Scada security: a review and enhancement for dnp3 based systems. *CSI transactions on ICT*, 1(4):301–308, 2014.
- [34] Frank Jiang and Michael R. Frater. Towards a reliable aquatic-based cyber physical system: A new contextsituation aware low overhead routing scheme. In *3rd Annual International Conference In CyberTechnology in Automation, Control and Intelligent Systems (CYBER)*, pages 30–35. IEEE, 2013.

- [35] S. Jindarat and P. Wuttidittachotti. Smart farm monitoring using raspberry pi and arduino. In *Computer, Communications, and Control Technology (I4CT), 2015 International Conference on*, pages 284–288, 2015.
- [36] Daren BLYTHE Pat SWANSON Dennis DULING Dar GIBSON Chad CARPENTER Graham ROBERTS Jeff LEMERY Steve FISCHBEIN Adam MELBY Justin BURNETT, Frank R. RACK. Developing a hot-water drill system for the wissard project: 3. instrumentation and control systems. *Annals of Glaciology*, 55(68):303–310, 2014.
- [37] M.E. Kabay. Attacks on power systems: Hackers, malware,. <https://www.networkworld.com/article/2217684/data-center/attacks-on-power-systems-hackers-malware.html>, 2010.
- [38] KiwiPumps. Centrifugal Pump,. <http://www.kiwipumps.com/centrifugal-pumps2.html>, 2017. [Online; accessed on 15th February 2017].
- [39] R. Krauss. Combining raspberry pi and arduino to form a low-cost, real-time autonomous vehicle platform. In *American Control Conference (ACC)*, pages 6628–6633, 2016.
- [40] Y. Kuang. Communication between plc and arduino based on modbus protocol. In *Fourth International Conference on Instrumentation and Measurement, Computer, Communication, and Control (IMCCC)*, 4, pages 370–373. IEEE, 2014.
- [41] P. N. Nesterenko L. Barron and B. Paull. Use of temperature programming to improve the resolution of inorganic anions, haloacetic acids and oxyhalides in drinking water by suppressed ion chromatography. *Journal of Chromatography*, 1072(2):207–215, 2005.
- [42] Y. Chengbo L. Yanfei, W. Cheng and Q. Xiaojun. Research on zigbee wireless sensors network based on modbus protocol. In *International Forum on Information Technology and Applications, Chengdu*, pages 487–490, 2009.
- [43] N. M. Lazarin and C. E. Pantoja. A robotic-agent platform for embedding software agents using raspberry pi and arduino boards. *9th Software Agents, Environments and Applications School*, 2015.
- [44] Seshia SA Lee EA. Introduction to embedded systems - a cyber-physical systems approach. 2011.
- [45] R. Lemos. Safety: Assessing the infrastructure risk. *Tech News*, pages <https://www.cnet.com/news/safety-assessing-the-infrastructure-risk/>, 2002.
- [46] John Leyden. Water treatment plant hacked, chemical mix changed for tap supplies. *The Register*, page https://www.theregister.co.uk/2016/03/24/water_utility_hacked/, 2016.
- [47] Libelium. Temperature Sensor (Pt-1000), Smart Water Sensor Board,. http://www.libelium.com/downloads/documentation/smart_water_sensor_board.pdf, 2017. [Online; accessed on 15th February 2017].
- [48] R. A. Kassim M. S. B. Bahrudin and N. Buniyamin. Development of fire alarm system using raspberry pi and arduino uno. In *Electrical, electronics and system engineering (iceese), 2013 international conference on*, pages 43–48, 2013.
- [49] R. McMillan. Hackers break into water system network: Pennsylvania breach occurred via compromised laptop. *COMPUTER WORLD*, pages <http://www.computerworld.com/article/2547938/security0/hackers-break-into-water-system-network.html>, 2006.
- [50] S. Mhapankar. Implementing an efficient lightweight modbus protocol over a wireless sensor network. In *International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 473–479, 2017.
- [51] Elinor Mills. Hacker says he broke into texas water plant, others. *Mathematical Systems Theory*, pages <https://www.cnet.com/news/hacker-says-he-broke-into-texas-water-plant-others/>, 2011.
- [52] Wagh S. Joshi K. More, A. A test-bed for habitat monitoring system using wi-fi in wireless sensor networks. In *international conference on computational intelligence and computing research (ICCIC)*, pages 1–6. IEEE, 2015.
- [53] S. Singhal N. Agrawal. Smart drip irrigation system using raspberry pi and arduino. In *Computing, Communication Automation (ICCCA), 2015 International Conference on*, pages 928–932, 2015.

- [54] NicoHood. Arduino raspberry pi serial communication protocol via usb and c c,. *Github.com*, pages <https://github.com/NicoHood/NicoHood.github.io/wiki/Arduino—Raspberry—Pi—Serial—Communication—Protocol—via—USB—and—C—C>, 2016.
- [55] Oikonomakos P. Papadakis V. Inglezakis A. Dimitriou A. G. Bletsas A. Ntilis, D. Frequency planning for a multi-radio 802.11s city-wide water management network. In *ACM international workshop on cyber-physical systems for smart water networks*, pages 1–6. ACM, 2015.
- [56] Department of Justice: Office of Public Affair. Seven Iranians Working for Islamic Revolutionary Guard Corps-Affiliated Entities Charged for Conducting Coordinated Campaign of Cyber Attacks Against U.S. Financial Sector. <https://www.justice.gov/opa/pr/seven-iranians-working-islamic-revolutionary-guard-corps-affiliated-entities-charged>, 2016. [Online; accessed 19-July-2016].
- [57] J. P. Oleson. *Greek and Roman Mechanical Water-Lifting Devices: The History of a Technology*. University of Toronto Press, Toronto, Canada, 1 edition, 1984.
- [58] Bishwajeet Pandey. WMCPS: Water Management Cyber Physical System, Our Project Website. <http://cps4wm.info>, 2018. [Online; accessed 19-July-2018].
- [59] M. Postolache. A modbus/sl4dn gateway implementation for interfacing wsans to scada systems. In The editor, editor, *International Conference on System Theory, Control and Computing (ICSTCC)*, Sinaia, 21, pages 660–665, The address of the publisher, 2017.
- [60] Herbert G. Tanner Adam Jardine Prasanna Kannappan, Konstantinos Karydis and Jerrey Heinz. Incorporating learning modules improves aspects of resilience of supervisory cyber-physical systems. In *Proceedings of 24th Mediterranean Conference on Control and Automation (MED)*, pages 996–1001. IEEE, 2016.
- [61] . Presidency of the counsel of minister. EItaly’s National Strategic Framework for Cyberspace Security. <https://www.enisa.europa.eu/topics/national-cyber-security-strategies/ncss-map/ITNCSS.pdf>, 2013. [Online; accessed 10-October-2016].
- [62] Takale S. B. Ranade, P. Smart irrigation system using fpga based wireless sensor network. *International Research Journal of Engineering and Technology (IRJET)*, 3(5):2229–2232., 2016.
- [63] T. S. Riis. Modeling water distribution systems. Master’s thesis, Norwegian University of Science and Technology (NTNU), Norway, 2016. integration between SCADA systems and hydraulic network simulation models.
- [64] RISI. California canal system hack. *Repository of Industrial Security Incidents*, page http://www.risidata.com/Database/Detail/California_Canal_System_Hack, 2007.
- [65] Marco Rocchetto and Nils Ole Tippenhauer. On attacker models and profiles for cyber-physical systems. In *In European Symposium on Research in Computer Security*, pages 427–449. Springer International Publishing, 2016., 2016.
- [66] RPI. Raspberry Pi 3 Model B. <https://www.raspberrypi.org/products/raspberry-pi-3-model-b/>, 2018. [Online; accessed 19-July-2018].
- [67] N. Theodossiou W. Li M. Valipour A. Tamburrino S. I. Yannopoulos, G. Lyberatos and A. N. Angelakis. Evolution of water lifting devices (pumps) over the centuries worldwide. *Water*, 7(9):5031–5060., 2015.
- [68] E. Abraham S. Kartakis and J. A. McCann. Waterbox: A testbed for monitoring and controlling smart water networks. In *Proceedings of the 1st ACM International Workshop on Cyber-Physical Systems for Smart Water Networks*, page 8, 2015.
- [69] US Homeland Security. Recommended Practice: Developing an Industrial Control Systems Cybersecurity Incident Response Capability, . https://ics-cert.us-cert.gov/sites/default/files/recommended_practices/final-RP_ics_cybersecurity_incident_response_100609.pdf, 2016. [Online; accessed 19-July-2016].

- [70] Pradeeka Seneviratne. *Building Arduino PLCs: The essential techniques you need to develop Arduino-based PLCs*. APress, Berkely, CA, USA, 1 edition, 2017.
- [71] K. M. Shahanas and P. B. Sivakumar. Framework for a smart water management system in the context of smart city initiatives in india. *Procedia Computer Science*, 92:142–147, 2016.
- [72] Ling Shi. Analysis and design of secure cyber-physical systems. *Control Theory and Technology*, 12(4):413–314, 2011.
- [73] Sandeep K. Shukla. Cyber security of cyber physical systems: Cyber threats and defense of critical infrastructures. In *Proceedings of 29th International Conference on VLSI Design and 2016 15th International Conference on Embedded Systems (VLSID)*, pages 30–31. IEEE, 2016.
- [74] Adam Hahn Siddharth Sridhar and Manimaran Govindarasu. Cyberphysical system security for the electric power grid. *Proceedings of the IEEE*, 100(1):210–224, 2012.
- [75] J. Slay and M. Miller. Lessons learned from the maroochy water breach. *International Conference on Critical Infrastructure Protection*, pages 73–82, 2007.
- [76] SmartAmerica. ENHANCED WATER DISTRIBUTION INFRASTRUCTURE Enbaled by Cellular Based CPS: The SmartAmerica Challenge. <http://smartamerica.org/teams/enhanced-water-distribution-infrastructure/>, 2016. [Online; accessed 10-October-2016].
- [77] SRP. Theodore roosevelt dam. <http://www.srpnet.com/water/dams/roosevelt.aspx>, 2018.
- [78] A. Tamburrino. *Water Technology in Ancient Mesopotamia: Ancient Water Technologies*. Springer, Netherland, 1 edition, 2010.
- [79] theoryCIRCUIT. Water Flow Sensor YF-S201 Arduino Interface. <http://www.theorycircuit.com/water-flow-sensor-yf-s201-arduino-interface/>, 2018. [Online; accessed 19-July-2018].
- [80] Z. Shayan Z. Hafezi V. Hassanpour, S. Rajabi and M. M. Arefi. Low-cost home automation using arduino and modbus protocol. In The editor, editor, *International Conference on Control, Instrumentation, and Automation (ICCIA)*, 5, pages 284–289, The address of the publisher, 2017.
- [81] Antonio Varriale. SECube Data Sheet. <https://www.secube.eu/download/SECube-Datasheet-R7.pdf>, 2019. [Online; accessed 19-Feb-2019].
- [82] Brian Van Leeuwen Vincent Urias and Bryan Richardson. Supervisory command and data acquisition (scada) system cyber security analysis using a live, virtual, and constructive (lvc) testbed. In *In MILCOM 2012-2012 IEEE Military Commu- nications Conference*, pages 1–8. IEEE, 2012.
- [83] S. Alarefi W. F. Domoney, N. Ramli and S. D. Walker. Smart city solutions to water management using self-powered, low-cost, water sensors and apache spark data aggregation. In *Renewable and Sustainable Energy Conference (IRSEC), 2015 3rd International*, pages 1–4, 2015.
- [84] Danny Yadron. Iranian hackers infiltrated new york dam in 2013. *The Wall Street Journal*, pages <https://www.wsj.com/articles/iranian-hackers-infiltrated-new-york-dam-in-2013-1450662559>, 2015.
- [85] Kim-Kwang Raymond Choo Joseph K. Liu Yanjiang Yang, Jiqiang Lu. On lightweight security enforcement in cyber-physical systems. In *International Work- shop on Lightweight Cryptography for Security and Privacy*, pages 97–112. Springer International Publishing,, 2015.
- [86] Rohan Chabukswar Yilin Mo and Bruno Sinopoli. Detecting integrity attacks on scada systems. *IEEE Transactions on Control Systems Technology*, 22(4):1396–1407, 2014.
- [87] T. assios. *Hellenic Ancient Technology*. Kathimerini, Thens, Greece,, 1 edition, 1998.



Appendix

A.1 Python Program to Update Water Temperature on MySQL Server

```
import os
import time
import datetime
import glob
from time import strftime
```

```
os.system('modprobe w1-gpio')
os.system('modprobe w1-therm')
temp_sensor='/sys/bus/w1/devices/28-00000622fd44/w1-slave'
```

```
db=MySQLdb.connect(host="63.141.225.189", port='3306', user="cpswminf_pandey", passwd="CINI2017", db="cpswminf_project")
```

```
cur=db.cursor()
```

```
def tempRead():
t=open(temp_sensor,'r')
lines=t.readlines()
```

```
t.close()
temp_output=lines[1].find('t=')
if temp_output!=-1
temp_string=lines[1].strip()[temp_output+2:]
temp_c=float(temp_string)/1000.0
return round(temp_c,1)

while True:
temp=tempRead()
print temp
datetimeWrite=(time.strftime("%Y-%m-%d") + time.strftime("%H:%M:%S:"))
print datetimeWrite
sql=("INSERT INTO templog(timestamp, purifier) VALUES(%s, %s)", (datetimeWrite, temp))
try:
print 'Writing to database...?'
#Execute the SQL Command
cur.execute(*sql)
# Commit our changes in the database
db.commit()
print "Write Complete"
except:
# Rollbacks in case there is any Error
db.rollback()
print "Failed writing to database"
cur.close()
db.close()
break
```

A.2 Python Program to test sensors and actuators attached with Lake Tank

A.2.1 Functions

- def countPulse (channel)
- def setup ()
- def destroy ()
- def checkdist ()
- def level_thread (a)
- def flow_thread (a)
- def pump_thread (a)
- def run_updating_server ()

A.2.2 Variables

- int FLOW_SENSOR = 16
- int RELAY = 26
- int US_TRI = 22
- int US_ECHO = 24
- int countFlow = 0

```
#!/usr/bin/env python
# import the modbus libraries we need

from pymodbus.server.async import StartTcpServer
from pymodbus.device import ModbusDeviceIdentification
from pymodbus.datastore import ModbusSequentialDataBlock
from pymodbus.datastore import ModbusSlaveContext, ModbusServerContext
from pymodbus.transaction import ModbusRtuFramer, ModbusAsciiFramer

import time
import RPi.GPIO as GPIO

FLOW_SENSOR=16
RELAY = 26
US_TRI = 22
US_ECHO = 24

global countFlow
countFlow = 0

def countPulse(channel):
    global countFlow
    countFlow = countFlow + 1

def setup():

    GPIO.setmode(GPIO.BOARD)
    GPIO.setwarnings(False)

    #setup pump relay
    GPIO.setup(RELAY,GPIO.OUT)
```

```
#Setup Flow Sensor
GPIO.setup(FLOW_SENSOR, GPIO.IN, pull_up_down=GPIO.PUD_UP)
```

```
Setup Ultrasonic Sensor
GPIO.setup(US_TRI,GPIO.OUT,initial=GPIO.LOW)
GPIO.setup(US_ECHO,GPIO.IN)
```

```
def destroy():
GPIO.cleanup()
```

```
def checkdist():
GPIO.output(US_TRI,GPIO.HIGH)
time.sleep(0.000015)
GPIO.output(US_TRI,GPIO.LOW)
while not GPIO.input(US_ECHO):
pass
t1=time.time()
while GPIO.input(US_ECHO):
pass
t2=time.time()
return (t2-t1)*34000/2
```

```
# import the twisted libraries we need
```

```
from twisted.internet.task import LoopingCall
```

```
# define our callback process
```

```
def level_thread(a):
```

```
context = a[0]
register = 3
slave_id = 0x00
address = 0x11
```

```
distance =checkdist()
print "Distance: ",distance,"cm"
```

```
values = context[slave_id].getValues(register, address, count=1)
values[0] = distance
context[slave_id].setValues(register, address, values)
```

```

def flow_thread(a) :
    context = a[0]
    register = 3
    slave_id = 0x00
    address = 0x14

    flow = countFlow*2.25/1000
    print "Flow: " + str(int(flow)) + " L"

    values = context[slave_id].getValues(register, address, count=1)
    values[0] = int(flow)
    context[slave_id].setValues(register, address, values)

def pump_thread(a):

    context = a[0]
    register = 3
    slave_id = 0x00
    address = 0x12

    values = context[slave_id].getValues(register, address, count=1)
    pumpStatus = int(str(values[0]))

    print("Pump status: " + str(pumpStatus))
    if pumpStatus == 1:
        print "pump on!"
        GPIO.output(RELAY, GPIO.HIGH) pump on
    else:
        print "pump off!"
        GPIO.output(RELAY, GPIO.LOW) pump off

def run_updating_server():
    # initialize our data store

    store = ModbusSlaveContext( di=ModbusSequentialDataBlock(0, [17]*100),
    co=ModbusSequentialDataBlock(0, [17]*100),
    hr=ModbusSequentialDataBlock(0, [0]*100),
    ir=ModbusSequentialDataBlock(0, [17]*100))
    context = ModbusServerContext(slaves=store, single=True)

    # initialize the server information

```

```
identity = ModbusDeviceIdentification()
identity.VendorName = 'pymodbus'
identity.ProductCode = 'PM'
identity.VendorUrl = 'http://github.com/bashwork/pymodbus/'
identity.ProductName = 'pymodbus Server'
identity.ModelName = 'pymodbus Server'
identity.MajorMinorRevision = '1.0'
```

```
time = 0.5 500 milliseconds delay
loop_level = LoopingCall(f=level_thread, a=(context,))
loop_level.start(time, now=False) initially delay by time
```

```
time = 0.1 500 milliseconds delay
loop_valve = LoopingCall(f=pump_thread, a=(context,))
loop_valve.start(time, now=False) initially delay by time
```

```
GPIO.add_event_detect(FLOW_SENSOR, GPIO.FALLING, callback = countPulse)
```

```
time = 0.5 500 milliseconds delay
loop_level = LoopingCall(f=flow_thread, a=(context,))
loop_level.start(time, now=False) initially delay by time
```

```
StartTcpServer(context, identity=identity,
address=("192.168.43.102", 5020))
```

```
if __name__ == "__main__":
    setup()
    try:
        run_updating_server()
    except KeyboardInterrupt:
        destroy()
```

A.3 Python Program to test sensors and actuators attached with Purifier Tank

A.3.1 Functions

- def countPulse (channel)
- def setup ()
- def destroy ()

- def checkdist ()
- def temperature_thread (a)
- def flow_thread (a)
- def level_thread (a)
- def pump_thread (a)
- def run_updating_server ()

A.3.2 Variables

- int FLOW_SENSOR = 16
- int RELAY = 26
- int US_TRI = 24
- int US_ECHO = 38
- int countFlow = 0

```
#!/usr/bin/env python
# import the modbus libraries we need
from pymodbus.server.async import StartTcpServer
from pymodbus.device import ModbusDeviceIdentification
from pymodbus.datastore import ModbusSequentialDataBlock
from pymodbus.datastore import ModbusSlaveContext, ModbusServerContext
from pymodbus.transaction import ModbusRtuFramer, ModbusAsciiFramer

import time
import RPi.GPIO as GPIO
from waterTemp import read_temp

FLOW_SENSOR=16
RELAY = 18
US_TRI = 24
US_ECHO = 22

global countFlow
countFlow = 0

def countPulse(channel):
    global countFlow
    countFlow = countFlow + 1

def setup():
```

Appendix A Appendix

```
GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)

#setup pump relay
GPIO.setup(RELAY,GPIO.OUT)

#Setup Flow Sensor

GPIO.setup(FLOW_SENSOR, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.add_event_detect(FLOW_SENSOR, GPIO.FALLING, callback=countPulse)

#Setup Ultrasonic Sensor
GPIO.setup(US_TRI,GPIO.OUT,initial=GPIO.LOW)
GPIO.setup(US_ECHO,GPIO.IN)

def destroy():
    GPIO.cleanup()

def checkdist():
    GPIO.output(US_TRI,GPIO.HIGH)
    time.sleep(0.000015)
    GPIO.output(US_TRI,GPIO.LOW)
    while not GPIO.input(US_ECHO):
        pass
    t1=time.time()
    while GPIO.input(US_ECHO):
        pass
    t2=time.time()
    return (t2-t1)*34000/2

# import the twisted libraries we need
from twisted.internet.task import LoopingCall

# define our callback process

def temperature_thread(a) :
    context = a[0]
    register = 3
    slave_id = 0x00
    address = 0x13
```

```
temperature = read_temp()
print "Temperature: " + str(temperature) + " C"

values = context[slave_id].getValues(register, address, count=1)
values[0] = int(temperature)
context[slave_id].setValues(register, address, values)

def flow_thread(a):
context = a[0]
register = 3
slave_id = 0x00
address = 0x14

flow = countFlow*2.25/1000
print "Flow: " + str(int(flow)) + " L"

values = context[slave_id].getValues(register, address, count=1)
values[0] = int(flow)
context[slave_id].setValues(register, address, values)

def level_thread(a):

context = a[0]
register = 3
slave_id = 0x00
address = 0x11

distance = checkdist()
print "Distance: ",distance,"cm"

values = context[slave_id].getValues(register, address, count=1)
values[0] = distance
context[slave_id].setValues(register, address, values)

def pump_thread(a):

context = a[0]
register = 3
slave_id = 0x00
address = 0x12
```

Appendix A Appendix

```
values = context[slave_id].getValues(register, address, count=1)
pumpStatus = int(str(values[0]))

print("Pump status: " + str(pumpStatus))
if pumpStatus == 1:
    GPIO.output(RELAY, GPIO.HIGH) pump on
else:
    GPIO.output(RELAY, GPIO.LOW) pump off

def run_updating_server():
    # initialize our data store
    store = ModbusSlaveContext( di=ModbusSequentialDataBlock(0, [17]*100),
    co=ModbusSequentialDataBlock(0, [17]*100),
    hr=ModbusSequentialDataBlock(0, [0]*100),
    ir=ModbusSequentialDataBlock(0, [17]*100))
    context = ModbusServerContext(slaves=store, single=True)

# initialize the server information

identity = ModbusDeviceIdentification()
identity.VendorName = 'pymodbus'
identity.ProductCode = 'PM'
identity.VendorUrl = 'http://github.com/bashwork/pymodbus/'
identity.ProductName = 'pymodbus Server'
identity.ModelName = 'pymodbus Server'
identity.MajorMinorRevision = '1.0'

# run the server we want

time = 0.5 # 500 milliseconds delay
loop_level = LoopingCall(f=level_thread, a=(context,))
loop_level.start(time, now=False) # initially delay by time

print "thread level ok"

time = 0.5 # 500 milliseconds delay
loop_valve = LoopingCall(f=pump_thread, a=(context,))
loop_valve.start(time, now=False) # initially delay by time

print "thread pump ok"
time = 0.5 # 500 milliseconds delay
loop_level = LoopingCall(f=temperature_thread, a=(context,))
```



```

loop_level.start(time, now=False) # initially delay by time

print "thread temp ok"
time = 0.5 # 500 milliseconds delay
loop_level = LoopingCall(f = flow_thread, a = (context,))
loop_level.start(time, now = False)#initiallydelaybytime

print "thread flow ok"

StartTcpServer(context, identity=identity, address=("192.168.43.103", 5020))

print "TCP server ok"

if __name__ == "__main__":
    setup()
    try:
        run_updating_server()
    except KeyboardInterrupt:
        destroy()

```

A.4 Python Program to test sensors and actuators attached with Clean Tank

A.4.1 Functions

- def setup ()
- def destroy ()
- def checkdist ()
- def level_thread (a)
- def temp_hum_thread (a)
- def valve_thread (a)
- def run_updating_server ()

A.4.2 Variables

- int RELAY = 24
- int TEMP_HUM = 22
- int US_TRI = 16
- int US_ECHO = 18

```
#!/usr/bin/env python
# import the modbus libraries we need
from pymodbus.server.async import StartTcpServer
from pymodbus.device import ModbusDeviceIdentification
from pymodbus.datastore import ModbusSequentialDataBlock
from pymodbus.datastore import ModbusSlaveContext, ModbusServerContext
from pymodbus.transaction import ModbusRtuFramer, ModbusAsciiFramer

import time
import RPi.GPIO as GPIO

RELAY = 24
US_TRI = 16
US_ECHO = 18

def setup():

    GPIO.setmode(GPIO.BOARD)
    GPIO.setwarnings(False)

    #setup pump relay
    GPIO.setup(RELAY,GPIO.OUT)

    #Setup Ultrasonic Sensor
    GPIO.setup(US_TRI,GPIO.OUT,initial=GPIO.LOW)
    GPIO.setup(US_ECHO,GPIO.IN)

def destroy():
    GPIO.cleanup()

def checkdist():
    GPIO.output(US_TRI,GPIO.HIGH)
    time.sleep(0.000015)
    GPIO.output(US_TRI,GPIO.LOW)
    while not GPIO.input(US_ECHO):
        pass
    t1=time.time()
    while GPIO.input(US_ECHO):
        pass
    t2=time.time()
    return (t2-t1)*34000/2
```

```
# import the twisted libraries we need
from twisted.internet.task import LoopingCall

# define our callback process

def level_thread(a):

    context = a[0]
    register = 3
    slave_id = 0x00
    address = 0x11

    distance = checkdist()
    print "Distance: ",distance,"cm"

    values = context[slave_id].getValues(register, address, count=1)
    values[0] = distance
    context[slave_id].setValues(register, address, values)

def valve_thread(a):

    context = a[0]
    register = 3
    slave_id = 0x00
    address = 0x12

    values = context[slave_id].getValues(register, address, count=1)
    valveStatus = int(str(values[0]))

    print("Valve status: " + str(valveStatus))
    if valveStatus == 1:
        GPIO.output(RELAY, GPIO.HIGH) #valve on
    else:
        GPIO.output(RELAY, GPIO.LOW) #valve off

def run_updating_server():
    # initialize our data store
    store = ModbusSlaveContext( di=ModbusSequentialDataBlock(0, [17]*100),
    co=ModbusSequentialDataBlock(0, [17]*100),
    hr=ModbusSequentialDataBlock(0, [0]*100),
    ir=ModbusSequentialDataBlock(0, [17]*100))
```

```
context = ModbusServerContext(slaves=store, single=True)

# initialize the server information

identity = ModbusDeviceIdentification()
identity.VendorName = 'pymodbus'
identity.ProductCode = 'PM'
identity.VendorUrl = 'http://github.com/bashwork/pymodbus/'
identity.ProductName = 'pymodbus Server'
identity.ModelName = 'pymodbus Server'
identity.MajorMinorRevision = '1.0'
# run the server we want

time = 0.5 # 500 milliseconds delay
loop_level = LoopingCall(f=level_thread, a=(context,))
loop_level.start(time, now=False) # initially delay by time

time = 0.1 # 500 milliseconds delay
loop_valve = LoopingCall(f=valve_thread, a=(context,))
loop_valve.start(time, now=False) # initially delay by time

StartTcpServer(context, identity=identity, address=("192.168.43.104", 5020))

if __name__ == "__main__":
    setup()
    try:
        run_updating_server()
    except KeyboardInterrupt:
        destroy()
```

A.5 Python Program to test sensors and actuators attached with House Tank

A.5.1 Functions

- def setup ()
- def destroy ()
- def checkdist ()
- def level_thread (a)
- def pump_thread (a)

- def rfid_thread (a)
- def run_updating_server ()

A.5.2 Variables

- int RELAY = 26
- int US_TRI = 16
- int US_ECHO = 18

```
#!/usr/bin/env python
# import the modbus libraries we need
from pymodbus.server.async import StartTcpServer
from pymodbus.device import ModbusDeviceIdentification
from pymodbus.datastore import ModbusSequentialDataBlock
from pymodbus.datastore import ModbusSlaveContext, ModbusServerContext
from pymodbus.transaction import ModbusRtuFramer, ModbusAsciiFramer

import time
import RPi.GPIO as GPIO

RELAY = 26
US_TRI = 16
US_ECHO = 18

def setup():

    GPIO.setmode(GPIO.BOARD)
    GPIO.setwarnings(False)

    #setup pump relay
    GPIO.setup(RELAY,GPIO.OUT)
    GPIO.output(RELAY, GPIO.LOW)

    #Setup Ultrasonic Sensor
    GPIO.setup(US_TRI,GPIO.OUT,initial=GPIO.LOW)
    GPIO.setup(US_ECHO,GPIO.IN)

def destroy():
    GPIO.cleanup()

def checkdist():
    GPIO.output(US_TRI,GPIO.HIGH)
```

```
time.sleep(0.000015)
GPIO.output(US_TRI,GPIO.LOW)
while not GPIO.input(US_ECHO):
    pass
t1=time.time()
while GPIO.input(US_ECHO):
    pass
t2=time.time()
return (t2-t1)*34000/2

# import the twisted libraries we need
from twisted.internet.task import LoopingCall

# define our callback process

def level_thread(a):

    context = a[0]
    register = 3
    slave_id = 0x00
    address = 0x11

    distance = checkdist()
    print "Distance: ",distance,"cm"

    values = context[slave_id].getValues(register, address, count=1)
    values[0] = distance
    context[slave_id].setValues(register, address, values)

def pump_thread(a):

    context = a[0]
    register = 3
    slave_id = 0x00
    address = 0x12

    values = context[slave_id].getValues(register, address, count=1)
    pumpStatus = int(str(values[0]))

    print("Pump status: " + str(pumpStatus))
    if pumpStatus == 1:
```

```
GPIO.output(RELAY, GPIO.HIGH) pump on
else:
GPIO.output(RELAY, GPIO.LOW) pump off

def run Updating server():
# initialize our data store
store = ModbusSlaveContext( di=ModbusSequentialDataBlock(0, [17]*100), co=ModbusSequentialDataBlock(0, [17]*100),
hr=ModbusSequentialDataBlock(0, [0]*100), ir=ModbusSequentialDataBlock(0, [17]*100))
context = ModbusServerContext(slaves=store, single=True)
# initialize the server information

identity = ModbusDeviceIdentification()
identity.VendorName = 'pymodbus'
identity.ProductCode = 'PM'
identity.VendorUrl = 'http://github.com/bashwork/pymodbus/'
identity.ProductName = 'pymodbus Server'
identity.ModelName = 'pymodbus Server'
identity.MajorMinorRevision = '1.0'

# run the server we want

time = 0.5 # 500 milliseconds delay
loop_level = LoopingCall(f=level_thread, a=(context,))
loop_level.start(time, now=False) initially delay by time

time = 0.1 # 500 milliseconds delay
loop_valve = LoopingCall(f=pump_thread, a=(context,))
loop_valve.start(time, now=False) initially delay by time

StartTcpServer(context, identity=identity, address=("192.168.43.105", 5020))

if __name__ == "__main__":
setup()
try:
run Updating server()
except KeyboardInterrupt:
destroy()
```

A.6 Python Program execute at Master controller to manage sensors and actuators attached with four slaves

A.6.1 Functions

- def errorMsgTelegram ()
- def rfid_thread ()

A.6.2 Variables

- slave_Lake = ModbusTcpClient('192.168.43.102', port=5020)
- slave_Purifier = ModbusTcpClient('192.168.43.103', port=5020)
- slave_CleanWater = ModbusTcpClient('192.168.43.104', port=5020)
- slave_House = ModbusTcpClient('192.168.43.105', port=5020)
- db = pymysql.connect(host="63.141.225.189", port=3306, user="cpswminf_pandey", passwd="CINI2017", db="cpswminf_proj")
- cur = db.cursor()
- target
- timeStart = datetime.datetime.now() - datetime.timedelta(seconds=20)
- rr = slave_Lake.read_holding_registers(0x11, 1, unit=0x01)
- distanceLake = int((str(rr.registers[0])))
- distancePurifier = int((str(rr.registers[0])))
- distanceCleanWater = int((str(rr.registers[0])))
- distanceHouse = int((str(rr.registers[0])))
- datetimeCurrent = time.strftime("%Y-%m-%d %H:%M:%S")
- temperaturePurifierWater = int((str(rr.registers[0])))
- temperatureCleanEnv = int((str(rr.registers[0])))
- humidityCleanEnv = int((str(rr.registers[1])))
- flowFromPurifier = int((str(rr.registers[0])))
- flowFromLake = int((str(rr.registers[0])))
- int pumpLakeStatus = 0
- int pumpPurifierStatus = 0
- int valveCleanStatus = 0
- int pumpHouseStatus = 0
- rows = cur.fetchall()
- works = row
- unit
- timeStop = datetime.datetime.now()


```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from pymodbus.client.sync import ModbusTcpClient
from pprint import pprint
import time
import pymysql
#import mysql.connector

#we need 4 slaves
#1- Lake
#2- Purifier
#3- Clean water
#4- House

print("Server is running")

slave_Lake = ModbusTcpClient('192.168.43.102', port=5020)
slave_Purifier = ModbusTcpClient('192.168.43.103', port=5020)
slave_CleanWater = ModbusTcpClient('192.168.43.104', port=5020)
slave_House = ModbusTcpClient('192.168.43.105', port=5020)

slave_Lake.connect()
slave_Purifier.connect()
slave_CleanWater.connect()
slave_House.connect()

#connection to the web server
db=pymysql.connect(host="63.141.225.189", port=3306, user="cpswminf_pandey", passwd="CINI2017", db="cpswminf_project")
cur=db.cursor()

#always true while
while(1):

#DISTANCES
#read distance from slaves -> distance in all the slaves is always at address 0x11

try:
rr = slave_Lake.read_holding_registers(0x11, 1, unit=0x01) #addr, count, slave
distanceLake = int((str(rr.registers[0])))
```

Appendix A Appendix

```
rr = slave_Purifier.read_holding_registers(0x11, 1, unit=0x01) #addr, count, slave
distancePurifier = int((str(rr.registers[0])))
```

```
rr = slave_CleanWater.read_holding_registers(0x11, 1, unit=0x01) #addr, count, slave
distanceCleanWater = int((str(rr.registers[0])))
```

```
rr = slave_House.read_holding_registers(0x11, 1, unit=0x01) addr, count, slave
distanceHouse = int((str(rr.registers[0])))
```

```
datetimeCurrent=time.strftime("cur.execute("""Insert Into Distance(timestamp,lake, purifier, clean, house)
values(%s,%s,%s,%s,%s)""",(datetimeCurrent, distanceLake, distancePurifier, distanceCleanWater, distanceHouse))
except Exception:
print"————— Exception in distance!!"
```

```
#TEMPERATURE
```

```
try:
```

```
rr = slave_Purifier.read_holding_registers(0x13, 1, unit=0x01) #addr, count, slave
temperaturePurifierWater = int((str(rr.registers[0])))
print "Temperature Purifier: " + str(temperaturePurifierWater) + " ÂC"
```

```
datetimeCurrent=time.strftime("%Y-%m-%d %H:%M:%S")
cur.execute("""Insert Into Temperature(timestamp, purifier) values(%s,%s)""",(datetimeCurrent, temperaturePurifierWater))
```

```
except Exception:
```

```
print"————— Exception in temperature!!"
```

```
#FLOW try:
```

```
rr = slave_Purifier.read_holding_registers(0x14, 1, unit=0x01) #addr, count, slave
flowFromPurifier = int((str(rr.registers[0])))
print "Flow Sensor Purifier: " + str(flowFromPurifier) + " L"
```

```
rr = slave_Lake.read_holding_registers(0x14, 1, unit=0x01) #addr, count, slave
flowFromLake = int((str(rr.registers[0])))
print "Flow Sensor Lake: " + str(flowFromLake) + " L"
```

```
datetimeCurrent=time.strftime("%Y-%m-%d %H:%M:%S")
cur.execute("""Insert Into Flow(timestamp, lake, purifier) values(%s,%s,%s)""",(datetimeCurrent, flowFromLake, flowFromPurifier))
```

```
except Exception:
print "————— Exception in flow!!"

pumpLakeStatus = 0
pumpPurifierStatus = 0
valveCleanStatus = 0
pumpHouseStatus = 0

#PUMPS
#if the distance is not too small I can switch on pumps -> all the pumps are always at 0x12
if(distancePurifier>3 and distancePurifier<40):
print "pump lake on"
slave_Lake.write_register(0x12, 1, unit=0x01) pump on
pumpLakeStatus=1
else:
print "pump lake off"
slave_Lake.write_register(0x12, 0, unit=0x01) pump off

if(distanceCleanWater>3 and distanceCleanWater<40):
print "pump purifier on"
slave_Purifier.write_register(0x12, 1, unit=0x01)
pump on
pumpPurifierStatus=1
else:
print "pump purifier off"
slave_Purifier.write_register(0x12, 0, unit=0x01)
pump off

if(distanceHouse>7 and distanceHouse<40):
print "valve clean water on"
slave_CleanWater.write_register(0x12, 1, unit=0x01)
valve on
valveCleanStatus = 1
else:
print "valve clean water off"
slave_CleanWater.write_register(0x12, 0, unit=0x01)
valve off

if(distanceLake>9 and distanceLake<40):
print "pump House on"
slave_House.write_register(0x12, 1, unit=0x01) pump on
pumpHouseStatus = 1
else:
print "pump House off"
```

```
slave_House.write_register(0x12, 0, unit=0x01) pump off
```

```
datetimeCurrent=time.strftime("%Y-%m-%d %H:%M:%S")
cur.execute("""Insert Into PumpStatus(timestamp,lake, purifier, house) values(%s,%s,%s,%s)""",(datetimeCurrent, pumpLakeSta-
tus, pumpPurifierStatus, pumpHouseStatus))
cur.execute("""Insert Into ValveStatus(timestamp, clean) values(%s,%s)""",(datetimeCurrent, valveCleanStatus))
```

```
time.sleep(0.1)
```

```
slave_Lake.close()
slave_Purifier.close()
```

A.7 Python Program to test sensors and actuators attached with Lake Tank using SEcube

```
#!/usr/bin/env python
# ----- #
# import the modbus libraries we need
# ----- #
from pymodbus.server.async import StartTcpServer
from pymodbus.device import ModbusDeviceIdentification
from pymodbus.datastore import ModbusSequentialDataBlock
from pymodbus.datastore import ModbusSlaveContext, ModbusServerContext
from pymodbus.transaction import ModbusRtuFramer, ModbusAsciiFramer

import time
import RPi.GPIO as GPIO

FLOW_SENSOR=16
RELAY = 26
US_TRI = 22
US_ECHO = 24

PUMP_LED = 40

global countFlow
countFlow = 0
```

```
global counterWrong
counterWrong = 0

import base64
from Crypto import Random
from Crypto.Cipher import AES
global cipher
import re
import hmac
import hashlib
import binascii

import random

#SEcube import
import secube
import getpass
global cube
SE3_ALGO_AES = 0
SE3_ALGO_SHA256 = 1
SE3_ALGO_HMACSHA256 = 2
SE3_ALGO_AES_HMACSHA256 = 3
SE3_ALGO_MAX = 8

SE3_FEEDBACK_ECB = 1
SE3_FEEDBACK_CBC = 2
SE3_FEEDBACK_OFB = 3
SE3_FEEDBACK_CTR = 4
SE3_FEEDBACK_CFB = 5

SE3_DIR_SHIFT = 8
SE3_DIR_ENCRYPT = (1 « SE3_DIR_SHIFT)
SE3_DIR_DECRYPT = (2 « SE3_DIR_SHIFT)

SE3_CRYPTO_FLAG_FINIT = (1 « 15)
SE3_CRYPTO_FLAG_RESET = (1 « 14)
SE3_CRYPTO_FLAG_SETIV = SE3_CRYPTO_FLAG_RESET
SE3_CRYPTO_FLAG_SETNONCE = (1 « 13)
SE3_CRYPTO_FLAG_AUTH = (1 « 12)

def countPulse(channel):
global countFlow
```

Appendix A Appendix

```
countFlow = countFlow + 1
```

```
def prepareDataEnc(msg):
```

```
    algoEnc = SE3_ALGO_AES
```

```
    algoDigest = SE3_ALGO_HMACSHA256
```

```
    mode = SE3_FEEDBACK_ECB
```

```
    key_id_enc = 0
```

```
    key_id_dig = 1
```

```
    msg2Enc = str(int(time.time())) + str(msg) + str(int(time.time()))
```

```
    #ENCRYPTION
```

```
    #encrypt the message
```

```
    msgEncryptedSEcube = cube.encrypt(algorithm = algoEnc, mode = mode, key_id = key_id_enc, dataToChipher = msg2Enc)
```

```
    #transformalistof2hexchar
```

```
    msgEncrypted_list = ["{: 02X}"].format(c)forcinmsgEncrypted_SEcube]
```

```
    #joinallanddivideingroupof4hex
```

```
    #msgEncrypted_list = " ".join(msgEncrypted_list)
```

```
    #vect_encryption = re.findall('...!', str(msgEncrypted_list))
```

```
    #createlistof16integertosend
```

```
    vect_encryption = list(range(32))
```

```
    foriinrange(0, len(msgEncrypted_list)) :
```

```
        vect_encryption[i] = int(msgEncrypted_list[i], 16)
```

```
    #print(vect_encryption)
```

```
    #DIGEST
```

```
    #hmac_signature = create_sha256_signature(key = key_HMACSHA256, message=msgEncrypted.decode("utf-8"))
```

```
    hmac_signature_SEcube = cube.digest(algorithm = algoDigest, key_id = key_id_dig, data2Hash = msgEncrypted_SEcube)
```

```
    decodefromISO - 8859 - 1toutf - 8andtransformtoascii
```

```
    hmac_signature_SEcube_decoded = hmac_signature_SEcube.decode('ISO - 8859 - 1')
```

```
    vect_hmac_signature = ["{: 02X}"].format(c)forcinhmac_signature_SEcube_decoded.encode()]
```

```
    #joinallanddivedeingroupof4hex
```

```
    vect_hmac_signature = " ".join(vect_hmac_signature)
```

```
    vect_hmac_signature = re.findall('...!', str(vect_hmac_signature))
```

```
    #createlistof23integertosend
```

```
    foriinrange(0, len(vect_hmac_signature)) :
```

```
        vect_hmac_signature[i] = int(vect_hmac_signature[i], 16)
```

```
    #print(vect_hmac_signature)
```

```
    #list pre-computed of 32+23 -> 55 integer
```

```
    values = list(range(55))
```

```

for i in range(0, len(vect_encryption)):
    values[i] = vect_encryption[i]

for i in range(0, len(vect_hmac_signature)):
    values[i+32] = vect_hmac_signature[i]

#print (values)

return values

def checkAndReturnDecipheredData(msg):

timeArrived = int(time.time())

algoDec = SE3_ALGO_AES
algoDigest = SE3_ALGO_HMACSHA256
mode = SE3_FEEDBACK_ECB
key_id_enc = 0
key_id_dig = 1

#receive msg encrypted
msgEncrypted_list = list(range(32))
for i in range(0, 32):
    msgEncrypted_list[i] = chr(int((str(msg[i])))
#print (msgEncrypted_list)
msgEncrypted = str("".join(msgEncrypted_list))
#print (msgEncrypted)
#time.sleep(5)

#received hash
hmac_signature_received_list = list(range(23))
for i in range(0, 23):
    hmac_signature_received_list[i] = str(msg[32+i])
hmac_signature_received = str("".join(hmac_signature_received_list))
#print (hmac_signature_received)

#DIGEST
#hmac_signature = create_sha256_signature(key = key_HMACSHA256, message=msgEncrypted.decode("utf-8"))
hmac_signature_computed =cube.digest(algorithm = algoDigest, key_id = key_id_dig, data2Hash =
str(msgEncrypted).encode('ISO-8859-1'))
#decode from ISO-8859-1 to utf-8 and transform to ascii
hmac_signature_computed = hmac_signature_computed.decode('ISO-8859-1')
vect_hmac_signature = [":02X".format(c) for c in hmac_signature_computed.encode()]
#join all and divide in group of 4 hex

```

```
vect_hmac_signature = ".join(vect_hmac_signature)
vect_hmac_signature = re.findall('...', str(vect_hmac_signature))
#create list of 23 integer to send
for i in range(0, len(vect_hmac_signature)):
vect_hmac_signature[i] = int(vect_hmac_signature[i], 16)
hmac_signature_computed = str(".join(str(vect_hmac_signature[i])
for i in range(len(vect_hmac_signature)))) #print (hmac_signature_computed)

#check if the hash match
if(hmac_signature_received == hmac_signature_computed):
#print ("HASH EQUALS")

try:
#msgDecrypted = (cipher.decrypt(msgEncrypted)).decode('utf-8')
msgDecrypted = cube.decrypt(algorithm=algoDec, mode=mode, key_id=key_id_enc, dataChipered=msgEncrypted.encode('ISO-8859-1'))
msgDecrypted = msgDecrypted.decode('ISO-8859-1')
print (msgDecrypted)

timeReceived = msgDecrypted[:10]

#check if the time received and captured are not too far
if(int(timeReceived) - int(timeArrived)<5 and int(timeArrived) - int(timeReceived)<5):

data = msgDecrypted.replace(msgDecrypted[:10], ")
data = data.replace(data[10:], ")

return data
else:
return -200

except Exception:
return -100
else:
return -100

def setup():

GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)
```

```

#setup pump relay
GPIO.setup(RELAY,GPIO.OUT)

#Setup Flow Sensor
GPIO.setup(FLOW_SENSOR,GPIO.IN,pull_up_down = GPIO.PUD_UP)
GPIO.add_event_detect(FLOW_SENSOR,GPIO.FALLING,callback = countPulse)

#Setup Ultrasonic Sensor
GPIO.setup(US_TRI,GPIO.OUT,initial=GPIO.LOW)
GPIO.setup(US_ECHO,GPIO.IN)
GPIO.setup(PUMP_LED,GPIO.OUT)

def destroy():
GPIO.cleanup()

def checkdist():
GPIO.output(US_TRI,GPIO.HIGH)
time.sleep(0.000015)
GPIO.output(US_TRI,GPIO.LOW)
while not GPIO.input(US_ECHO):
pass
t1=time.time()
while GPIO.input(US_ECHO):
pass
t2=time.time()
return (t2-t1)*34000/2

# ----- #
# import the twisted libraries we need
# ----- #
from twisted.internet.task import LoopingCall

# ----- #
# define our callback process
# ----- #

def sensor_thread(a):

context = a[0]
register = 3
slave_id = 0x00
addressDistance = 0x82

```

```
addressFlow = 0x164
```

```
distance =int(checkdist())
print ("Distance: ",distance,"cm")
```

```
values = prepareDataEnc(msg=distance)
context[slave_id].setValues(register, addressDistance, values)
flow = countFlow*2.25/1000
print ("Flow: " + str(int(flow)) + " L")
```

```
values = prepareDataEnc(msg=int(flow))
context[slave_id].setValues(register, addressFlow, values)
```

```
def pump_thread(a):
```

```
context = a[0]
register = 3
slave_id = 0x00
address = 0x00
```

```
global counterWrong
```

```
values = context[slave_id].getValues(register, address, count=55)
pumpStatus_str = str(checkAndReturnDecipheredData(msg = values))
check = re.match(r"[0-9] + &[0-9] + |?[0-9] + ", pumpStatus_str)
if (check is not None and check.end() == len(pumpStatus_str)) :
    print("----- okcheckre")
    pumpStatus = int(pumpStatus_str)
else :
    print("----- errorcheckre")
    pumpStatus = -100
    print("pumpStatus" + str(pumpStatus))
    if (pumpStatus == -100) :
        print("Valvestatus - Errorcontrolmessage")
        counterWrong = counterWrong + 1

    raiseException("Errorcontrolmessage")
    elif (pumpStatus == -200) :
        print("Valvestatus - Errortime")
        counterWrong = counterWrong + 1

    raiseException("Errortime")
    elif (pumpStatus == -300) :
```

```

counterWrong = counterWrong + 1
print("Valvestatus - Genericerror")

```

```

if(counterWrong > 5 and pumpStatus<0): pumpStatus=2
if(counterWrong > 65000):
counterWrong = 6

```

```

if(pumpStatus > -1):
print ("Pump status: " + str(pumpStatus))
if pumpStatus == 1:
counterWrong=0
print ("pump on!")
GPIO.output(RELAY, GPIO.HIGH) pump
GPIO.output(PUMP_LED, GPIO.HIGH)
elif pumpStatus == 2:
print ("pump off! too many errors: " +
str(counterWrong))
GPIO.output(RELAY, GPIO.LOW) pump off
GPIO.output(PUMP_LED, GPIO.LOW)
else:
counterWrong=0
print ("pump off!")
GPIO.output(RELAY, GPIO.LOW) pump off
GPIO.output(PUMP_LED, GPIO.LOW)

```

```

def run_updating_server():

```

```

# ----- #
# initialize our data store
# ----- #

```

```

store = ModbusSlaveContext(
di=ModbusSequentialDataBlock(0, [17]*100), co=ModbusSequentialDataBlock(0, [17]*100), hr=ModbusSequentialDataBlock(0,
[0]*100), ir=ModbusSequentialDataBlock(0, [17]*100)) context = ModbusServerContext(slaves=store, single=True)

```

```

# ----- #
# initialize the server information
# ----- #
identity = ModbusDeviceIdentification()
identity.VendorName = 'pymodbus'
identity.ProductCode = 'PM'
identity.VendorUrl = 'http://github.com/bashwork/pymodbus/'
identity.ProductName = 'pymodbus Server'
identity.ModelName = 'pymodbus Server'
identity.MajorMinorRevision = '1.0'

```

Appendix A Appendix

```
# ----- #
# run the server we want
# ----- #
time = 0.5 500 milliseconds delay
loop_level = LoopingCall(f=sensor_thread, a=(context,))
loop_level.start(time, now=False) initially delay by time

time = 0.5 500 milliseconds delay
loop_level = LoopingCall(f=pump_thread, a=(context,))
loop_level.start(time, now=False) initially delay by time

StartTcpServer(context, identity=identity, address=("192.168.43.102", 5020))

if __name__ == "__main__":

    global cube
    print("NOW %d"%int(time.time()))
    devs=secube.SEcube.discover()
    print("Found %d devices"%len(devs))
    for dev in devs:
        print(dev)

    if len(devs)>0:

        cube=secube.SEcube(devs[0])
        password=getpass.getpass()
        password = "test"
        cube.login(pin=password, force=True)
        cube.crypto_set_time(int(time.time()))

    setup()
    try:
        run_updating_server()
    except KeyboardInterrupt:
        destroy()

    else:
        print ("No SEcube connected")
```

A.8 Code of <https://cps4wm.info/temperaturesensor.php>

```

<html>
<head>
<title>Water Flow Cyber Physical System</title>
<link href="http://fonts.googleapis.com/css?family=Bitterssubset=latin" rel="stylesheet" type="text/css">
<link rel="stylesheet" href="ultimatedropdown.css" />

<script type="text/javascript" src="https://www.google.com/jsapi"></script>
<script type="text/javascript">
google.load("visualization", "1", packages:["corechart"]);
google.setOnLoadCallback(drawChart);
function drawChart() {

var dataPurifier = google.visualization.arrayToDataTable([

['Current Time', 'Temperature'],
<?php
$result1=$db->query("SELECT * FROM Temperature ORDER BY timestamp DESC LIMIT $start_from,$results_per_page");
while($row = $result1-> fetch(PDO :: FETCH_ASSOC))
{
echo "[".$row['timestamp'].",".$row['purifier']."]";
}
? >

]);

var options = {
width: 600,
height: 500,
title: 'Temperature At Purifier Tank ',
colors: ['#e0440e', '#e6693e', '#ec8f6e', '#f3b49f',
'#f6c7b6']
;
var chart = new google.visualization.ColumnChart(document.getElementById("chart_temperaturePurifier"));
chart.draw(dataPurifier, options);
}
</script>

<script type="text/javascript" src="https://www.google.com/jsapi"></script>
<script type="text/javascript">
google.load("visualization", "1", packages:["corechart"]);

```

Appendix A Appendix

```
google.setOnLoadCallback(drawChart);
function drawChart() {

var dataPurifier = google.visualization.arrayToDataTable([

['Current Time', 'Temperature'],
<?php
$result2=$db->query("SELECT * FROM TempHumEnv ORDER BY timestamp DESC LIMIT $start_from2, $re-
sults_per_page");
while( $row = $result2->fetch(PDO::FETCH_ASSOC))
{
echo "[".$row['timestamp'].",".$row['temperature']."],"";
}
?>

]);

var options = {
width: 600,
height: 500,
title: 'Temperature Outside Clean Tank',
colors: ['#e0440e', '#e6693e', '#ec8f6e', '#f3b49f', 'f6c7b6']
};
var chart = new google.visualization.ColumnChart(document.getElementById("chart_temperatureClean"));
chart.draw(dataPurifier, options);
}
</script>

</head>

<body>
<div class="ultimatedropdown">
<ul>
<li><a href="index.php">About Our Project</a></li>

<li><a href="javascript:void(0)">Physical and Ions Sensor</a>
<ul>
<li><a href="ultrasonicsensor.php">Ultrasonic Sensor</a></li>
<li><a href="flowsensor.php">Flow Sensor</a></li>
<li><a href="javascript:void(0)">Temperature Sensor</a>
<ul>
<li><a href="temperaturesensor.phpwaterproof_temperature_sensor">Waterproof Temperature Sensor</a></li>
<li><a href="temperaturesensor.phphumidity_temperature_sensor">Humidity and Environment Temperature Sen-
sor</a></li>
</ul>
</li>
</ul>
</li>
</ul>
</div>
</body>
</html>
```

```

</li>
</ul>
</li>

```

```

<li><a href="javascript:void(0)">Actuators</a>
<ul>
<li><a href="pump.php">Pump</a></li>
<li><a href="valve.php">Valve</a></li>
</ul>
</li>

```

```

<li><a href="javascript:void(0)">Communication Standards</a> <ul>
<li><a href="wifi.php">WI-FI</a></li>
<li><a href="#">Modbus</a></li>
</ul>
</li>

```

```

<li><a href="javascript:void(0)">Controller</a>
<ul>
<li><a href="RPI3.php">Raspberry</a></li>
<li><a href="javascript:void(0)">SEcube</a>
<ul>
<li><a href="#">USEcube</a></li>
<li><a href="#">SEcube Board</a></li>
</ul>
</li>
<li><a href="#">Arduino</a></li>
</ul>
</li>

```

```

<li><a href="javascript:void(0)">Cyber Security</a>
<ul>
<li><a href="#">Previous Attacks on WMCPS</a></li>
<li><a href="#">Attacks on Our Emulator</a></li>
</ul>
</li>

```

```

<li><a href="contact.php">Contact</a></li>
</ul>
<br style="clear: left" />
</div>

```

```

<div id="waterproof_temperature_sensor">
<h2><center>Waterproof Temperature Sensor</center></h2>

```

Appendix A Appendix

It measures the temperature of water flow in the water distribution system from either tank to users or reservoirs to tanks. DS18B20 sensor as shown in below Figure measures temperature of water flowing in water distribution system from either tank to users or reservoirs to tanks. Its colour is predominantly black. The material used to build this sensor is durable stainless steel. Cable length is 100 cm. Stainless steel tube size is about 6 * 50mm (diameter * length). Range of power supply is 3.0 V-5.5V. It is an original chip DS18B20, of three wire temperature sensor. There are three Cables: Red (VCC), yellow (information), black (GND).

```
<center>

</center>
<table>
<tr>
<td>
<h3>Total Number of Sensor Reading Till Now:</h3>
<?php

$result3=$db->query("SELECT COUNT(timestamp) AS total FROM Temperature");
$row = $result3->fetch(PDO::FETCH_ASSOC);
$total_pages = ceil($row["total"]/$results_per_page);
echo$row["total"];

echo "<table border='1'>
<tr>
<th> Date Time </th>
<th>Temperature in Celcius</th>
</tr>";
$result4=$db->query("SELECT * from Temperature ORDER BY timestamp DESC LIMIT $start_from,$results_per_page");
while($row = $result4-> fetch(PDO :: FETCH_ASSOC))
{
echo" <tr > ";
echo" <td > ".$row['timestamp'].</td > ";
echo" <td > ".$row['purifier'].</td > ";
echo" </tr > ";
}
echo" </table > ";

echo "New Data-> ";
for ($i=1; $i<=$total_pages; $i++) {
echo "<a href='temperaturesensor.php?page=".$i.">".$i."</a> ";
};

echo " <-Old Data ° ";

?>
</td>
```



```

<td>
<div id="chart,temperaturePurifier" ></div >
</td >
</tr >
</table >

```

```

</div>

```

```

<div align="center" id="humidity,temperature,sensor" >
<h2 >< center > HumidityandEnvironmentSensor </center ></h2 >
DHT11sensorrequires3.5Vto5.5VDCforoperation.Itmeasuresoutsidetemperatureintherangeof0to50 < sup > o <
/sup > Cusingthermistor.Itmeasureshumidityintherangeof20–95%usingacapacitivehumiditysensor.Itgivesresultafteratleasteven
< center >
< imgsrc = "images/humidity,temperature.jpg" style = "width : 504px; height : 328px;" >< br / >
</center >

```

```

<table>
<tr>
<td>
<h3>Total Number of Sensor Reading Till Now:</h3>
<?php

```

```

$result5=$db->query("SELECT COUNT(timestamp) AS total FROM TempHumEnv");
$row = $result5->fetch(PDO::FETCH_ASSOC);
$total_pages = ceil($row["total"]/$results_per_page);
echo$row["total"];

```

```

echo "<table border='1'>
<tr>
<th>Time </th>
<th>Temperature in Celcius</th>
<th>Humidity %</th>
</tr>";
$result6=$db->query("SELECT * from TempHumEnv ORDER BY timestamp DESC LIMIT $start_from2,$results_per_page");
while($row = $result6-> fetch(PDO :: FETCH_ASSOC))
{
echo" < tr > ";
echo" < td > ".$row['timestamp'].</td > ";
echo" < td > ".$row['temperature'].</td > ";
echo" < td > ".$row['humidity'].</td > ";
echo" </tr > ";
}echo" </table > ";

```

```
echo "New Data-> ";
for ($i=1; $i<=$total_pages; $i++) {
echo "<a href='temperaturesensor.php?temppage2=".$i.">".$i."</a> ";
};
```

```
echo "<-Old Data ° ";
?>
</td>
<td>
<div id="chart_temperatureClean"></div>
</td>
</tr>
</table>

</div>
```

A.9 Python Code to Manage TeleBot on Telegram

```
$ sudo python /home/pi/telepot/telegrambot_thread.py
```

```
# -*- coding: utf-8 -*-
```

```
import sys
import time
import random
import datetime
import telepot
import os
import os.path
```

```
import threading
```

```
def handleCommand(command, chat_id):
if (command == 'Ping' or command == '/ping'):
bot.sendMessage(chat_id, "Pong")
```

```
elif (command=='/start'):
bot.sendMessage(chat_id, 'Welcome in the bot of the cyber range Water Supply System./commands to know all the
features of this bot!')
```

```
elif (command == 'Shutdown' or command == '/shutdown'):
bot.sendMessage(chat_id, "Bye Bye :)")
os.system('sudo shutdown -t 00')
```

```
elif (command == 'Reboot' or command == '/reboot'):
bot.sendMessage(chat_id, "A tra poco! :)")
os.system('sudo reboot')
```

```
elif (command == 'Core' or command == '/core'):
file_name = '/home/pi/telepot/core_' + str(chat_id) + '.txt'
cmd = '/opt/vc/bin/vcgencmd measure_temp > ' + file_name
os.system(cmd)
with open(file_name, "r") as f:
lines = f.read().splitlines()
s = lines[0]
bot.sendMessage(chat_id, s)
cmd = 'rm ' + file_name
os.system(cmd)
```

```
elif (command == 'Sensors' or command == '/sensors'):
file_name = '/home/pi/telepot/sensors_' + str(chat_id) + '.txt'
cmd = 'bash /home/pi/telepot/sensors.sh > ' + file_name
os.system(cmd)
```

```
with open(file_name, "r") as f:
lines = f.read().splitlines()
```

```
s = "— Water level —: " + lines[0] + " cm: " + lines[1] + " cmWater: " + lines[2] + " cm: " + lines[3] + " cm— Flow
—: " + lines[4] + " L: " + lines[5] + " L— Temperature —: " + lines[6] + " °Cwater: " + lines[7] + " °C - " + lines[8]
+ " bot.sendMessage(chat_id, s)
cmd = 'rm ' + file_name
os.system(cmd)
```

```
elif (command == 'Temperature' or command == '/temperature'):
file_name = '/home/pi/telepot/temperature_' + str(chat_id) + '.txt'
cmd = 'bash /home/pi/telepot/sensors_onlyTemperature.sh > ' + file_name
os.system(cmd)
```

```
with open(file_name, "r") as f:
lines = f.read().splitlines()
```

```
s = "— Temperature —: " + lines[0] + " °Cwater: " + lines[1] + " °C - " + lines[2] + " bot.sendMessage(chat_id, s)
cmd = 'rm ' + file_name
```

```
os.system(cmd)
```

```
elif (command == 'Distance' or command == '/distance'):
file_name = '/home/pi/telepot/distance_' + str(chat_id) + '.txt'
cmd = 'bash /home/pi/telepot/sensors_onlyDistance.sh >' + file_name
os.system(cmd)
```

```
with open(file_name, "r") as f:
lines = f.read().splitlines()
```

```
s = "— Water level —: " + lines[0] + " cm: " + lines[1] + " cmWater: " + lines[2] + " cm: " + lines[3] + " cm"
bot.sendMessage(chat_id, s)
cmd = 'rm ' + file_name
os.system(cmd)
```

```
elif (command == 'Flow' or command == '/flow'):
file_name = '/home/pi/telepot/flow_' + str(chat_id) + '.txt'
cmd = 'bash /home/pi/telepot/sensors_onlyFlow.sh >' + file_name
os.system(cmd)
```

```
with open(file_name, "r") as f:
lines = f.read().splitlines()
```

```
s = "— Flow —: " + lines[0] + " L: " + lines[1] + " L"
bot.sendMessage(chat_id, s)
cmd = 'rm ' + file_name
os.system(cmd)
```

```
elif (command == 'Actuators' or command == '/actuators'): file_name = '/home/pi/telepot/actuators_' + str(chat_id)
+ '.txt'
cmd = 'bash /home/pi/telepot/actuators.sh >' + file_name
os.system(cmd)
```

```
with open(file_name, "r") as f:
lines = f.read().splitlines()
```

```
if(lines[0] == "0"):
pumpLake = "OFF"
else:
pumpLake = "ON"
```

```

if(lines[1] == "0"):
    pumpPurifier = "OFF"
else:
    pumpPurifier = "ON"

if(lines[2] == "0"):
    pumpHouse = "OFF"
else:
    pumpHouse = "ON"

if(lines[3] == "0"):
    valveClean = "OFF"
else:
    valveClean = "ON"
s = "--- Pumps --- " + pumpLake + ": " + pumpPurifier + " : " + pumpHouse + "--- Valves ---water: " + valveClean
+ ""
bot.sendMessage(chat_id, s)
cmd = 'rm ' + file_name
os.system(cmd)

elif (command == 'Controlon' or command == '/controlon'):
    cmd = 'bash /home/pi/telepot/controlON.sh'
    os.system(cmd)
    s = "System runs automatically"
    bot.sendMessage(chat_id, s)

elif (command == 'Controloff' or command == '/controloff'):
    cmd = 'bash /home/pi/telepot/controlOFF.sh'
    os.system(cmd)
    s = "System stopped"
    bot.sendMessage(chat_id, s)

elif (command == 'Commands' or command == '/commands'):
    s = "/actuators -> Actuators status"+"/commands -> Commands list"+"/controlon -> Start the system"+"/controloff
-> Stop the system"+"/core -> Core temperature"+"/distance -> Distance from sensors"+ "/flow - Flow from sen-
sors"+ "/ping -> Are you still alive?" + "/reboot -> Reboot master" + "/sensors -> Sensors values" + "/shutdown ->
Shutdown master" + "/start - Start message" + "/temperature - Temperature from sensors" bot.sendMessage(chat_id, s)

else:
    s = command + ' -> Command not found'
    bot.sendMessage(chat_id, s)

```

```
def handle(msg):
    chat_id = msg['chat']['id']
    command = msg['text']

    if (chat_id == 622305966 or chat_id==656350789 or chat_id==484913053):
        print ('Got command: %s' % command)
        thread = threading.Thread(target=handleCommand, args=(command,chat_id))
        thread.start()
    else:
        bot.sendMessage(chat_id, "Access Denied!")
    return

bot = telepot.Bot('743066254:AAHxlTF0rxh-gug0h7tyxgbPt2QOLip0c4I')
bot.sendMessage('656350789', 'Master ON') Bishwajeet Pandey
bot.message_loop(handle)
print ('I am listening...')

while 1:
    time.sleep(10)
```