Western University Scholarship@Western

Electronic Thesis and Dissertation Repository

6-26-2017 12:00 AM

Design and Implementation of Smart Sensors with Capabilities of Process Fault Detection and Variable Prediction

An He The University of Western Ontario

Supervisor Dr. Jin Jiang *The University of Western Ontario*

Graduate Program in Electrical and Computer Engineering A thesis submitted in partial fulfillment of the requirements for the degree in Master of Engineering Science © An He 2017

Follow this and additional works at: https://ir.lib.uwo.ca/etd

Part of the Controls and Control Theory Commons, Electrical and Electronics Commons, and the Process Control and Systems Commons

Recommended Citation

He, An, "Design and Implementation of Smart Sensors with Capabilities of Process Fault Detection and Variable Prediction" (2017). *Electronic Thesis and Dissertation Repository*. 4687. https://ir.lib.uwo.ca/etd/4687

This Dissertation/Thesis is brought to you for free and open access by Scholarship@Western. It has been accepted for inclusion in Electronic Thesis and Dissertation Repository by an authorized administrator of Scholarship@Western. For more information, please contact wlswadmin@uwo.ca.

Abstract

A typical sensor consists of a sensing element and a transmitter. The major functions of a transmitter are limited to data acquisition and communication. The recently developed transmitters with 'smart' functions have been focused on easy setup/maintenance of the transmitter itself such as self-calibration and self-configuration. Recognizing the growing computational capabilities of microcontroller units (MCUs) used in these transmitters and underutilized computational resources, this thesis investigates the feasibility of adding additional functionalities to a transmitter to make it 'smart' without modifying its footprint, nor adding supplementary hardware. Hence, a smart sensor is defined as sensing elements combined with a smart transmitter. The added functionalities enhance a smart sensor with respect to performing process fault detection and variable prediction.

This thesis starts with literature review to identify the state-of-the-arts in this field and also determine potential industry needs for the added functionalities. Particular attentions have been paid to an existing commercial temperature transmitter named NCS-TT105 from Microcyber Corporation. Detailed examination has been made in its internal hardware architecture, software execution environment, and additional computational resources available for accommodating additional functions. Furthermore, the schemes of the algorithms for realizing process fault detection and variable prediction have been examined from both theoretical and feasibility perspectives to incorporate onboard NCS-TT105.

An important body of the thesis is to implement additional functions in the MCUs of NCS-TT105 by allocating real-time execution of different tasks with assigned priorities in the real-time operating system (RTOS). The enhanced NCS-TT105 has gone through extensive evaluation on a physical process control test facility under various normal/fault conditions. The test results are satisfactory and design specifications have been achieved.

To the best knowledge of the author, this is the first time that process fault detection and variable prediction have been implemented right onboard of a commercial transmitter. The enhanced smart transmitter is capable of providing the information of incipient faults in the process and future changes of critical process variables. It is believed that this is an initial step towards the realization of distributed intelligence in process control, where important decisions regarding the process can be made at a sensor level.

Keywords: Smart Sensor, Process Fault Detection, Variable Prediction, Real-time Operating System (RTOS)

Acknowledgments

I would like to express my sincere gratitude to my supervisor, Dr. Jin Jiang, for his motivation, expertise, immense knowledge, patience, and dedication. His guidance and inspiration helped me in the research and methods of studying. It was a grateful journey to learn from him and improve myself. I would further like to thank Dr. Xinhong Huang for her insightful guidance, encouragement, commitment and friendship. Without her support, this thesis would never have reached this stage.

I also feel very grateful to Dr. Vijay Parsa, Dr. Lyndon Brown, and Dr. Ahmed Hussein for their valuable courses. Leveraging the knowledge I learned from the course, I was able to realize my research and made rapid progress in my study.

I would also like to thank Dr. Mehrdad R. Kermani, Dr. Ilia Polushin, and Dr. Rajni Patel for letting me auditing their courses. The crossed knowledge broadened my views and inspired my ideas.

I have deep gratitude to thank the Natural Sciences and Engineering Research Council of Canada (NSERC), and University Network of Excellence in Nuclear Engineering (UNENE) for providing financial support throughout my graduate program.

I would also like to thank the generous support of transmitter implementation platform from Microcyber Corporation. I would also like to acknowledge the cooperation and technical support from Dr. Hong Wang, Mr. Jianwei Wei, Mr. Dekui Ning, and Mr. Zuye Yang from Microcyber Corporation; and Dr. Aidong Xu from Shenyang Institute of Automation Chinese Academy of Sciences. Without their support, I would not have achieved the accomplishment on practical application.

I feel very lucky that I have a wonderful research team. I am grateful to my colleague, Dr. Sungwhan Cho for his expertise and research suggestions; Dr. Ataul Bari for giving generously of his time offering help in the lab; Dr. Drew J. Rankin, Dr. Dennis Michaelson, Xirong Ning, and Syed Ahmed Raza for advices, discussions, and conversations.

Finally, I feel so grateful to my family for educating me to have a brave heart and providing me constant encouragement. They always back me up. This accomplishment would not have been possible without them.

This thesis is dedicated to my parents for their love, endless support, and encouragement

Abstract ii
Acknowledgmentsiii
Table of Contentsv
List of Tables ix
List of Figures xi
List of Abbreviationsxiv
Nomenclature xviii
Chapter 11
1 Introduction
1.1 Overview of Industrial Smart Sensors2
1.1.1 Brief Review of Industrial Sensors2
1.1.2 Composition of Smart Transmitter4
1.2 Shortcomings of Existing Smart Sensors and Potential Solutions
1.2.1 Shortcomings of Existing Smart Transmitters
1.2.2 A New Generation of Smart Sensors7
1.2.3 Potential Solutions through Integrating Data Analysis
1.3 Research Objectives, Methodologies, and Scope9
1.3.1 Objectives
1.3.2 Methodologies9
1.3.3 Scope10
1.4 Contributions of the Thesis11
1.5 Organization of the Thesis11
Chapter 213
2 Literature Review

Table of Contents

	2.1	1 Review of Existing Smart Sensors		
2.2 Existing Fault Detection Techniques			15	
		2.2.1	Classification of Fault Detection Methods	16
		2.2.2	Residual Generation by Model-based Methods	18
		2.2.3	Residual Evaluation for Fault Detection	20
	2.3	Existir	ng Techniques for Prediction of System Responses	21
		2.3.1	Models with and without Exogenous Variables	21
		2.3.2	Model-based Prediction Methods	24
	2.4	Real-ti	me Operating Systems in Transmitters	28
	2.5	Summ	ary	29
C	hapte	er 3		
3 Introduction of a Platform for Smart Sensor Implementation			30	
	3.1	Functi	onalities of NCS-TT105	30
	3.2	Hardw	are and Software	31
		3.2.1	Hardware	31
		3.2.2	Software	34
	3.3	Standa	rds of Compliance and Specifications of the Platform	35
		3.3.1	Standards followed by NCS-TT105	35
		3.3.2	Specifications of NCS-TT105	
	3.4	Summ	ary	
C	hapte	er 4		40
4	Inve	estigatio	on of the Algorithms for Smart Functions	40
	4.1	Goals	and Specifications	40
		4.1.1	Process Fault Detection Smart Function	40
		4.1.2	Variable Prediction Smart Function	41

	4.2	Algori	thms for Process Fault Detection	43
		4.2.1	ARX and EWRLS for Residual Generation	43
		4.2.2	CUSUM for Residual Evaluation	48
	4.3	Algori	thms for Variable Prediction	52
		4.3.1	Kalman Predictor Prediction	52
		4.3.2	Grey Model Prediction	55
	4.4	Summ	ary	64
C	hapte	er 5		65
5	Imp	lement	ation of Smart Functions in Real-time Operating System	65
	5.1	The Pr	ocess Fault Detection Task	65
		5.1.1	EWRLS for Residual Generation	66
		5.1.2	CUSUM for Residual Evaluation	69
	5.2	The V	ariable Prediction Task	72
		5.2.1	Kalman Predictor	73
		5.2.2	Grey Model	78
	5.3	The Bi	directional Communication Task	84
	5.4	Impler	nentation of Multitasking in the Real-time Operating System	86
		5.4.1	Configuration of Multitasking Scheduling with RMS Rules	87
		5.4.2	Implementation of Three Tasks in Nucleus RTOS	92
	5.5	Instruc	ctions of using the Designed Smart Sensor	96
	5.6	Summ	ary	100
C	hapte	er 6		
6	Ver	ificatio	n and Validation of the Smart Sensor	
	6.1	Introdu	action to the Test Environment	102
		6.1.1	Introduction of the Test Facility	

		6.1.2	Introduction of the DCS Systems	.104
	6.2	Verific	ation to Communication Channels in Smart Sensors	.107
	6.3	Offline	e Validation of Smart Functions in MATLAB	.109
		6.3.1	Process Fault Detection	.110
		6.3.2	Variable Prediction	.116
	6.4	Online	Validation	.125
		6.4.1	Multitasking in RTOS	.126
		6.4.2	Process Fault Detection	.129
		6.4.3	Variable Prediction	.138
	6.5	Summa	ary of the Designed Smart Sensor	.146
	6.6	Summa	ary	.147
Cl	napte	er 7		.148
7	Sun	nmary a	nd Conclusions	.148
	7.1	Summa	ary	.148
	7.2	Conclu	isions	.149
	7.3	Future	Works	.150
Re	efere	nces		.152
Aj	open	dices		.160
	App	endix A	A: Standards for Sensing in NCS-TT105	.160
	App	endix F	3: Introduction of PROFIBUS-PA	.162
	App	endix (C: Introduction of Nucleus RTOS	.166
Cı	ırricı	ulum Vi	itae	.172

List of Tables

Table 2.1: Classification of Fault Detection Methods [54][55] 16
Table 3.1: NCS-TT105 Basic Technical Specifications [94]
Table 3.2: RTD and TC Specifications of NCS-TT105 [94]
Table 4.1: Specifications of Process Fault Detection Smart Function
Table 4.2: Specifications of Variable Prediction Smart function
Table 4.3: Matrix Dimension List of Kalman Predictor
Table 4.4: Algorithms Summary of Process Fault Detection and Variable Prediction64
Table 5.1: The procedure of EWRLS Parameter Estimation for Residual Generation68
Table 5.2: The Procedure of Two-sided CUSUM for Residual Evaluation
Table 5.3: The Procedure of the Kalman Predictor and Multistep Iteration
Table 5.4: The Procedure of Grey Model Prediction 78
Table 5.5: List of Communication Interface of PROFIBUS-PA in NCS-TT10585
Table 5.6: Three Tasks for Integration in RTOS 93
Table 5.7: Implementation Summary of Three Tasks 96
Table 5.8 The Summary of Setting for Process Fault Detection Function
Table 5.9 The Summary of Setting for Variable Prediction Function 98
Table 6.1: Essential Equipment in Primary Water Loop Subsystem of the NPCTF104
Table 6.2: Four Essential Variables for the Algorithms in SMART NCS-TT105107
Table 6.3: Simulated Faults in Primary Water Loop System for Offline Validation111

Table 6.4: Result of Offline Validation of Fault Detection Test	116
Table 6.5: Scenarios of Offline Variable Prediction Test	116
Table 6.6: Comparison of Prediction by Grey Model and Combined Method	120
Table 6.7: Validation Results of Offline Prediction Test	124
Table 6.8: Validation Results of Execution Time of Three Tasks	128
Table 6.9: Validation Results of Period Time of Three Tasks	128
Table 6.10: Simulated Faults in Primary Water Loop System for Online Validation	130
Table 6.11: Online Validation Results of Fault Detection Test	138
Table 6.12: Scenarios of Online Variable Prediction Test	140
Table 6.13: Online Validation Results of Prediction Test	145
Table 6.14: Performance of Designed Smart Functions in the Smart Sensor	146
Table 6.15: Summary of V&V Test	147
Table B.1: The Specifications of PROFIBUS-PA [100]	163
Table C.1: Execution Time and Period Time of Tasks in Scenario-1	167
Table C.2: Analysis of Three Tasks in Scenario-1	169
Table C.3: Execution Time and Period Time of Tasks in Scenario-2	169
Table C.4: Analysis of Three Tasks in Scenario-2	171

List of Figures

Figure 1.1: Overview of a Typical Industrial Sensor and Smart Sensor	2
Figure 1.2: Siemens SITRAN TF Functions Diagram [12]	3
Figure 1.3: Smart Transmitter Architecture	4
Figure 2.1: Model-based Fault Detection Framework [52]	17
Figure 2.2: Illustration of a Prediction with Known Inputs [70]	22
Figure 2.3: Illustration of a Prediction with Unknown Inputs [70]	22
Figure 2.4: Diagram of Grey Model and ARX Model for Prediction	26
Figure 2.5: One-step Prediction Using Kalman Filter [51]	27
Figure 3.1: Assembly Structure of NCS-TT105 with a Sensing Element [94]	31
Figure 3.2: Schematic Diagram of a NCS-TT105 Sensor [94]	32
Figure 3.3: A Typical PROFIBUS-PA Architecture [99]	36
Figure 3.4: Block Structure of a PA devices Profile [100]	37
Figure 4.1: Parameter Estimation using ARX Model and EWRLS [51]	44
Figure 4.2: Relationship of Raw Data and Prediction in Grey Model	52
Figure 5.1: Implementation of Process Fault Detection in RTOS on NCS-TT105	66
Figure 5.2: Implementation of Variable Prediction in RTOS on NCS-TT105	73
Figure 5.3: Multitasking Implementation of Three Tasks in Nucleus RTOS	87
Figure 5.4: Timing Terminologies of a Task in an RTOS System	89
Figure 5.5: Task Flow of Multitasking Scheduling in RTOS	93

Figure 5.6: Program Context Refer to NU_Sleep() and Kernel Tick
Figure 5.7: Overview of the Multitasking Implementation in Nucleus RTOS101
Figure 6.1: Primary Water Loop Subsystem of the NPCTF
Figure 6.2: DigiVis HMI Graphic of NPCTF System105
Figure 6.3: Schematic Diagram of Validation System for SMART NCS-TT105106
Figure 6.4: Verification of Communication Channels of SMART NCS-TT105109
Figure 6.5: Diagram of Smart Functions Offline Validation110
Figure 6.6: Overview of Variables in MATLAB for Fault Detection Validation112
Figure 6.7: Residual Generation and Evaluation for Fault Alarm115
Figure 6.8: Overview of the Process Changes for Prediction Validation118
Figure 6.9: Offline T1 Prediction by the Grey Model Prediction
Figure 6.10: Offline (T2-T1) Prediction by Kalman Predictor and Multi-step Iteration 122
Figure 6.11: Offline T2 Prediction Validation for Combined Prediction123
Figure 6.12: Offline T2 Prediction Validation for Combined Shifted Prediction124
Figure 6.13: Diagram of Smart Functions Online Validation
Figure 6.14: The Method of Computing Execution Time of a Task
Figure 6.15: The Method of Computing Period Time of Three Tasks
Figure 6.16: Validation of Process Fault Detection in SMART NCS-TT105129
Figure 6.17: Validation Picture of Fault Detection Graphic in WinCC
Figure 6.18: Fault Detection Validation of Heater Tripping Fault

Figure 6.19: Fault Detection Validation of Cooling System Fault1	132
Figure 6.20: Fault Detection Validation of Pipeline Plug Fault1	133
Figure 6.21: Fault Detection Validation of F1 Loss Fault1	134
Figure 6.22: Fault Detection Validation of T1 Loss Fault1	135
Figure 6.23: Fault Detection Validation of Pipeline Leak Fault1	136
Figure 6.24: Effects of Alarm Indication by CUSUM1	137
Figure 6.25: Validation of Variable Prediction in SMART NCS-TT1051	139
Figure 6.26: Validation Picture of Prediction Graphic in WinCC1	140
Figure 6.27: T2 Prediction Validation with the Changes of Flow Rate F11	141
Figure 6.28: T2 Prediction Validation with the Changes of Heater Control Signal C21	142
Figure 6.29: T2 Prediction Validation with Heater Shutdown1	143
Figure 6.30: T2 Prediction Validation with Chiller Shutdown1	145
Figure B.1: Block Structure of a PA devices Profile [101]1	164
Figure B.2: Internal Diagram of a Temperature Transducer Block [102]1	165
Figure B.3: Internal Diagram of an Analog Input Function Block [102]1	165
Figure C.1: Task Flow of Multitasking Scheduling in Scenario-11	68
Figure C.2: Task Flow of Multitasking Scheduling in Scenario-21	70

List of Abbreviations

AR	Auto Regressive
ARMA	Autoregressive Moving Average
ARIMA	Autoregressive Integrated Moving Average
ARX	Auto-Regression with Exogenous Inputs
ARMAX	Auto Regressive Moving Average with Exogenous Inputs
ADC	Analog-to-Digital Converter
AGO	Accumulating Generator
ATC	Auto Target Compensation
APU	Application Processing Unit
BCU	Bidirectional Communication Unit
CUSUM	Cumulative Sum
DCS	Distributed Control System
DSP	Digital Signal Processor
EWRLS	Exponentially Weighted Recursive Least Squares
ELS	Extended Least Squares
EMF	Electromotive Force
FB	Function Block
FFT	Fast Fourier Transform
FISCO	Fieldbus Intrinsically Safe Concept

GCD	Greatest Common Divisor
GLR	Generalized Likelihood Ratio
GM	Grey Model
IAGO	Inverse Accumulating Generator
IC	Integrated Circuit
IIoT	Industrial Internet of Things
ISP	Intelligent Signal Processing
ISR	Interrupt Service Routine
IWSN	Industrial Wireless Sensor Network
LS	Least Squares
LPWAN	Low-Power Wide-Area Network
MAPE	Mean Absolute Percentage Error
MBP	Manchester Encoded and Bus Powered
MCU	Microcontroller Unit
MEMS	Micro-Electro-Mechanical System
NCS-TT105	Transmitter using as the implementation platform
NPCTF	Nuclear Power Control Test Facility
РВ	Physical Block
PCA	Principal Component Analysis
PI	PROFIBUS & PROFINET International Group

PROFIBUS-PA	An Industrial Fieldbus Technology
PSD	Power Spectral Density
PLS	Partial Least Square
PNO	PROFIBUS Nutzerorganisation
RDE	Riccati Difference Equation
RTD	Resistance Temperature Detector
RTOS	Real-Time Operating System
RLS	Recursive Least Squares
RMS	Rate Monotonic Scheduling
SEU	Sensing Elements Unit
SJF	Shortest Job First
SMART NCS-TT105	NCS-TT105 with Smart Functions
SoC	System-on-Chip
SPRT	Sequential Probability Ratio Test
SPU	Signal Processing Unit
SRT	Shortest Response Time
STM	Statistical Process Monitoring
ТВ	Transducer Block
TC	Thermocouple
UART	Universal Asynchronous Receiver/Transmitter

V&V	Verification and Validation
WPAN	Wireless Personal Area Networks
WSN	Wireless Sensor Network

Nomenclature

у	Measured system output
ŷ	Estimated system output
r	Residual between the measured output and estimated output
a_m	Parameters of system responses in ARX and ARMAX model
b_m	Parameters of system inputs in ARX and ARMAX model
C _m	Parameters of second inputs in ARX or noise in ARMAX model
p	The steps of prediction
y _p	The redicted response variable
R_{t}	Resistance at the temperature <i>t</i> .
R_0	Resistance at the temperature $t = 0^{\circ}C$
U	Input of the system used in ARX model
Y	Output of the system used in ARX model
U_{op}	Input value $U(k)$ on operating point value
Y_{op}	Output value $Y(k)$ on operating point value
и	The system inputs eliminated with operating point value
<i>Y</i> _u	The system response eliminated with operating point value
T_s	Sampling period of the algorithm in transmitter

d	Discrete dead-time of the process variables
λ_{avg}	Exponential forgetting factor for computing average value
$\lambda_{_{var}}$	Exponential forgetting factor for computing variance value
T_{1st}	Time constant of the first order system
$\hat{ heta}$	Estimated parameter vector
$\psi^{^{T}}$	Data vector including $y_u(k)$ and $u(k)$
$P_{_{W}}$	Inverse correlation matrix of EWRLS
$\lambda_{_{w}}$	Exponential forgetting factor of EWRLS
Ι	Identity matrix
m	Orders of EWRLS
е	Residual of EWRLS
r	Residual for the inputs of two-sided CUSUM
υ	Drift threshold of two-sided CUSUM
h	Alarm threshold of two-sided CUSUM
g ⁽¹⁾	Positive cumulation of two-sided CUSUM
g ⁽²⁾	Negative cumulation of two-sided CUSUM
$K_{_{v}}$	Ratio parameters for dynamic drift threshold
K_h	Ratio parameters for dynamic alarm threshold

X	Signal value, state in state-space equations
μ̂	Estimated mean value
<i>s</i> ²	Sample variance
$\lambda_{_{e}}$	Exponential forgetting factor for recursively computing mean and sample variance
k	Discrete time index
Α	Process matrix of State-space matrix
В	Control matrix of State-space matrix
С	Measurement matrix of State-space matrix
Ζ	Sensor measurements in state-space equations
W	Process noise in state-space equations
v	Measurement noise in state-space equations
Q	Process noise covariance
R	Measurement noise covariance
P^-	Covariance matrices of the estimation error in Kalman predictor
Κ	Kalman filter gain
P_{KP}	The steps of prediction via Kalman predictor and multi-step iteration
M _{KF}	The dimension of Kalman predictor
X _{KP1st}	One-step prediction of Kalman predictor

<i>x</i> ⁽⁰⁾	Raw data in grey model
$X^{(0)}$	A sequence of the raw data
<i>x</i> ⁽¹⁾	The value calculated by the first-order accumulating generator
$X^{(1)}$	A sequence of the first-order accumulating generator
$lpha^{(1)}x^{(0)}$	The value calculated by the first-order inverse accumulating generator
$lpha^{(1)}X^{(0)}$	A sequence of the first-order inverse accumulating generator
$Z^{(1)}$	Adjacent neighbor means of the sequence
n	The length of sequence in grey model
$\hat{ heta}_{_{GM11}}$	Parameters of grey model GM(1,1)
$\hat{ heta}_{_{GM21}}$	Parameters of grey model GM(2,1)
$\hat{x}^{(0)}$	Restored value of grey model
$\hat{x}_{p}^{(0)}$	Predicted value of grey model
$X_{p}^{(0)}$	Prediction sequence of grey model
$\gamma_{dco}(k)$	Development coefficient of grey model GM(1,1)
W_{GM}	Weight value of combined prediction of $GM(1,1)$ and $GM(2,1)$
<i>M</i> _{<i>G</i>11}	The length of $GM(1,1)$
<i>M</i> _{<i>G</i>21}	The length of $GM(2,1)$

Y_{KPp}	The prediction via Kalman predictor and multi-step iteration
P _{GM}	The steps of prediction via grey model
$\hat{x}_{GM11p}^{(0)}$	Predicted value of GM(1,1)
$\hat{x}_{GM21p}^{(0)}$	Predicted value of GM(2,1)
$\hat{x}_{GMComp}^{(0)}$	Predicted value of weighted between the prediction from $GM(1,1)$ and $GM(2,1)$
T_e	Execution time of a task in RTOS
T_p	Period time of a task in RTOS
T _r	Response time of a task in RTOS
T_d	Deadline of a task in RTOS
T _{int}	Serving interrupts time of a task in RTOS
T _{ever}	Event response time of a task in RTOS
T _{Tick}	Kernel tick time of RTOS
N _{Tick}	Number of kernel ticks
WL	Microprocessor workload for running tasks
T_1	Water temperature at inlet of the heater
T_2	Water temperature at outlet of the heater
ΔT	Water temperature difference between outlet and inlet of the heater

C2	Control signal	for the current	power to the	heater

F1 Flow rate signal in the primary water loop

Chapter 1

1 Introduction

It is well known that sensors are the essential devices in condition monitoring and control systems. A typical sensor consists of a sensing element and a transmitter. Nowadays, advanced functionalities are embedded into the transmitters which are combined with signal processing unit, data computing unit and communication unit, enabling the transmitters to be smart transmitters. Therefore, the sensing elements and a smart transmitter are combined as known a 'smart sensor' [1][2]. The major functions of mainstream smart sensors mostly focus on accurately measuring data and transmitting the measurement via network.

In industrial applications, data analysis functions in condition monitoring systems are often performed and are in the form of central monitoring stations in large plants such as power plants, chemical plants, and petrol plants [3][4]. These functions can collect interactive data to provide valuable information of equipment health status [5]. However, many small to medium sized systems such as standalone machinery systems often lack of conditions and funds to apply such large and costly central monitoring systems [5]. Therefore, it is imperative that such useful functions to be decentralized into field devices to achieve cost-effective and easy-to-use data analysis functions for serving unit systems.

To achieve cost-effective distributed intelligence in process control, numerous intelligent networked devices with ubiquitous networks are needed in industrial systems. Therefore, systems which combine network and intelligent devices so called Industrial Internet of Things (IIoT) emerge [6]. To enhance the capabilities of those field devices, the data analysis functions of condition monitoring can be the functionalities to be integrated. Hence, smart transmitters can be embedded with specific algorithms to flexibly achieve data analysis functions of condition monitoring. Meanwhile, since the integrations of these additional functions in smart transmitters do not need to modify the foot-print or add supplementary hardware, the decentralized condition monitor functions via smart sensors can be cost efficient.

1.1 Overview of Industrial Smart Sensors

1.1.1 Brief Review of Industrial Sensors

Sensor is an electronic device for resonding stimulus from physcial environment and transferring stimulus into signals or data [7][8]. A typical industrial sensor consisits of two parts: a sensing element and transmitter, which is shown in Figure 1.1 (a). The sensing element is used for sensing the physical environment, while the transmitter is used for signal conversion, filtering, and trasmitting the signals or data to orther systems. In comparison, a smart sensor is combined with sensing elements and a smart transmitter, which is shown in Figure 1.1 (b).



Figure 1.1: Overview of a Typical Industrial Sensor and Smart Sensor

The first commercial smart sensor was produced by Honeywell company in 1983 [9]. More sensors that fell into the smart category also emerged in the mid-1980s [10][11]. Up to date, smart sensors have been used in a wide range of measurement and monitoring applications such as temperature, pressure, flow, level, weight, and so on. A typical industrial smart sensor and its internal diagram is shown in Figure 1.2 [11]. An industrial smart transmitter typically includes analog-to-digital converters (ADCs), CPU, EEPROM and Fieldbus communication functions.



Figure 1.2: Siemens SITRAN TF Functions Diagram [12]

Mainstream smart transmitter in industrial verticals are mostly designed to focus on features including accurate measurement, self-diagnostic such as sensing elements break or short-circuit, self-calibration, self-configuration, and wired or wireless communications [13][14][15]. Some well-known brands are Rosemount, Siemens, Honeywell, Endress+Hauser, KROHN, and Yokogawa. For instance, Siemens SITRANS TF [12] series sensors offer high-accuracy measurement with less than 0.05 % absolute accuracy, as well as self-diagnostic, linearization, calibration functions, and Profibus-PA and Foundation Fieldbus communication. Honeywell SmartLine ST800 pressure sensors can provide up to 0.0375% accuracy for static pressure measurement through internal temperature compensation [16]. As shown above, the essential features of mainstream industrial smart transmitters are high-accuracy measurement, easy set-up and maintenance, and communication.

With the technological advances and pull of application demands, innovative transmitters with on-board signal processing and data analysis functions have emerged in recent years [17]. One pioneer in this field is the OPTISWIRL 4070C flowmeter [18] offered by KROHN, which is integrated with the Intelligent Signal Processing (ISP) function. The ISP can eliminate unwanted noise and external perturbations for measurement via its signal processing unit, so that users can read the stable flow results of either liquids or vapors without any impacts from fluctuating pressures and temperatures. Another example of recent innovation in transmitter design is the Rosemount 3051S [19] pressure transmitter. Its deviation calculation algorithm, drawn from integrated Statistical Process

Monitoring (SPM) functions, can analysis data for indicating abnormal process status such as cavitation and flame instability.

The above examples demonstrate signal processing and analysis functions which are integrated in smart transmitters. Even through there are only handful smart transmitters with built-in functions in market, they shows a trend that smart transmitters provide not only accurate measurement, but also signal processing and data analysis functions.

1.1.2 Composition of Smart Transmitter

To integrate data analysis functions into smart transmitter, two major components are required, which are the hardware and software system of smart transmitters, respectively.

A. Hardware components in a smart transmitter

The hardware components of a smart transmitter could be composed with three main parts, including signal processing unit (SPU), application processing unit (APU), and bidirectional communication unit (BCU), as shown in Figure 1.3. Benefiting from the development of embedded systems and integrated circuits technology, those three parts and sensing elements unit (SEU) can be integrated into one chip, or grouped into different components based on different application scenarios. In addition, some auxiliary parts such as power units, communication antenna, electric isolation circuits, and display units are also basic components in a smart transmitter.



Smart Sensor

Figure 1.3: Smart Transmitter Architecture

Signal Processing Unit: SPUs play the role of translating analog signals into digital signals. Components such as signal conditioners, analog-to-digital converters (ADCs), and sometimes signal processing processors are in this unit. SPUs connect with SEUs and transfer SEU signals into digital output. The digital output so called raw data are then used for further processing. Moreover, if signal processing processors are involved, the signal processing functions, such as signal compensation, calibration, digital filters, drift testing, and fast Fourier transform (FFT), can be integrated in this unit.

Bidirectional Communication Unit: BCUs serve as bridges, exchanging data for smart sensors with outside network. BCUs allow the smart transmitters to exchange data with other systems or devices, such as upper lever controllers, actuators, or other sensors. Applying standard communication protocols in BCUs is an efficient approach to achieve unification on communication and accelerates the progress of smart sensors for various networking applications.

Application Processing Units: APUs are the pools for organizing and running embedded algorithms for different applications. To execute complex algrithms, powerful MCUs need to be assigned to APUs. Furthermore, to achieve multiple tasks and real-time operation, RTOS needs to be applied for a microcontroller. An MCU with a RTOS is an excellent combination for APU. This allows APU to be integrated with algorithms to realize data analysis functions. Therefore, the algorithms and functions in this thesis are implemented in this unit.

B. RTOS for smart transmitters

To process various smart functions within a deterministic response time, RTOS is commonly involved in smart transmitters [20]. A task, which includes an application program code and works independently from other tasks, is dispatched by the RTOS kernel's scheduler [20]. The preemptive kernel, which is the core of RTOS, is used for scheduling and dispatching multiple tasks in real-time. The preemptive kernel not only allows multiple independent tasks to share the processor resources but also guarantees that tasks with high priority are prioritized [21]. To realize different application algorithms, programs can be grouped and allocated to each of task. Each task can be designated as an independent loop, memory resources, and priority. The algorithms and functions in the tasks can be organized flexibly and scheduled in real-time.

1.2 Shortcomings of Existing Smart Sensors and Potential Solutions

Smart sensor has become a fashionable word in contemporary technological among industrial circles. However, there are still some shortcomings in the existing smart transmitters.

1.2.1 Shortcomings of Existing Smart Transmitters

A. Lack of data analysis functions

So far, there are very few analysis functions in existing smart transmitters. The computing processors in the conventional smart transmitters are used to compute calibration and linearization for accurate measurement. Communication is responsible for transmitting measured data to other systems. Even though powerful computing processors are embedded, the main task of conventional smart transmitters mainly focuses on measurement and data transmission. By leveraging powerful computing capacity, smart transmitters can be integrated with data analysis functions beyond providing measurement.

B. Lack of information extracted from data

Another shortcoming of conventional smart transmitters is that, even though diagnostic functions are integrated, the efforts are focused on easy set-up or maintenance of sensor itself. However, the information which could have been extracted from measurement data is rarely extracted to provide further benefit [22]. For instance, the valuable information includes detected faults in process, foreseeing the abnormal process changes, and guiding preventive maintenance. Different from the conventional self-diagnostic capabilities, the information of the process system could support users to gain the insight on the process system, and take actions for avoiding severe accidents in process.

1.2.2 A New Generation of Smart Sensors

As previously discussed, integrating functions to provide valuable information is the key feature to potentially enhance the existing smart transmitters. The smart sensors combined with smart transmitters can be described as following:

A smart sensor is a multi-capabilities, algorithms embedded device in which measuring, analyzing, collaborating, and self-governing can be integrated to convert physical variables into accurate data, process data and extract valuable information, and utilize comprehensive information to provide decision-making suggestions.

From above concepts, the embedded algorithms play key roles to allow smart transmitters to extract valuable information from collected data. Moreover, the platform of the embedded algorithms can be realized by embedded RTOS to enable customized functions to be integrated into smart transmitters.

1.2.3 Potential Solutions through Integrating Data Analysis

To overcome the shortcomings of conventional smart transmitters, smart transmitters can be embedded algorithms to realize data analysis functions. The improved smart transmitters should extract valuable information from the collected data instead of just sending measurement.

A. Realizing data analysis for process condition monitoring

The data analytical functions for analyzing the status of industrial process and facilities are commonly realized by condition monitoring systems. The typical form of condition monitoring system is a central management software system. For instance, Emerson AMS Suite provides a software solution named Essential Asset Monitoring [23] to analyze collected data to alert operators for any process faults, such as fouling and plugging problems in heat exchanger systems, or pump shutdown in cooling systems. Siemens Condition Monitoring Library [24] in its Distributed Control System (DCS) PCS 7 systems can analyze the health status of centrifugal pump damage, valve wear, process steady state, and any pressure loss. Another example is Process and Equipment Monitoring [25] system named Uniformance Asset Sentinel offered by Honeywell. This system is integrated with preprogrammed first-principles models, which include a pump, compressor, heat exchanger, and turbine models, can be used for predicting the status in either the process or the equipment.

The widespread applications of these systems demonstrate the usefulness of the data analysis functions in condition monitoring systems are used for industrial process systems. However, to employ and operate such central monitoring systems, the users need to design the monitoring systems, prepare central rooms and computers for the software, and spend time to attend the professional trainings for learning the operations.

Generally, small to medium sized systems are often lack of conditions and funds to apply such large and high-cost centralized monitoring systems [5]. Realizing data analysis functions at sensor level rather than employing centralized monitoring systems is costeffective way to equip analytical systems for small and medium sized systems. Since smart sensors are broadly used small and medium sized systems, integrating data analysis functions into existing smart transmitters without adding extra cost, allows the status of facilities monitored and the process data can be analyzed in a cost-effective way.

B. Providing information of process fault detection and variable prediction

Faults detection information and variable prediction information are two kinds of process information which are provided by process fault detection function and variable prediction functions respectively in condition monitoring systems. The process fault detection function can monitor the health of the process and alert faults. The detected faults information in incipient period means that the faults are detected before they developed into deterioration or failure. The incipient fault detection can be used to prevent serious failures in process or esstential equipment and to avoid deterioration. Equally important, the prediction function can predict process variables to foresee the future changes of the vital variables. The predicted information allows controllers and engineering teams to take early actions rather than simply passively waiting for something to go worse. Therefore, the information of process fault detection and variable prediction are valuable for industrial users. The implementation of process fault detection and variable prediction functions in smart transmitters can enable smart sensors to provide information of process faults and variable prediction. Smart transmitters can be embedded with algorithms to achieve such functions to overcome the shortcomings of existing smart transmitters.

1.3 Research Objectives, Methodologies, and Scope

1.3.1 Objectives

To enable smart transmitters to perform fault detection and variable prediction functions, the objectives of this thesis include the following two aspects:

A. Investigating the feasibility of adding smart functions in smart transmitters

The methods which can realize the smart functions of process fault detection and variable prediction and suitable to be applied into smart transmitters are investigated. Furthermore, the hardware capability of the MCU in the transmitter implementation platform and its RTOS software system are studied. Based on the investigated methods and studied capability of the transmitter, the specific algorithms for realized process fault detection and variable prediction are designed.

B. Implementing the algorithms of the smart functions into a smart transmitter

The algorithms which can realize fault detection and variable prediction are implemented in MCU of the transmitter. The added smart functions enable the implemented smart transmitter to extract the process information of fault detection and variable prediction from collected data, using its the underutilized computational resources without adding extra expense.

1.3.2 Methodologies

The core research tasks include three main steps: studying algorithms of process fault detection and variable prediction, implementing two group algorithms and designing RTOS multitasking configuration, and validating the enhanced smart sensor in a semi-practical test facility.

1) The feasibility of integrated functions and embedded algorithms for realizing fault detection and variable prediction in smart transmitter are investigated. The capabilities and specifications of the implementation transmitter are studied.

2) The algorithms for realizing fault detection and variable prediction are implemented in MCU of the transmitter. The algorithms are organized in independent tasks and scheduled in real-time by RTOS of the transmitter. The RTOS multitasking configuration, which includes the consideration of task priority, event response time, and processor workload, are discussed and implemented.

3) The enhanced smart sensor are verified in a semi-practical test facility. The designed process fault detection and variable prediction functions are validated both in offline and online test. The multitasking design of three tasks in RTOS of smart transmitters are also validated.

1.3.3 Scope

The scope of this thesis mainly focuses on integrating fault detection and variable prediction functions into the smart transmitters of the smart sensors. The procedures of investigating feasible methods and algorithms for realizing the targeted functions, implementation of the algorithms, and V&V test, are three main tasks in this research, which provide a demonstration for further functions and algorithms to be integrated into transmitters. Therefore, not all the data analysis functions of condition monitoring systems are involved. The faults which are demonstrated in the test environment for the validation are only the typical faults, and will not cover all the faults in the test facility.

Furthermore, the designed functions are realized in software layer, and the embedded algorithms only use the underutilized computational resources of MCU in the existing transmitters. Therefore, some technologies in the smart sensors, such as sensing elements technologies, digital signal processing, and communication protocols of the smart sensor, are not the main focuses in this research.

1.4 Contributions of the Thesis

This thesis has made three major contributions:

A. Investigating the feasible methods and algorithms to embed into smart transmitter

Based on the capabilities and specification of the transmitters, the feasible methods and algorithms of process fault detection and variable prediction are investigated. The algorithms are embedded into a commercial transmitter and scheduled by its RTOS successfully.

B. Improving capabilities of the existing sensors with data analysis capabilities

The integrated smart functions improve the capabilities of the existing transmitter. The data analysis functions which include process fault detection and variable prediction enable smart sensors to analyze process data locally, allowing the small and middle sized system can be economically deployed with process fault detection and variable prediction.

C. Demonstrating the feasibility of integrated smart functions into existing transmitter

The successful implementation of the feasible methods and algorithms of fault detection and variable prediction functions demonstrates the procedures of the implementation of the smart functions in a transmitter hardware platform. The procedure such as methods survey, algorithms implementation, and V&V test can be referenced for further functions integration.

1.5 Organization of the Thesis

The thesis consists of seven chapters. The remainder six chapters are organized as follows:

The progress of smart sensors are reviewed, while the existing methods for targeted smart functions including fault detection and prediction are surveyed in Chapter 2. The hardware structure, RTOS and corresponding standards of the implementation platform which is a commercial industrial transmitter NCS-TT105 are introduced in Chapter 3. In

Chapter 4, the specifications of the designed functions are proposed, and algorithms of process fault detection and variable prediction are investigated. Furthermore, implementation of algorithms and multitasking design are discussed in Chapter 5. In Chapter 6, the verification and validation of smart functions in smart transmitter are carried out on a physical test environment. Finally, summary, conclusions, and future work are presented in Chapter 7.

Chapter 2

2 Literature Review

The literature of smart sensors can be traced back to thirty years [1][2]. The development of smart sensors over this time can be classified into following four phases, i.e. Data acquisition, integration of hardware and software, communication, and information. As the software system of the smart transmitters, RTOS is the important compositions in smart sensors for organization various programs. Furthermore, to extract information from measurement data in smart sensors, algorithms which is embedded in transmitters paly the key role. Therefore, it is necessary to review the appropriated algorithms to implement process fault detection and variable prediction functions.

2.1 Review of Existing Smart Sensors

A. Data acquisition from physical world

In order to provide accurate measurement, the smart sensors are integrated with microprocessors in their transmitters to realize digital signal processing and data computing. According to Schödel [26], smart sensors are described as a microprocessor device with filtering and other signal processing and data computing functions. Comparing with the traditional analog circuit design, the digital microprocessors such as MCUs and Digital Signal Processors (DSP) are integrated into smart sensors [1]. MCUs are commonly used for data processing such as calibration, linearization, and compensation for accurate measurement [27]. The typical MCU chips are Texas Instruments MSP430 [28], and LPC4000 series from NXP [29]. DSP is ultilized for signal processing in smart sensors [30], such as FIR/IIR filtering, and Fourier transforms.

B. Integration of hardware and software

With the development of semiconductor technologies, the sensing elements and smart transmitters can be potentially integrated into one chip [31][32], which makes the smart sensor to become a system-in-package device [33]. Especially, the sensing elements for sensing thermal, gravity, acceleration and so on, can be combined with smart transmitter
and integrated into a single processor chip [11][34][35]. The high integration allows sensors to be more compact in size, multiple sensing integrations, and lower power consumption.

Typical hardware and software integration technologies used in smart sensors are Micro-Electro-Mechanical Systems (MEMS) and System-on-Chips (SoCs) [33][36]. MEMS uses microfabrication technology to make miniaturized mechanical and electromechanical elements in electronics and microscopic devices [37]. Owing to its microembodiment, MEMS technology can integrate a large number of sensing elements into one chip [33]. SoC enables signal conditioning parts and other circuits to be integrated together. SoC is integrated circuit (IC) system which is designed for integrating microsystems and multiple technologies, such as CMOS, MEMS, microprocessors, RT transceivers [36] that allow digital circuits, analog circuits, memories, ADCs, and microprocessors to be integrated into one chip and enable smart transmitter as a small foot-print[38][39].

C. Data transfer via communication

Thanks to the digitization and integration development in phase 1 and phase 2, smart sensor have become powerful sensing and measurement devices. However, the conventional analog transmission such as 4-20mA or 0-10v remains as bottlenecks for allowing smart sensors to exchange more data with other systems. As sensors combining with bus interface [2] and communicating with host systems [40], network functions become the essential capabilities of smart sensors. With wired and wireless network technologies, transmission barriers of smart sensors have been broken through.

There are many communication methods to achieve network functions. From wired network angle, Fieldbus which is followed by IEC 61158 specifications is particularly popular, such as PROFIBUS, PROFIBUS-PA, PROFINET, EtherCAT, Foundation Fieldbus [41]. From wireless network angle, Industrial Wireless Sensor Networks (IWSNs) and Wireless Sensor Networks (WSNs) provide many attractive features to the smart sensors [42][43][44][45]. To achieve transmission of large amounts of data, communication has been the necessary functions in smart sensors.

D. From data to information

Benefited from the rapid development of sensing, microelectronics, and communication technologies, smart sensors have become essential devices for collecting rich data in various networks. However, a challenge of rich data but poor information has been found in many industries. The valuable information allows people, control systems to make effective decisions, and to be used as reference for preventing accidents, foreseeing the problem, improving productivity [46]. Therefore, extracting valuable information from the collected data in smart sensors is imperative.

In order to extract valuable information from data, the embedded algorithms play the key roles. In early stages, smart sensors are incorporated with dedicated signal processing algorithms [47], such as power spectral density(PSD) and FFT, to provide frequency analysis information. Further progress on the development and implementation of fault detection and diagnosis [48], and artificial neural network functions [49] have allowed smart sensors to be information providers. For instance, leveraging image identification algorithms, smart cameras BOA [50] can identify the broken bottles in production line.

In the industrial field, the process information of the process faults and the future changes of the essential variable is valuable for users. The detected faults information in the process systems can be used to guide the planned maintenance, prevent deterioration, and to avoid serious accidents and failure in the process system. The predicted information can be used to guide timely preventive actions rather than simply passively waiting for system to go worse. To realize fault detection and prediction information to be provided in field level, process fault detection and variable prediction functions can be integrated into smart transmitters as smart functions in smart sensors. Therefore, the appropriated methods and algorithms to implement fault detection and variable prediction functions in smart transmitters are reviewed in the later sections.

2.2 Existing Fault Detection Techniques

An abnormal status, or fault, is defined as an "unpermitted deviation of at least one characteristic property or parameter of the system from normal or acceptable condition"

[3]. Faults may cause a decline in or even total loss of the designed capability in process systems. Unmonitored faults pose a significant risk to the safety of a plant or a machine system, and can lead to serious consequences for industrial production. Fault detection methods which can be used to extract faults information from the collected data of sensors are reviewed. Moreover, as a smart functions which is integrated into smart transmitter, the selected fault detection method must be applicable for embedded system. The computational complexity and operability of the algorithms should be feasible to be used in smart transmitter.

2.2.1 Classification of Fault Detection Methods

Many different approaches to monitor and detect imminent faults have been developed and discussed over the last thirty years [51][52][53]. Fault detection methods can be classified into two main groups: model-free methods and model-based methods [54][55]. An overview table of different fault detection methods is shown in Table 2.1.

Classification of Fault Detection Methods			
Model-based	Model-free		
	Signal-based	Data-driven	
Parameter estimation	Correlation	Artificial neural networks (ANN)	
Observers	Time-Frequency analysis	Principal component analysis (PCA)	
Parity equations	Spectrum analysis	Partial least squares (PLS)	
Kalman filter	Wavelet analysis	Multivariate state estimate technique (MSET)	

Table 2.1: Classification of Fault Detection Methods [54][55]

Model-free methods can be further broken down into signal-based methods and datadriven methods [55]. Neither of these methods utilizes a mathematical model of the objects. Signal-based methods mostly use signal spectrum analysis tool and are normally applied for rotating machine fault detection via analysis of vibration and noise. This method requires analytical devices with high-frequency capabilities [3], which means that the transmitters are needed to be configured with high-frequency data acquisition hardware. Data-driven methods use multivariate statistical methods via plenty of historical data and complex matrix computing. For instance, the typical data-driven methods are Principal component analysis (PCA) and Partial least squares (PLS) [56]. PCA and PLS methods are usually applied in central supervision systems [56][57]. Although model-free fault detection methods do not need the prior knowledge of mathematical model, due to the requirements of high computational complexity and large capacity storage space for the long-term data archiving, this kind of methods is hard to be broadly applied in transmitter devices.

Model-based methods need to use explicit mathematical models of the monitored process to represent the relationship between manipulated variables and controlled variable [3]. model-based fault detection methods have been tested and verified in many industrial applications such as actuators, sensors, machines and plants [58][59]. A diagram of this framework is shown in Figure 2.1 [52]. The inputs and outputs of a system are the data source of Model-based methods, allowing model-based methods to be feasible for implementation in the devices [3].



Figure 2.1: Model-based Fault Detection Framework [52]

Besides the models and their parameters, another two important steps in a model-based fault detection framework are residual generation and residual evaluation.

Residual generation is produced by the different model-based methods such as parameter estimation, observers, and parity equations [51][57], while residual evaluation involves making decisions on fault alarms using thresholds or statistical tools [60]. Based on the different application scenarios, the different combination of residual generation and residual evaluation methods could be applied. Since the model-based fault detection methods are suitable to be implemented in devices, therefore, the model-based methods are selected to be used in the smart transmitters.

2.2.2 Residual Generation by Model-based Methods

Residuals represent the deviation between the observed variables and the analytical output values of the mathematical model [60]. As illustrated in Figure 2.1, the residual is generated from a residual generation block. One straightforward method for calculating residuals is the difference between the measured output y(t) from detected system output and an estimated output $\hat{y}(t)$ from the system model, as

$$r(t) = y(t) - \hat{y}(t)$$
 (2.1)

The three major model based methods to generate residuals via estimating output $\hat{y}(t)$ are parameter estimation, observers, and parity equations [53][54][55][61]. Parameter estimation method is used to determine the unknown parameters of a process model, using dynamic excitation inputs and outputs of the process [51][53]. The structure of the model must be determined firstly, while the parameters can be estimated by this method. Through the model and its parameters, the output y(t) can be estimated via exciting inputs. Typical observer methods are Luenberger observers and Kalman filters [51][54]. The observer methods are suitable for observing states in state space matrix model of machinery or process system. The estimated output $\hat{y}(t)$ can be derived by observed states. Parity equation method is used for exploiting system measurements and comparing the consistency between the mathematical parity equations and the parallel system. The mathematical system model can be turned into different forms to generate output errors, equation errors, or input errors for detecting system additive faults, such as sensor bias or actuator sticking [51].

As shown above, both observer and parity equation methods rely on known mathematical matrix and equations. However, the prior matrix and equations are hard to derive in some systems and are scarcely possible to be preprogrammed with amount of models into smart sensors for various process systems, allowing the methods which require specific mathematical matrix and equations are hardly to be used in smart transmitter. Contrarily, parameter estimation methods can estimate the parameters of the model. Only structure of the models need to be determined. The determination of model structure is easier than the determination of both structure and parameters of model. Leveraging parameter estimation, the modeling system can learn the parameters of model with the determined structure of the model. This strategy without intervention allows parameters estimation methods to be broadly used, and enabling parameter estimation methods are selected for integrating into smart transmitters.

Among parameter estimation methods, the Least Squares (LS) and its recursive algorithms are commonly utilized [51]. The basic LS algorithms is batch processing algorithm, which need to be recomputed with all the data in matrix and matrix inverse computation [62]. Comparing with LS, Recursive Least Squares (RLS) reduces the computational effort. Leveraging recursive strategy, RLS has been widely used for system identification and adaptive filter domain [63][64]. With fast convergence performance and recursive as well as adaptive features, RLS is broadly used for online parameter estimation both in open-loop and closed-loop [63]. However, the original RLS is suitable for estimating the parameters which are stable or time invariant [51]. To estimate the parameters in a time-varying system, the alternative algorithms named Exponentially Weighted Recursive Least Squares (EWRLS) with a forgetting factor can be used [51]. Since the process system working around operating point can be described as linear dynamic models [3], this algorithm is suitable for identifying parameters in process system.

Smart transmitters are the embedded system which is a system of unattended, and limited computation. The adaptive and recursive algorithms of parameter estimation can adaptively "learn" the parameters of linear process models and enable computational

efficiently. Therefore, to online estimate the parameters of process system working around operating point, EWRLS are suitable to be used in smart transmitters. The more details of algorithms to implement recursive parameters estimation are discussed in section 4.2.

2.2.3 Residual Evaluation for Fault Detection

After the residuals are generated, the residual evaluation is used to examine the faults occurrence. Residual evaluation is a procedure for using residual and validation technologies to make decisions about the faults detection [53]. The residual evaluation manners can be grouped into two paradigms which are threshold approaches and statistical tools, respectively [65][60].

The basic idea of threshold approaches is to validate residual deviation from normal values [66]. Once sensors, actuators, or equipment faults occur, residuals will deviate from zero. If the amplitude of residuals changes is larger than the threshold, the evaluation test will consider faults occurred. In practical applications, the residuals are corrupted by unknown disturbances, process noises, and uncertainties in the system model. Leveraging threshold method alone could miss the false or generate wrong fault alerts. Therefore, statistical tools methods of residuals evaluation could be involved to overcome the influences from disturbances and noises when evaluating residuals changes.

Three well-known statistical tools for residuals change detection are Sequential Probability Ratio Test (SPRT), Generalized Likelihood Ratio (GLR) Test, and Cumulative Sum (CUSUM) test [60][66]. SPRT uses recursive method to compute the cumulate sum of the log-likelihood ratio can detect faults in short time[66]. GLR has good performance on detecting additive and component faults[60]. However, the original GLR cannot compute recursively [60][67]. Comparatively, Cumulative Sum (CUSUM) [66][67] which can be seen as repeated SPRT [60] uses recursive cumulative results to compare with decision bound. It has been broadly investigated for online and offline change detection [66][68]. Since recursive methods and repeated calculation are suitable for online fault detection in devices, CUSUM is suitable to be used as statistical tool of residual changes evaluation for fault detection in smart transmitters.

In summary, the parameter estimation and CUSUM methods can be considered to be combined as fault detection functions in smart sensors. In practice, the industrial process system model is often unknown or hard to be derived in most cases. The parameter estimation method is suitable for industrial users in such situations. Adaptive algorithms and recursive computing solutions are suitable for online parameter estimation in smart sensors. Moreover, considering the performance and computational complexity, the twoside CUSUM as residual evaluation tool can be employed. A detailed algorithms and implementation for fault detection function are introduced in Chapter 4 and 5.

2.3 Existing Techniques for Prediction of System Responses

To predict the responses of a system, the system model play significantly roles [69][70]. Besides model, the responses of the system which are needed to be predicted should be measured or known, while the leading forces (inputs) of the system can be known or unknown [70]. Based on different scenarios of leading forces which are either known or unknown, the different prediction models and prediction methods can be applied. Moreover, as a smart functions which is integrated into smart transmitter, the selected prediction algorithms must be applicable for embedded system and should be feasible to be used in smart transmitter.

2.3.1 Models with and without Exogenous Variables

Industrial process systems can be classified as input-implicit systems and input-explicit systems. In an input-implicit system, the leading forces for the response of a system are either undetermined or unknown. Such systems commonly show as strong inertia characteristics, such as furnace temperature, reservoir level, and water quality. Since the exogenous variables cannot be determined or unknown in the input-implicit system, the models without exogenous variables can be used to describe the input-implicit systems. While in an input-explicit system, the leading forces for the response of a system are known. For instance, the heater is powered by the observed current, and the speed of the

motor is controlled by manipulated voltage. Involving exogenous inputs, prediction by the models with exogenous variables can be proactive and accurate. Therefore, the models with exogenous variables are suitable to describe the input-explicit system.

The diagrams to illustrate the prediction system considering without known inputs and with known inputs are shown in Figure 2.2 and Figure 2.3, respectively.



Figure 2.2: Illustration of a Prediction with Known Inputs [70]



Figure 2.3: Illustration of a Prediction with Unknown Inputs [70]

A. Models without exogenous variables

In models without exogenous variables, only observed response of the system is used. The common used models without exogenous variables are Auto Regressive (AR) model [71], Autoregressive moving average (ARMA) model [69], Autoregressive integrated moving average (ARIMA) model [69][72], and grey model [73]. AR model is often used as linear prediction models for the stochastic process in statistical and signal processing field [71]. Although the AR is simple and widely used, if the noise and disturbances are unaccounted, the prediction derived from AR model may be less accurate [69][71]. ARMA and ARIMA shows better performance since the noise structure and integrated noise structure are considered in two models. However, the parameters estimation for high orders coefficient of these two models is high computational complexity [69][71]. The industrial process includes obvious disturbance, while huge computing tasks are not suitable to be embedded into smart transmitters. Therefore, the model which have robust performance and moderate computational complexity are suitable to be used in smart transmitters for implementing prediction function.

Grey model shows good and robust performance in uncertainty and noise condition [74]. grey model are proposed from grey system theory [73]. This methodology focuses on modeling the system with small data samples, and extracting useful information from uncertain systems [75][76]. The typical popular GM(1,1) model which is used for modeling monotonic data series only needs two coefficients to be estimated, whereas GM(2,1) which is used for modeling none monotonic data series needs three coefficients to be estimated. Since the estimated coefficients only have two or three matrices, the straightforward Least Square (LS) parameter estimation can be involved [74]. Leveraging these two model and using n length of data series, the data series can be modeled easily, and the further ahead steps of data can be predicted iteratively. Using grey model, the performance and computational complexity can be balanced. Therefore, GM(1,1) and GM(2,1) models can be the model without exogenous variables for implementing prediction in smart transmitters for the input-implicit process.

B. Models with exogenous variables

Different from models without exogenous variables, the model with exogenous variables involved the leading forces (inputs) of system response. Considering the impacts of system inputs and known output response, the prediction can be proactive and accurate. According to [69][77][63], the Auto Regression with exogenous inputs (ARX) model and Auto Regressive Moving Average with exogenous inputs (ARMAX) are well-known models with exogenous variables. Involved with inputs U(k) and system response y(k), ARX model can be described as equation (2.2) in which the parameters a_m and b_m with morder can be estimated via parameter estimation method, i.e., LS or RLS. Comparing with ARX, ARMAX which is shown in the equation (2.3) is added with moving average. The parameters a_m , b_m and c_m with m order can be estimated via Extended Least Squares (ELS) method [63].

$$y(k) + a_1 y(k-1) + \dots + a_m y(k-m)$$

= $b_1 U(k-1) + \dots + b_m U(k-m) + n(k)$ (2.2)

$$y(k) + a_1 y(k-1) + \dots + a_m y(k-m)$$

= $b_1 U(k-1) + \dots + b_m U(k-m) + n(k) + c_1 n(k-1) + \dots + c_m n(k-m)$ (2.3)

where n(k) is assumend to be white noise sequence.

Even through ARMAX can be more precise to describe a dynamic system compared to ARX [78], the computation for parameters estimation are more complicated. On the contrary, because of the simplicity of the parameters estimation for ARX, this model is often used for industrial system identification [79]. Considering the performance and complexity of ARX and ARMAX, ARX can balance the precise and complexity for modeling a system, and suitable to be applied in embedded systems. Therefore, ARX is suitable to be used in smart transmitters to model the process system when exogenous variables should be considered in the process model.

2.3.2 Model-based Prediction Methods

Once model is determined, with the known system inputs and system response outputs, the response of the system can be predicted. Leveraging the grey model, the known output response can be used to predict the response of the input-implicit process, while utilizing ARX model, the known inputs and output response of the system can be used together to predict the response of the input-explicit process.

To achieve multi-step prediction, the one-step prediction and multi-step iterative computation are commonly combined together. During the multi-step iteration, if the prediction error exist in every prediction step, the error in multi-step prediction results is cumulated. This will result in lower prediction accuracy for further steps ahead prediction. To accurately predict variables for multi-step ahead, the effective solutions to reduce the effects of noise and disturbance in multi-step prediction need to be selected. Otherwise, the noise and error will be amplified via multi-step ahead iteration. Considering the robust performance of grey model in uncertainty and noise condition, GM(1,1) and GM(2,1) are used for the multi-step iterative prediction directly. Comparatively, since there is no disturbance term described in ARX model[77], the error in the initial iterative prediction step will be accumulated after multi-step iterative prediction if using ARX directly. To reduce the error in the initial iterative prediction, Kalman filter [80] is involved as an observer for one-step prediction. The states observed and predicted by Kalman filter will be used for multi-step ahead prediction. The further step prediction based on the predicted states from Kalman filter can be more accurate comparing with using ARX directly in a noisy process system. The diagram of two groups of prediction methods is shown in Figure 2.4.



Figure 2.4: Diagram of Grey Model and ARX Model for Prediction

A. Kalman filter for ARX one-step prediction

Kalman filter is a well-known estimator for solving linear-quadratic problem and has been extensively used in the engineering field [80]. The property of Kalman filter is a minimum mean-square estimator of the state of a determined linear state-space model in a Gaussian environment [62]. A set of recursive discrete-time equations enables this method to be computed very efficiently. The diagram of one-step states predictor [62] [81] derived from Kalman filter is shown in Figure 2.5. This one-step states predictor from Kalman filter is called Kalman one-step predictor in this thesis.

To use Kalman one-step predictor, the state-space model must be determined firstly. After the parameters of ARX model are estimated, the ARX model can be transferred into the state-space model using multiple inputs companion-form [82] for Kalman predictor. Through this combination, states which represent the response of the system can be observed and predicted [80]. This Kalman predictor can be used for accurately predicting one-step states prediction for dynamic systems perturbed by white noise. Leveraging the results from Kalman one-step predictor, the further steps prediction of states which represents the response of system can be achieved accurately.



Figure 2.5: One-step Prediction Using Kalman Filter [51]

B. Repeat iteration for multi-step ahead prediction

To predict multi-step ahead response of the system, the one-step ahead prediction can be derived combined with the observed input and output variables of the system firstly. Then using predicted value from the last step, the multi-steps ahead prediction can be calculated iteratively. For instance, using ARX model and *m* numbers of past response variable, after *p* times of iteration, *p*-step ahead predicted response variable $y_p(k+p)$ could be derived interactively using equation (2.4).

$$y(k) = -a_{1}y(k-1) \dots - a_{m+1}y(k-m+1) - a_{m}y(k-m) + \vec{b} \cdot \vec{U}$$

$$y_{p}(k+1) = -a_{1}y(k) - a_{2}y(k-1) \dots - a_{m}y(k-m+1) + \vec{b} \cdot \vec{U}$$

$$\vdots \qquad \vdots \qquad \vdots$$

$$y_{p}(k+p) = -a_{1}y_{p}(k-1+p) \dots - a_{m}y(k-m+p) + \vec{b} \cdot \vec{U}$$
(2.4)

In summary, to predict the response of an input-implicit process, grey model can be used to construct a model without exogenous variables, and repeat iteration method can be utilized for multi-step ahead prediction. To predict the response of an input-explicit process, ARX is suitable to describe a model with an exogenous variables, and Kalman predictor and repeat iteration method can be combined for multi-step ahead prediction.

2.4 Real-time Operating Systems in Transmitters

RTOS has been used as reliable software system for real-time processing multiple tasks in embedded systems [83][84]. RTOS is also broadly used in the transmitter of the sensors [2], such as wireless sensors [85], vibration sensors [86]. The mainstream RTOS includes VxWorks [87], Nucleus RTOS [88][89], ARM mbed [90], MicroC/OS [91], FreeRTOS [92] and so on. Leveraging RTOS, the multiple tasks can be scheduled by the kernel in real-time. This will enable users to focus on the development of function instead of multitasking scheduling. The RTOSs allow the development of functions in embedded system to be more efficient and reliable.

To realize multitasks scheduling for achieving task period time and response deadlines, the preemptive schedulers are commonly employed in RTOS. The different preemptive scheduling policies can be realized in the RTOS by different parameters setting.

The priority-based preemptive static scheduling policies are the main scheduling policies used in RTOS [87]. Using the preemptive static scheduling policies, the task which is assigned with highest static priority and positioned in the ready queue is to be prioritly executed. Therefore, the assignment of static priority for the tasks plays the key role in priority-based preemptive static scheduling policies. Some popular static priority assignment schemes under the umbrella of priority-based preemptive static scheduling (RMS), Shortest Job First (SJF) scheduling, Shortest Response Time (SRT) scheduling [87]. Considering RMS is optimum scheduling solutions among the others and is widely applied by RTOS designers [87], the RMS is selected as the scheduling policy to configure RTOS of smart transmitters.

2.5 Summary

This chapter firstly reviews the progress of smart sensors. The current is information phase. The information which can be provided from smart sensors is highly demanded by industrial users. The information of process faults and essential variable approaching changes are valuable for industrial users. To extract faults and prediction information from measurement, the algorithms and models to implement process fault detection and variable prediction are reviewed. From the review, parameter estimation and CUSUM can be used together for fault detection, while grey model and ARX model combining with Kalman filter can be used for input-implicit system and input-explicit system prediction respectively. To serve these algorithms, the RTOSs of smart transmitters are important. The mainstream RTOSs using in smart transmitters are reviewed.

Chapter 3

3 Introduction of a Platform for Smart Sensor Implementation

An industrial networking temperature transmitter with PROFIBUS-PA Fieldbus network functions, named NCS-TT105, is used as implementation platform to be integrated with designed functions. The NCS-TT105 series transmitters are commercial products designed and manufactured by the Microcyber Corporation in Shenyang, China. In 2009, this series' products passed the test of PROFIBUS Nutzerorganisation (PNO) certification [93]. Nowadays, NCS-TT105 have become high-quality temperature transmitters and boardly used in industrial system [94]. For the purpose of simplicity and alignment, the NCS-TT105 transmitter and its sensing elements are collectively called NCS-TT105 sensor. This chapter introduces the functions, hardware and its integrated RTOS software system. The product standard and specifications of this sensor are also summarized.

3.1 Functionalities of NCS-TT105

NCS-TT105 transmitter is an industrial standard designed temperature transmitter. It also combines PROFIBUS-PA Fieldbus technology as its network capabilities. As a commercial and industrial designed device, using its technical specifications and metallic housing assembly, NCS-TT105 can be deployed in the industrial environment.

To measure temperature, NCS-TT105 has two sensing channels for connecting external sensing elements such as resistance temperature detectors (RTDs). The NCS-TT105 can measure two channels of temperature by two sensing channels independently and can provide accurate measurement via its internal program including linearization, calibration, and cold junction compensation functions.

Through PROFIBUS-PA, NCS-TT105 can transmit temperature data of two channels to upper systems. Beside the measurement, the inner faults message such as communication failure, hardware failure, and cold reference junction failure can also sent to upper systems for NCS-TT105 inner problem diagnosis. NCS-TT105 can also acquire power from Fieldbus and the current consumption of the transmitter is less than 14mA.

From smart sensor angle, current NCS-TT105 sensors are designed mainly for providing accurate measurements and transmitting measurements via network. The current NCS-TT105 exists obvious gaps with performance of expected smart sensor which can provide process information. With the capabilities of the hardware and RTOS in NCS-TT105, this transmitter can be further developed and improved.

3.2 Hardware and Software

3.2.1 Hardware

As temperature transmitters, NCS-TT105 series transmitters need to be connected with external sensing elements. The entire assembly structure is shown in Figure 3.1. The electric diagram structure of an NCS-TT105 consists of five main components, as shown Figure 3.2. All the components are assembled in a metallic housing.



Figure 3.1: Assembly Structure of NCS-TT105 with a Sensing Element [94]



Figure 3.2: Schematic Diagram of a NCS-TT105 Sensor [94]

The five main components include [94]:

Terminal board: The terminal board is designed for wiring the connection with PROFIBUS-PA cable. It has two channels for temperature sensing elements, such as RTDs. The whole transmitter is powered by PROFIBUS-PA Fieldbus cable.

<u>Acquisition board:</u> The signal conditioners, amplifiers, a single 24bit high accuracy ADC, and one low-power consumption microcontroller typed MSP430, are designed as part of this acquisition board. The main function of this board is to convert the sensed analog signal to digital data. In addition, the signal compensation and calibration are processed in the microcontroller. The raw measurement data will be transferred via

Universal Asynchronous Receiver/Transmitter (UART) communication to the main processing board. Since ADC are controlled by MSP430, the acquired signals can be processed in acquisition board for higher sampling frequency comparing with transmitting data to processing board. Therefore the SPU of smart sensor which is shown in Figure 1.3 can be assigned in this board.

Main processing board: This board is the core of the transmitters for data processing, Fieldbus communication, internal diagnosis, data storage, and data computing. The ARM7 microcontroller (MCU), with 40MHz processor speed, 32-bit RISC architecture and a 512KBytes flash, is the core of this board. The application functions and PROFIBUS-PA profile are processed in this MCU. The other core chip is the PA Fieldbus interface chip, named FBC0409. This chip contains a Manchester data encoder, designed to comply with IEC 61158-2 and to act as the physical layer and link layer of PA communication. The programming software MULTI, provided by GreenHills Software Company is the development environment.

The APU and BCU of smart sensor which is shown in Figure 1.3 are served by ARM7 MCU in the main processing board. To describe this two unit briefly, the APU and BCU are grouped as the named networking and application unit (NAAU). The designed algorithms will be served by NAAU and processed in the ARM7 MCU. The program firmware needs to be downloaded into the ARM7 MCU through a JTAG port.

Isolation board: This board is used for the isolation of physical and electric between the main processing board and the acquisition board, enabling the circuits in main processing board to be electromagnetically immune with acquisition board.

Display component: The display component is an optional component used to display parameters or measurement. Using this component, users can check the parameters and inner fault codes from the transmitters.

3.2.2 Software

The Mentor embedded Nucleus RTOS serves as the operating system platform in NCS-TT105 transmitters. Nucleus RTOS is commercial RTOS for embedded system, and is designed for essential applications with deterministic performance demands [95].

Nucleus RTOS is typically written in ANSI C and implemented as a C library. In an embedded device, a Nucleus RTOS file requires about 20K bytes of ROM memory with all service use scenario, and can be tailored as small as 2K bytes. This operating system can therefore easily accommodate both low-end and high-end microprocessors. This system has been deployed in over 3 billion devices, and has been proven to be a quality commercial RTOS in embedded markets [96].

There are some highlights related to the use of this operating system as a platform for smart sensor implementation:

<u>Scalability</u>: At the system level, the components of a Nucleus RTOS can be added or removed by sensor developers, providing flexibility for integration schemes for various kinds of sensors. The Nucleus RTOS supports a task integration tool known as a Nucleus Platform Solution [96]. Using this tool, multiple tasks can be integrated with both flexibility and scalability.

<u>Reliability:</u> The Nucleus RTOS is a commercial RTOS widely used in industrial automation systems. Its technical communities and service support teams are mature, with users receiving strong support. Moreover, the Nucleus process model component can rapidly isolate software faults using its self-diagnostic function. In short, the Nucleus RTOS is a market-proven product.

<u>Portability</u>: The Nucleus RTOS system can be integrated into a wide range of microprocessors. The tasks and programs inside the RTOS can be migrated easily among different hardware platform. The migration solution can save much time on the bottom of embedded system development.

From the investigation, the MCU workload for executing current programs in NCS-TT105 is less than 15%. The RAM and ROM, and RTOS have enough resources for further program integration and execution. Moreover, the algorithms can be executed in real-time and the implementation can be efficient in Nucleus RTOS. Therefore, utilizing the computational and memory resources, and leveraging those commercial and mature hardware and software system, the designed algorithms and implemented functions can be integrated into NCS-TT105.

3.3 Standards of Compliance and Specifications of the Platform

The measurement from two channels sensing elements, and PROFIBUS-PA Fieldbus communication are the two main functions from NCS-TT105. The standards and specifications of these two parts are introduced in this section.

3.3.1 Standards followed by NCS-TT105

A. Sensing standards

Resistance temperature detectors (RTDs) and thermocouples (TCs) are common used sensing elements in industrial systems for sensing temperature. IEC 60751, and IEC 60584-1 are the two standards followed by NCS-TT105 developers.

IEC 60751 [97]: This standard specifies the relationship between temperature and industrial platinum resistance. This relationship can be found from Appendix A. Using the principle in this standard, the temperature can be calculated by measuring RTD resistance. As many brands of smart transmitters, NCS-TT105 uses equation computing method to measure and transfer temperature value. Generally, if the temperature of the measured object is less than 850 °C, the RTDs are preferred to be used as sensing elements.

IEC 60584-1 [98]: This standard specifies the relationship between temperature and multiple types of TCs using polynomial equations and tables. This relationship can be found from Appendix A. Generally, the temperature range of TCs is large. Most of TC

types are higher than 1000 °C. Therefore, if the temperature of the measured objects is higher than 850 °C, TCs are suitable to be utilized as sensing elements.

As shown above, to convert signal raw data to temperature measurement, the computation of mathematical equations are required following these standards. In NCS-TT105, the ARM 7 MCU in main processing board execute these computations. The measurement functions such as calibration, linearization, and scaling measured data are realized in Transducer Block of PA device profile, which are introduced in the following subsection.

B. PROFIBUS-PA

PROFIBUS-PA is an industrial leading Fieldbus technologies used for connecting field level sensors and actuators to automation control systems [99]. For instance, the field devices, such as temperature, pressure, and flow sensors, can all be integrated seamlessly into mainstream control systems such as the Siemens PCS 7, the Emerson Delta V, or the Honeywell C-300. The typical architecture of PROFIBUS-PA Fieldbus is shown in Figure 3.3. With PROFIBUS-PA, NCS-TT105 can be widely used in mainstream control systems.



Figure 3.3: A Typical PROFIBUS-PA Architecture [99]

According to IEC 61158-2, Manchester Encoded and Bus Powered (MBP) technology is used for PROFIBUS-PA [100]. MBP provides a two-wire data and power combination technology, in which digital communication is represented as alternating current, while the power is direct current. The minimum base current in the bus' physical layer is 10mA, and the current will be around 1 to 19 mA for data transmission over time. This enable NCS-TT105 to be powered and communicate via PROFIBUS-PA cable directly.

To develop transmitters that fulfill PROFIBUS-PA standards, PA device profile must be implemented into transmitters [101]. A PA device profile complies with Fieldbus protocols and profile specifications, and defines all functions and parameters for different field devices. Sensor developers should ensure they are compliant with the PA devices profile standard to define Physical Block (PB), Transducer Block (TB), and Function Block (FB) in transmitters. These blocks, and the structure of a PA device profile, can be seen in Figure 3.4. The detail roles of these three types of blocks are summarized in Appendix B [101].



Figure 3.4: Block Structure of a PA devices Profile [100]

In NCS-TT105, FBs include several kinds of blocks such as Analog Input FB, and Analog Output FB. Using FBs, PROFIBUS-PA devices can exchange data and diagnosis status with controller in PROFIBUS-PA Fieldbus [101]. To identify the devices in PROFIBUS-PA Fieldbus, the configuration of controller needs to import the corresponding GSD files [101]. The original PA devices profile in NCS-TT105 only has Analog Input FBs. Therefore, NCS-TT105 transmitters only have one-way direction communication function for transmitting measurements to upper lever systems. To achieve data exchanging function, the bidirectional communication needs to be implemented via adding Analog Output FBs. The GSD file is needed to be modified. The implementation of bidirectional communication is introduced in section 5.3.

3.3.2 Specifications of NCS-TT105

As a commercial industrial transmitter, the specifications of NCS-TT105 are beyond the average of the common temperature transmitters. The basic technical specifications of NCS-TT105 are shown in Table 3.1. The NCS-TT105 can connect many types of sensing elements for measuring temperature. Furthermore, the specifications of measuring temperature with different type of RTDs and TCs are summarized in Table 3.2 [94]. The typical configuration for practical application uses PT100 to connect with NCS-TT105. When the condition temperature of NCS-TT105 is 25 °C, the accuracy of PT100 is $\pm 0.3^{\circ}$ C not subject to the working range. From the review of standards and specifications, it can be concluded that, NCS-TT105 is a standard followed and high accuracy industrial transmitter. It is a qualified transmitter platform for implementing algorithms to realize designed smart transmitter.

Basic Technical Specifications		
Sensing Channels	2	
Input Signal:		
- Resistance	PT100, PT1000, CU50, CU100, 0~500Ω, 0~4000Ω	
- Thermocouple	B, E, J, N, K, R, S, T	
- Voltage Signal:	-100mV ~ 100mV	
RTD connection	2, 3 wire	
Update Time	200ms	
Power Supply of Fieldbus	PA: $9 \sim 32$ VDC / Current: ≤ 14 mA	
Operation Temperature	- 30 ~ 70 °C (With display)	
Protection Grade	IP 65	
EMC	IEC 61000	

Table 3.1: NCS-TT105 Basic Technical Specifications [94]

RTD Specifications			
Sensor Type	Working Range (°C)	Accuracy (°C) (25°C)	
PT 100	-200 ~ 850	±0.3	
PT 1000	-200 ~ 850	±0.3	
TC Specifications			
Sensor Type	Working Range (°C)	Accuracy (°C) (25°C)	
В	500 ~ 1810	± 1.0	
Е	-200 ~ 1000	± 0.4	
J	-190 ~ 1200	± 0.4	
K	-200 ~ 1372	± 0.4	
N	-190 ~ 1300	± 0.8	
R	0 ~ 1768	±1.0	
S	0 ~ 1768	±1.0	
Т	-200 ~ 400	±0.4	

Table 3.2: RTD and TC Specifications of NCS-TT105 [94]

3.4 Summary

This chapter introduces hardware compositions, embedded Nucleus RTOS, specifications as well as sensing and PROFIBUS-PA Fieldbus standard of NCS-TT105 transmitter. From the introduction of MCU, RTOS, and specifications, it can be seen that NCS-TT105 is qualified industrial transmitter both in internal hardware architecture, software execution environment. This examines that NCS-TT105 has additional computational resources available for accommodating additional functionalities. It can be a competent platform for further integrated smart functions implementation and practical applications.

Chapter 4

4 Investigation of the Algorithms for Smart Functions

The feasible algorithms to realize process fault detection and variable prediction functions in the NCS-TT105 transmitter are investigated in this chapter.

4.1 Goals and Specifications

4.1.1 Process Fault Detection Smart Function

The fault detection information is valuable for the industrial users if the faults are detected in the incipient period which means that faults are detected before deterioration or failure occurs and within the conventional reaction time of the controller. In practical applications, the process system works around an operating point, which means the above equipment and entire system work under normal status. Once a fault occurs in the equipment or facilitates, the process system may work under abnormal status. More seriously, the fault in incipient period can potentially result in deterioration and severe accident if the fault cannot be detected and resolved in time. Especially, most of process systems have large inertia. The reaction time from fault occurrence till the effective actions taken by the controller may be a long time. Therefore, the incipient fault detection can be used to prevent serious failures in process or esstential equipment and to avoid deterioration.

To exam if the process fault detection function can detect the faults in the process in the incipient period, the fault indication response time which is the interval time between faults occurrence and fault indication is important criteria. This fault indication response time must be less than the reaction time of the loop controller against the faults, allowing the provided fault information can be used to take early actions for the controller. Considering the typical reaction time of the loop controller in the test facility system which is used for validating the designed smart transmitter is 10s, the fault information needs to be provided by smart transmitters within 10s once faults occurred in the monitored test facility system. Therefore, 10s is selected as the specification of the fault indication response time of the integrated process fault detection function.

The specifications of the integrated process fault detection function is shown in Table 4.1. The fault indication can be binary value which is from 0 to 1 as alarm signals. The response time of the fault indication need to be less than 10s after the faults occurs. Furthermore, to guarantee the integrated fault detection function workable in smart transmitters, the workload of MCU to execute process fault detection function should be less than 25%, which means the execution time of the algorithms should be less than 100ms according with 400ms period time. This also means that the data sample time and execution period of the algorithms are 400ms. All those specifications are the targets for fault detection algorithms implementation and are validated in the validation chapter.

Specifications of Smart Function: Fault Detection		
Faults in Process	Heater tripping, Chiller shut down, Pipeline leak, Pipeline plugged, etc	
Faults in Sensing Elements	Sensing elements disconnection (RTD disconnected, Sensing signals loss)	
Algorithms Output	Alarm signals 0 ->1	
Alerts Response Time	< 10s Average	
Algorithms Workload	< 25%	

Table 4.1: Specifications of Process Fault Detection Smart Function

4.1.2 Variable Prediction Smart Function

The predicted information allows plant operators to foresee the future changes of the process variables and take actions as early as possible. The target of integrated variable prediction function in smart transmitters is to predict the process essential variables after dead-time or further steps to foresee the process for preventive actions. The predicted results can be used for prediction alarms, preventive maintenance, or predictive control.

The variable prediction function needs to predict the process variables under two scenarios; one is that the system works in normal status, and the other one is that the system is in abnormal status. In practical application, the normal status of the plant process systems is the system working around operating point. The system is in a stable status, and the variables are small fluctuated. The abnormal status of the process systems is the system working far away with the required working point caused by equipment faults, or essential equipment shut down. The prediction in system normal status can be used for optimizing control performance or preventing faults occurrence. The prediction should be more accurate. The prediction in system abnormal status can be used for indicating the trend of the essential variables to guide preventive actions or execute emergency protection when the variables of the process fluctuate dramatically. Therefore, the prediction accuracy in system abnormal status is different in system normal status.

The steps of prediction is also essential criteria. The prediction function needs to foresee the process changes further than dead-time. For example, if the dead-time of a system is 8s, the predicted results further 8s can support operators to foresee the process changes. The specific parameters of this steps value should be referred to application requirements and process behavior.

According to the discussion from previous two paragraphs, the specifications of the integrated variable prediction function is shown in Table 4.2. Since the dead-time of the monitored variable in test facility system is 8s, the prediction steps is set as 10s which is further than the dead-time. Furthermore, to exam the error of the prediction according to the dynamic changed actual value, the prediction error rate which is an average value calculated using the MAPE (mean absolute percentage error) method [102] is used. The calculation of the prediction error rate is shown as in equation (4.1)

$$Prediction \ error \ rate = Avg[Abs(\frac{Prediction(i) - Actual(i+p)}{Actual(i+p)})] \times 100\%$$
(4.1)

where *p* is the prediction steps, *Prediction(i)* is the *p*-step ahead prediction at *ith* step Actual(i + p) is the actual process value after *p* steps from *ith* step. It should be noted that the Actual(i + p) must not equal to 0 when using equation (4.1), if the unit of the Actual(i + p) is Celsius rather than Kelvins. The prediction error rate criteria for system around operating point is set as 1.2%, which means that the average error between the

prediction and actual valuable after 10s needs within ± 0.3 °C if the actual process variable is 25°C. While if the process system has faults or series problems, the variables change dramatically under this condition. Therefore, the prediction error rate criteria for system abnormal are 6% which means the average error between the prediction and actual valuable after 10s needs within ± 1.5 °C if the actual process variable is 25°C.

Specifications of Smart Function: Variable Prediction		
Prediction Steps	10s (> dead-time of the process)	
Prediction Accuracy (Prediction Error Rate)	< 1.2% (0.3°C/25°C) process works around operating point	
	< 6% (1.5°C/25°C) process works under abnormal status	
Algorithms Workload	< 15%	

Table 4.2: Specifications of Variable Prediction Smart function

Furthermore, to guarantee the integrated variable prediction function workable in the transmitter, the workload of MCU to execute variable prediction function needs to be less than 15%, which means the execution time of the algorithms should be less than 150ms according with 1000ms period time. All those specifications will be the targets for prediction algorithms implementation and are validated in the validation chapter.

4.2 Algorithms for Process Fault Detection

As discussed in section 2.2, the residual generation and residual evaluation are two main steps. The ARX model, and its online parameter estimation by EWRLS for residual generation, as well as CUSUM algorithms for residual evaluation are studied in the following subsections. Furthermore, to apply the fault detection methods to the NCS-TT105 transmitter, the algorithms which are adaptive and recursive are investigated.

4.2.1 ARX and EWRLS for Residual Generation

The ARX model involving inputs of the system is used as model to describe process system. Considering the parameters of the process are not constant, the Exponential

Weighting Recursive Least Squares (EWRLS) algorithm, with an exponential forgetting memory, are used for parameter estimation and residual generation in smart transmitters. The diagram of residual generation using ARX and EWRLS is shown in Figure 4.1 [51].



Figure 4.1: Parameter Estimation using ARX Model and EWRLS [51]

A. ARX Model

In the engineering field, the process can be seen as a linearized and dynamic behavior around an operating point [63][103]. Therefore, linear model ARX model are employed to describe the relationship between input u(k) and response $y_u(k)$ of the process system. The discrete time model of ARX can be represented as (4.2)

$$y_{u}(k) + a_{1}y_{u}(k-1) + \dots + a_{m}y_{u}(k-m) = b_{1}u(k-d-1) + \dots + b_{m}u(k-d-m)$$
(4.2)

where,

$$u(k) = U(k) - U_{op}$$

$$y_{\mu}(k) = Y(k) - Y_{op}$$
(4.3)

The dynamic inputs and outputs at the operating point of the process are U(k) and Y(k). *k* is the discrete time $k = t/T_s = 0, 1, 2, ..., T_s$ is the sampling period of the algorithm. $d = T_d/T_s = 0, 1, 2, ...$ represents the discrete dead-time of the process response [51]. U_{op} and Y_{op} represent the inputs and outputs at the operating point. The U_{op} and Y_{op} can either be recorded from the specific process technology or derived from the mean value. The equation (4.4) can be used to compute mean value as slow time-varying operating point of system [63].

$$\hat{Y}_{op}(k) = \lambda_{avg} \hat{Y}_{op}(k-1) + (1 - \lambda_{avg}) Y(k-1)$$
(4.4)

where λ_{avg} is exponential forgetting factor, Y is process variable, and \hat{Y}_{op} is the mean value used as operating point of process system.

Equation (4.4) corresponds to a first order discrete-time low pass filter $G_f(s) = \frac{1}{T_{1st}s+1}$ for approaching operating point value. T_{1st} is the time constant to decay to 37% from the initial value of a first order system [51]. λ_{avg} is the forgetting factor. e.g. $\lambda_{avg} = e^{-T_s/T_{1st}} = 0.98$.

The corresponding transfer function (4.2) of the ARX model in the z-domain is

$$G_{P} = \frac{y_{u}(z)}{u(z)} = \frac{B(z^{-1})}{A(z^{-1})} z^{-d} = \frac{b_{1}z^{-1} + \dots + b_{m}z^{-m}}{1 + a_{1}z^{-1} + \dots + a_{m}z^{-m}} z^{-d}$$
(4.5)

where the parameter $a_1 \cdots a_m, b_1 \cdots b_m$ will be estimated by EWRLS.

From equation (4.5), using the estimated $\hat{A}(z^{-1})$ and $\hat{B}(z^{-1})$, the residual shown in Figure 4.1 can be computed as equation (4.6)

$$\hat{A}(z^{-1})y(z) - \hat{B}(z^{-1})z^{-d}u(z) = e(z)$$
(4.6)

To represent unknown parameters a_i and b_i in equation (4.5), the estimated parameter vector $\hat{\theta}$ can be written as

$$\hat{\theta} = [\hat{a}_1 \dots \hat{a}_m \,|\, \hat{b}_1 \dots \hat{b}_m]^T \tag{4.7}$$

The data vector $\psi^{T}(k)$ which represent for inputs u(k) and response $y_{u}(k)$ can be written as

$$\psi^{T}(k) = \left[-y_{u}(k-1) \dots - y_{u}(k-m) \mid u(k-d-1) \dots u(k-d-m)\right]$$
(4.8)

B. Exponential Weighting Recursive Least Squares (EWRLS)

In order to estimate the parameters of time-varying systems, EWRLS are used. The core idea of EWRLS can be described in equation (4.9) as

$$\hat{\theta}(k+1) = \hat{\theta}(k) + \gamma(k) \left[\underbrace{y(k+1)}_{new} - \underbrace{\psi^{T}(k+1)\hat{\theta}(k)}_{one-step-ahead} \right]$$
(4.9)

Using the data vector $\psi^{T}(k)$ in (4.8), the parameter vector $\hat{\theta}$ in (4.7) can be updated recursively by EWRLS [51]. u(k) is the inputs of the EWRLS, $y_{u}(k)$ is the reference value of EWRLS.

The EWRLS algorithm can be presentated as [51]

$$\gamma_{w}(k) = \frac{1}{\psi^{T}(k+1)P_{w}(k)\psi(k+1) + \lambda_{w}}P_{w}(k)\psi(k+1)$$
(4.10)

$$e(k+1) = y(k+1) - \psi^{T}(k+1)\hat{\theta}(k)$$
(4.11)

$$\hat{\theta}(k+1) = \hat{\theta}(k) + \gamma_{w}(k)e(k+1)$$
(4.12)

$$P_{w}(k+1) = (P_{w}(k) - \gamma_{w}(k)\psi^{T}(k+1)P_{w}(k))\frac{1}{\lambda_{w}}$$
(4.13)

where $P_w(k)$ is referred to as the inverse correlation matrix, λ_w is the exponential forgetting factor.

To run EWRLS, the $P_w(k)$ and $\hat{\theta}(0)$ of the first recursive loop should be set as the initial value. In [63], the initial value of $P_w(0)$ and $\hat{\theta}(0)$ can be preconfigured as

$$P_{w}(0) = \alpha I , \alpha = 100...10000$$

$$\hat{\theta}(0) = 0$$
(4.14)

where α can be chosen as the smaller value if the signals in ψ vector become larger.

For single input and single output parameter estimation, if the dimension of the parameter a or b in (4.7) is m, then the dimension of estimated parameter vector $\hat{\theta}$ is $2m \times 1$; the dimension of data vector $\psi^{T}(k)$ is $1 \times 2m$; the correction factor $\gamma_{w}(k)$ is $2m \times 1$; and the dimension of the covariance matrix P(k) is $2m \times 2m$.

Through the above algorithms, the process parameters in $\hat{\theta}$ are estimated recursively using EWRLS. If λ_w is less than 1, it enables EWRLS to gradually reduce the impacts from the old data in vector $\psi^T(k)$, and the latest data will be weighted strongly. This allows EWRLS algorithm to track the time variant parameters better than RLS. The forgetting factor λ_w should be set between $0.9 < \lambda_w < 0.999$. If λ_w is close to 0.9, the EWRLS can better track time-varying systems, but the process noise should be small. If λ_w is close to 0.999, the process noise can be larger, but the tracking performance will be decreased. The λ_w value should be set so as to trade off the tracking performance with disturbance suppression, based on the required practical application. Once the initial value and the forgetting factor are set, the algorithms can run recursively and adaptively.

It should be mentioned that if the EWRLS needs to obtain unbiased parameter estimations, the system disturbances have to be generated from white noise, or the e(k)which is assued to be equal to disturbance is independent from the data vector $\psi^{T}(k)$, if the parameter estimation is used in closed loop [63][77]. Also, the input vector ψ should be exciting sufficiently. Otherwise, $P_{w}(k+1)$ will increase continuously. This will cause data value in the correcting vector $\gamma_{w}(k)$ to be increased continuously, and estimator becomes more sensitive, even unstable [51]. Therefore, disturbances in the estimated process should be investigated before the algorithm is applied, and the input excitation should be monitored carefully. If the exciting of the inputs is not enough, λ_w has to be time-variant and needs to be set as 1 under this scenario.

C. Residual generation

After the parameter vector $\hat{\theta}(k)$ is estimated, combining with data vector $\psi^{T}(k)$ and according to the equation (4.6), the residual can be computed as the equation (4.15)

$$y_{\mu}(k) - \psi^{T}(k)\hat{\theta} = e(k) \tag{4.15}$$

The (4.15) can be used as a residual and can be updated for every sampling time. Further, e(k) can be used for further residual evaluation for fault detection.

4.2.2 CUSUM for Residual Evaluation

As discussed in section 2.2.3, CUSUM test is one of the most commonly used statistical tools for detecting signal changes to make fault detection decisions. In this subsection, a specific CUSUM algorithms named two-sided CUSUM test [66] algorithms are studied.

A. Two-sided CUSUM test

Residual is the data source of residual evaluation. Two-sided CUSUM [66] test are suitable to be used as a statistical tool for residual evaluation when residual r_k is a fluctuated signal around zero. The residuals can be calculated through the subtraction of real measurements y_k and estimation \hat{y}_k . From the residual generation procedure, the residual r_k can be represented as

$$r_k = e_k = y_k - \hat{y}_k \tag{4.16}$$

where, the residual e_k can be calculated using equation (4.15)

A two-sided CUSUM test algorithm is presented as follows:

$$s_k^{(1)} = r_k$$

 $s_k^{(2)} = -r_k$
(4.17)

$$g_{k}^{(1)} = \max(g_{k-1}^{(1)} + s_{k}^{(1)} - \upsilon, 0)$$

$$g_{k}^{(2)} = \max(g_{k-1}^{(2)} + s_{k}^{(2)} - \upsilon, 0)$$
(4.18)

if
$$g_{k}^{(1)} > h$$
 or $g_{k}^{(2)} > h$
Alarm Indication = 1 (4.19)
reset $g_{k}^{(1)} = 0, \ g_{k}^{(2)} = 0$

where v is a small drift term to be set as a small threshold for cumulation in every recursive step, and *h* is the threshold for triggering the fault detection alarm. v and *h* could be set to a small value when the noise from residual r_k is low; then the faster detection can be sought. While if the noise of the residual is high, v and *h* should be set to a larger value to avoid false detection. After an alarm indication is triggered from 0 to 1, $g_k^{(1)}$ positive cumulation and $g_k^{(2)}$ negative cumulation are reseted to 0, and the new recursive cumulation is continuous. The value of alarm indication can be used for faults detection alerts.

The parameters drift v and threshold h are two essential parameters for two-sided CUSUM test algorithm. The setting of threshold h and drift v will decide the sensitivity, delay, or falsity of alarms. In order to apply smart sensor for different practical application scenarios, h and v should be set accordingly. Therefore, a dynamic methods for solving threshold h and drift v are appropriate.

B. Dynamic drift and alarm threshold

To enable drift v and threshold h to be adaptive with different applications, the dynamic reference value as well as two ratio parameters are proposed. The square root of residual variance and the average of the absolute value of the residual will be computed as dynamic reference value. The dynamic v and h are shown as follows:
$$\upsilon_k = K_{\upsilon} \times Sqrt(Var(e_k)) \tag{4.20}$$

$$h_k = K_h \times Avg(Abs(e_k)) \tag{4.21}$$

where K_{v} and K_{h} are the ratio parameters to be set. From practical test, the arrangement of K_{v} and K_{h} could be $0 < K_{v} < 1$ and $K_{h} > 1$. The Var() is the function for calculating variance. The Avg() is the function for calculating mean value. With the $Var(e_{k})$ and $Avg(e_{k})$, the reference value can computed dynamically, and the value of drift v_{k} and threshold h_{k} can be determined setting K_{v} and K_{h} .

C. Estimation method for mean and variance

The mean Avg() and variance Var() can be calculated using the methods:

The original method

The mean estimation for the length N of the discrete time signal x(i) is defined as [51]

$$\mu_x = \frac{1}{N} \sum_{i=1}^{N} x(i)$$
(4.22)

Moreover, the unbiased variance for sampled signals can be calculated as [51]

$$\sigma_x^2 = \frac{1}{N-1} \sum_{i=1}^{N} (x(i) - \mu_x)^2$$
(4.23)

To calculate equations (4.22) and (4.23), the smart sensors need to storage and compute N length of data in every sampling time. To compute mean and variance more efficiently, the recursive method can be used.

Recursive exponential forgetting method

As an efficient online computing solution, recursive methods are suitable for either computers or embedded systems applications. Therefore, the recursive methods to calculate mean and variance are used. For variance Var() computation, as shown in [51], the *kth* step of mean can be derived through the *k*-1*th* step, mean value $\hat{\mu}_x(k)$ can be computed as the equation (4.24)

$$\hat{\mu}_{x}(k) = \hat{\mu}_{x}(k-1) + \frac{1}{k}(x(k) - \hat{\mu}_{x}(k-1)) = \left(1 - \frac{1}{k}\right)\hat{\mu}_{x}(k-1) + \frac{1}{k}x(k)$$
(4.24)

Revisiting the low pass filter equation (4.4), if the number of samples $k = \frac{1}{1 - \lambda_e}$ or

 $\lambda_e = 1 - \frac{1}{k}$, $\lambda_e = e^{-T_s/T_{1st}}$. Then the equation (4.24) can be rewritten as

$$\hat{\mu}_{x}(k) = \lambda_{e} \hat{\mu}_{x}(k-1) + (1-\lambda_{e})x(k)$$
(4.25)

The parameter λ_e , the so-called exponential forgetting factor, produces a lower weighting for recent inputs. If the value of $(1 - \lambda_e)$ is higher, it means that the recent inputs are higher weighted, and the past inputs will be forgotten soon.

For variance Var() computation, as shown in [104], the definition of sample variance is described as the equation (4.26)

$$s_x^2(k) = \frac{\sum_{i=1}^k [x(i) - \hat{\mu}_x(k)]^2}{k - 1}$$
(4.26)

Combining the equation (4.24), the recursive sample variance is represented as [104]

$$s_x^2(k) = \frac{k-2}{k-1} s_x^2(k-1) + k[\hat{\mu}_x(k) - \hat{\mu}_x(k-1)]^2$$

$$= \frac{k-2}{k-1} s_x^2(k-1) + \frac{1}{k} [x(k) - \hat{\mu}_x(k-1)]^2$$
(4.27)

Subtracting $k = \frac{1}{1 - \lambda_e}$ into the (4.27), the recursive sample variance will result in [51]

$$s_x^2(k) = \frac{2\lambda_e - 1}{\lambda_e} s_x^2(k-1) + (1 - \lambda_e) [x(k) - \hat{\mu}_x(k-1)]^2$$
(4.28)

Equations (4.25) and (4.28) can be used as recursive methods to calculate the estimated mean and sample variance. The exponential for forgetting $\lambda_e = 1 - \frac{1}{k}$, where k corresponds to the number of samples.

4.3 Algorithms for Variable Prediction

To enable the NCS-TT105 transmitter to predict the variable both for system with known inputs and unknown inputs, the algorithms of prediction functions for those two kinds of systems are investigated. To implement prediction in system with known inputs, the prediction algorithms named Kalman predictor prediction including ARX model, Kalman predictor algorithms, and multi-step iteration are combined. To implement prediction in system with unknown inputs, the prediction algorithms named grey model prediction including GM(1,1) and GM(2,1) as well as weighted decisions tool are grouped to predict the response of the system without considering the inputs of the system.

4.3.1 Kalman Predictor Prediction

Leveraging Kalman predictor, and using the determined state-space model and system inputs, the states of state-space model can be one-step predicted accurately through the Kalman predictor. The predicted states can be derived to the predicted response of the system and can be used for further steps iterative prediction.

A linear process system with discrete time input and output signals and stochastic noise can be represented using state space equations, as in [62] and [80]:

$$x(k+1) = Ax(k) + Bu(k) + w(k)$$

$$z(k) = Cx(k) + v(k)$$
(4.29)

where the process state-space matrices A, B, C are assumed to be known.

u(k): are the control inputs of the process, which come from controller.

- x(k): are the internal state variables of state-space equation.
- z(k): are sensor measurements.
- w(k): represents process noise.
- v(k): represents measurement noise.

w(k) and v(k) are stochastic variables, and are assumed to follow an independent, zeromean normal gaussian distribution. The mean values of two kinds of noise signal can be represented as

$$E\{w(k)\}=0, E\{v(k)\}=0$$
 (4.30)

The covariance matrices

$$E\left\{w(k)w^{T}(k)\right\} = Q, \quad E\left\{v(k)v^{T}(k)\right\} = R,$$
(4.31)

where Q is the process noise covariance and R is the measurement noise covariance.

The optimal estimated states are represented as

$$\hat{x}(k \mid j) = E\{x(k) \mid Z_j\}, \quad Z_j = \{z(0), z(1), \dots z(j)\}$$
(4.32)

If k > j, equation (4.32) will solve a prediction problem, which means that the estimated states in present time *k*, are predicted through the measurements in the *j* time instance.

To solve one-step ahead prediction $\hat{x}(k+1|k)$, the z(k) and $\hat{x}(k|k-1)$ are recursively computed by Kalman predictor [51][62][80][81]. The algorithms are introduced as follows:

$$\overline{K}(k) = P^{-}(k)C^{T}[CP^{-}(k)C^{T} + R]^{-1}, where P^{-}(k) = P(k \mid k - 1)$$
(4.33)

$$\hat{x}(k+1|k) = A\hat{x}(k|k-1) + Bu(k) + A\overline{K}(k)[z(k) - C\hat{x}(k|k-1)] = A\{\hat{x}(k|k-1) + \overline{K}(k)[z(k) - C\hat{x}(k|k-1)]\} + Bu(k)$$
(4.34)

$$P^{-}(k+1) = P(k+1|k) = A\left\{ [I - \overline{K}(k)C]P^{-}(k) \right\} A^{T} + Q$$
or $P^{-}(k+1) = AP^{-}(k)A^{T} - AP^{-}(k)C^{T}[CP^{-}(k)C^{T} + R]^{-1}CP^{-}(k)A^{T} + Q$
(4.35)

The dimension of the matrices and vectors in equations (4.33) (4.34) and (4.35) are shown in Table 4.3.

Variable	Definition	Dimension
x(k)	States	$m \times 1$
y(k)	Measurement	q imes l
u(k)	Control inputs	$p \times 1$
Α	Process matrix	$m \times m$
В	Control matrix	$m \times p$
С	Measurement matrix	$q \times m$
Q(k)	Process noise covariance matrix	$m \times m$
R(k)	Measurement noise covariance matrix	q imes q
$\overline{K(k)}$	Kalman Filter gain	$m \times q$
P(k)	Correlation Matrix of the error	$m \times m$

Table 4.3: Matrix Dimension List of Kalman Predictor

The calculation includes two main steps:

Prediction:

$$\hat{x}(k+1|k) = A\hat{x}(k|k) + Bu(k)$$

$$P^{-}(k+1) = AP(k)A^{T} + Q$$
(4.36)

Correction:

$$\hat{x}(k \mid k) = \hat{x}(k \mid k-1) + \bar{K}(k)[z(k) - C\hat{x}(k \mid k-1)]$$

$$P(k) = [I - \bar{K}(k)C]P^{-}(k)$$
(4.37)

The initial step for running (4.33) to (4.35) recursively should be set with

$$\hat{x}(0) = E[x(0)], \quad P^{-}(0) = E[(x(0) - \hat{x}(0))(x(0) - \hat{x}(0))^{T}]$$
(4.38)

The covariance matrices of the estimation error are [51]:

$$P^{-}(k+1) = P(k+1|k) = E\left\{\tilde{x}(k+1|k)\tilde{x}^{T}(k+1|k)\right\}$$
prediction error $\tilde{x}(k+1|k) = \hat{x}(k+1|k) - E\left\{\hat{x}(k+1|k)\right\}$
(4.39)

Equation (4.35) can be recognized as Riccati Difference Equation (RDE) [81], which is for solving infinite-horizon optimal state control problems. The $P^-(k+1)$ which means

the variance of estimation error can result in an asymptotic result with a steady state matrix. Since the RDE is independent of u(k) and z(k), it could be offline computed when *A*, *B*, *C*, *Q*, and *R* are fixed.

Once the one-step-ahead states prediction $\hat{x}(k+1|k)$ is solved, the multi-steps prediction can be computed cyclically using multi-step iteration prediction from equation (2.4) or the equation (4.40).

$$\hat{x}(k+1) = A\hat{x}(k) + Bu(k)$$
(4.40)

Therefore, the final *p*-steps prediction of process response that considers inputs effects are derived by equation (4.41)

$$\hat{z}(k+p) = C\hat{x}(k+p)$$
 (4.41)

where p is the p-steps for further ahead prediction.

4.3.2 Grey Model Prediction

In this subsection, two grey models, named first order GM(1,1) and second order GM(2,1) will be presented. These two models are widely applied in various areas. The core concepts of these two models both use a grey differential equation as their model structure, and LS as parameters estimation method. A small sets of sampled data are used as data source for determining a grey model and calculating for accumulating generator sequences. Once its parameters are determined, the grey model can be used to predict the process variables. A more detailed procedure of modeling and iterative variables prediction are shown in the follows.

A sequence of the raw data $x^{(0)}(k)$ with length *n* which is acquired from $x^{(0)}(k-n+1)$ to $x^{(0)}(k)$ is denoted as $X^{(0)} = (x^{(0)}(1), x^{(0)}(2), \dots x^{(0)}(n))$. $x^{(0)}(k)$ could be used to represent the response of the system in *kth* step of algorithm sampling and the response of the system is need to be predicted. The accumulation of $X^{(0)}$ is generated as a new sequence

and denoted as $X^{(1)} = (x^{(1)}(1), x^{(1)}(2), \dots x^{(1)}(n))$. The $X^{(1)}$ is called the first-order accumulating generator (1-AGO) of sequence $X^{(0)}$ [105], where

$$x^{(1)}(i) = \sum_{k=n+1}^{k=n+i} x^{(0)}(i), \quad i = 1, 2, \cdots, n$$
(4.42)

Correspondingly, the $\alpha^{(1)}X^{(0)}$ is called the first-order inverse accumulating generator (1-IAGO) of sequence $X^{(0)}$ [105], where

$$\alpha^{(1)}X^{(0)} = \left\{ \alpha^{(1)}x^{(0)}(2), \dots, \alpha^{(1)}x^{(0)}(n) \right\}$$

$$\alpha^{(1)}x^{(0)}(i) = x^{(0)}(i) - x^{(0)}(i-1), \quad i = 2, \dots, n$$
(4.43)

Let $Z^{(1)} = (z^{(1)}(2), z^{(1)}(3), \dots z^{(1)}(n))$ be the adjacent neighbor means of the sequence $X^{(1)}$, where

$$z^{(1)}(i) = \frac{1}{2} (x^{(1)}(i) + x^{(1)}(i-1)), \quad i = 2, 3, \cdots, n$$
(4.44)

It can be seen that the length of $Z^{(1)}$ is n-1.

A. GM(1,1) Model

A so-called basic form of a GM(1,1) model, which refers to a "first order grey model in one variable" [74], can be written as

$$x^{(0)}(i) + az^{(1)}(i) = b \tag{4.45}$$

The GM(1,1) model is used to describe a sequence of data in which $X^{(0)}$ is monotonic change, and the corresponding $X^{(1)}$ satisfies the law of exponentiality [105].

To solve the parameters of equation (4.45), it can be rewritten as

$$x^{(0)}(i) = a(-z^{(1)}(i)) + b \cdot 1 \tag{4.46}$$

Using the LS method to estimate the parameters vector $\hat{\theta}_{GM11} = (\hat{a}, \hat{b})^T$ with the n-1 dimension, and

$$\hat{\theta}_{GM11} = \begin{bmatrix} \hat{a} \\ \hat{b} \end{bmatrix} = \begin{bmatrix} B^T B \end{bmatrix}^{-1} B^T Y$$
(4.47)

where

$$Y = \begin{bmatrix} x^{(0)}(2) \\ x^{(0)}(3) \\ \vdots \\ x^{(0)}(n) \end{bmatrix}, B = \begin{bmatrix} -z^{(1)}(2) & 1 \\ -z^{(1)}(3) & 1 \\ \vdots & \vdots \\ -z^{(1)}(n) & 1 \end{bmatrix}$$
(4.48)

Then if $X^{(0)}$ is a non-negative sequence, the whitenization equation of the GM(1,1) model (4.45) is derived as [105]

$$\frac{dx^{(1)}}{di} + \hat{a}x^{(1)} = \hat{b}$$
(4.49)

The solution of the differential equation (4.49) is given by

$$\hat{x}^{(1)}(i) = (x^{(1)}(i) - \frac{\hat{b}}{\hat{a}})e^{-\hat{a}(i-1)} + \frac{\hat{b}}{\hat{a}}, i = 1, 2, \dots n \quad \text{or}$$

$$\hat{x}^{(1)}(i) = (x^{(1)}(n) - \frac{\hat{b}}{\hat{a}})e^{-\hat{a}(i-n)} + \frac{\hat{b}}{\hat{a}}, i = 1, 2, \dots n \qquad (4.50)$$

The estimated $\hat{x}^{(0)}(i)$ can be restored by the calculation from $\hat{x}^{(1)}(i)$ via equation (4.51)

$$\hat{x}^{(0)}(i) = \hat{x}^{(1)}(i) - \hat{x}^{(1)}(i-1), \quad i = 2, \dots n$$

$$\hat{x}^{(0)}(1) = \hat{x}^{(1)}(1)$$
(4.51)

From equation (4.50), it can be seen that the calculation of $\hat{x}^{(1)}(i)$ should involve high order exponent computation by differential equations. It leads complex computation in smart transmitters.

To derive $\hat{x}^{(1)}(i)$ and $\hat{x}^{(0)}(i)$ via iterative equations (4.45), an alternative method to solve $x^{(0)}(i)$ are derived and are shown in (4.52) and (4.53) as

$$x^{(0)}(i) = a(-z^{(1)}(i)) + b \cdot 1$$

= $a[-0.5x^{(1)}(i) - 0.5x^{(1)}(i-1)] + b$
= $a[-0.5(x^{(0)}(i) + x^{(1)}(i-1)) - 0.5x^{(1)}(i-1)] + b$
= $-0.5a \cdot x^{(0)}(i) - a \cdot x^{(1)}(i-1) + b$ (4.52)

Then from (4.52), the restored $\hat{x}^{(0)}(i)$ can be iteratively estimated through $X^{(1)}$ and $\hat{\theta}_{GM11}$ as

$$\hat{x}^{(0)}(i) = \frac{-\hat{a}}{1+0.5\hat{a}} \cdot \hat{x}^{(1)}(i-1) + \frac{\hat{b}}{1+0.5\hat{a}}, i = 2, 3 \cdots n$$

$$\hat{x}^{(1)}(i) = \hat{x}^{(0)}(i) + \hat{x}^{(1)}(i-1)$$
(4.53)

If assuming $\hat{x}^{(0)}(i) \approx x^{(0)}(i)$, $\forall i \in (1, 2, \dots, n)$, the $\hat{x}^{(1)}(i-1)$ can be estimated via $x^{(0)}(i)$ using equation (4.54) which is derived from equation (4.53)

$$\hat{x}^{(1)}(i-1) = \frac{-(1+0.5\hat{a})}{\hat{a}} x^{(0)}(i) + \frac{\hat{b}}{\hat{a}}, \forall i \in (1, 2, \dots n)$$
(4.54)

Then, using $\hat{x}^{(1)}(i-1)$ and $x^{(0)}(i)$ as an initial value and utilizing equation (4.53), the restored value $\hat{x}^{(0)}(i)$ can be estimated iteratively.

B. GM(2,1) Model

If $X^{(0)}$ is a non-monotonic wavelike sequence and the corresponding $X^{(1)}$ satisfies the law of second-order differential equation, GM(1,1) which is used to describe monotonic changes is not appropriate to describe non-monotonic wavelike sequence. Instead of the first order grey model, a so-called GM(2,1) model, which means second order grey model in one variable [74], can be written as

$$\alpha^{(1)}x^{(0)}(i) + a_1 x^{(0)}(i) + a_2 z^{(1)}(i) = b$$
(4.55)

To solve the parameters of the equation, (4.55) can be rewritten as

$$\alpha^{(1)}x^{(0)}(i) = a_1(-x^{(0)}(i)) + a_2(-z^{(1)}(i)) + b \cdot 1$$
(4.56)

Using a least squares method to estimate the parameter matrix $\hat{\theta}_{GM21} = (\hat{a}_1, \hat{a}_2, \hat{b})^T$ with the n-1 dimension, and

$$\hat{\theta}_{GM\,21} = \begin{bmatrix} \hat{a}_1 \\ \hat{a}_2 \\ \hat{b} \end{bmatrix} = \begin{bmatrix} B^T B \end{bmatrix}^{-1} B^T Y \tag{4.57}$$

where

$$Y = \begin{bmatrix} x^{(0)}(2) - x^{(0)}(1) \\ x^{(0)}(3) - x^{(0)}(2) \\ \vdots \\ x^{(0)}(n) - x^{(0)}(n-1) \end{bmatrix}, B = \begin{bmatrix} -x^{(0)}(2) & -z^{(1)}(2) & 1 \\ -x^{(0)}(3) & -z^{(1)}(3) & 1 \\ \vdots & \vdots & \vdots \\ -x^{(0)}(n) & -z^{(1)}(n) & 1 \end{bmatrix}$$
(4.58)

Then the whitenization equation of the GM(2,1) model (4.55) is derived as [105]

$$\frac{d^2 x^{(1)}}{di^2} + \hat{a}_1 \frac{dx^{(1)}}{di} + \hat{a}_2 x^{(1)} = \hat{b}$$
(4.59)

As shown in the second order differential equation (4.59), $x^{(1)}(i)$ is the solution. The $x^{(0)}(i)$ can be calculated by $x^{(1)}(i) - x^{(1)}(i-1)$. As discussed in GM(1,1), solving $x^{(1)}(i)$ from second order differential equation (4.59) is high computational complexity in smart transmitter. Therefore, the iterative method for solving $x^{(0)}(i)$ and $x^{(1)}(i)$ are derived as follows.

From the equation (4.55), the $x^{(0)}(i)$ can be derived as shown in the equation (4.60)

$$\begin{aligned} &\alpha^{(1)}x^{(0)}(i) = a_1(-x^{(0)}(i)) + a_2(-z^{(1)}(i)) + b \cdot 1 \\ &x^{(0)}(i) - x^{(0)}(i-1) = -a_1x^{(0)}(i) - a_2[0.5x^{(1)}(i) + 0.5x^{(1)}(i-1)] + b \\ &x^{(0)}(i) - x^{(0)}(i-1) = -a_1x^{(0)}(i) - a_2[0.5(x^{(0)}(i) + x^{(1)}(i-1)) + 0.5x^{(1)}(i-1)] + b \\ &x^{(0)}(i) + a_1x^{(0)}(i) + 0.5a_2x^{(0)}(i) = x^{(0)}(i-1) - a_2x^{(1)}(i-1) + b \end{aligned}$$

$$(4.60)$$

From (4.60), the restored value $\hat{x}^{(0)}(i)$ can be estimated through $X^{(0)}, X^{(1)}$ and $\hat{\theta}_{GM21}$ as

$$\hat{x}^{(0)}(i) = \frac{1}{1 + \hat{a}_1 + 0.5\hat{a}_2} [\hat{x}^{(0)}(i-1) - \hat{a}_2 \hat{x}^{(1)}(i-1) + \hat{b}], i = 2, 3 \cdots n$$

$$\hat{x}^{(1)}(i) = \hat{x}^{(0)}(i) + \hat{x}^{(1)}(i-1)$$
(4.61)

Assuming $\hat{x}^{(0)}(i) \approx x^{(0)}(i)$ and $\hat{x}^{(0)}(i-1) \approx x^{(0)}(i-1)$, $\forall i \in (2,3,\dots n)$, the $\hat{x}^{(1)}(i-1)$ could be estimated via $x^{(0)}(i)$ and $x^{(0)}(i-1)$ using equation (4.62) which is derived from equation (4.61)

$$\hat{x}^{(1)}(i-1) = -\frac{1+\hat{a}_1+0.5\hat{a}_2}{\hat{a}_2}\hat{x}^{(0)}(i) + \frac{1}{\hat{a}_2}\hat{x}^{(0)}(i-1) + \frac{\hat{b}}{\hat{a}_2}, \forall i \in (1,2,\cdots n)$$
(4.62)

Then, using $\hat{x}^{(1)}(i-1)$ and $x^{(0)}(i-1)$ as the initial values and utilizing equation (4.61), the restored value $\hat{x}^{(0)}(i)$ can be estimated iteratively.

C. Prediction with grey model

As shown above, if changes in $X^{(0)}$ are shown as a monotonic trend, then GM(1,1) can be used for modeling the data sequence, whereas if the trend of $X^{(0)}$ is shown as nonmonotonic and wavelike, then the GM(2,1) can better model the data sequence.

Once the parameters of the two models are estimated, the predicted value can be calculated iteratively through GM(1,1) and GM(2,1). If the length of $X^{(0)} = (x^{(0)}(1), x^{(0)}(2), \dots x^{(0)}(n))$ is $n, x^{(0)}(n) = x^{(0)}(k)$, where the prediction sequence can be denoted as $X_p^{(0)} = (\hat{x}_p^{(0)}(1), \hat{x}_p^{(0)}(2), \dots x_p^{(0)}(P))$, where $x_p^{(0)}(P)$ means the *P*-step ahead predicted value corresponding with the original $X^{(0)}$.

Assuming $\hat{x}^{(0)}(n) \approx x^{(0)}(n)$, using an estimated $\hat{x}^{(1)}(n-1)$ as the initial value, then the prediction $\hat{x}_p^{(0)}(p)$, p = P+2 using GM(1,1) can be computed as

$$\hat{x}_{p}^{(0)}(p) = \frac{\hat{a}}{1 - 0.5\hat{a}} \cdot \hat{x}_{p}^{(1)}(p - 1) + \frac{\hat{b}}{1 - 0.5\hat{a}}, p = 2 \cdots P + 2$$

$$\hat{x}_{p}^{(1)}(p) = \hat{x}_{p}^{(0)}(p) + \hat{x}_{p}^{(1)}(p - 1)$$
initial: $\hat{x}_{p}^{(1)}(1) = \hat{x}^{(1)}(n - 1)$
where $\hat{x}^{(1)}(n - 1) = \frac{(1 - 0.5\hat{a})}{\hat{a}} \cdot x^{(0)}(n) - \frac{\hat{b}}{\hat{a}}$
(4.63)

Assuming $\hat{x}^{(0)}(n) \approx x^{(0)}(n)$ and $\hat{x}^{(0)}(n-1) \approx x^{(0)}(n-1)$, the $\hat{x}^{(1)}(n-1)$ can be estimated via $x^{(0)}(n)$ and $x^{(0)}(n-1)$. Then using $\hat{x}^{(1)}(n-1)$ and $x^{(0)}(n-1)$ as the initial values, the prediction $\hat{x}_p^{(0)}(p)$, p = P + 2 using GM(2,1) can be calculated as

$$\hat{x}_{p}^{(0)}(p) = \frac{1}{1 + \hat{a}_{1} + 0.5\hat{a}_{2}} [\hat{x}_{p}^{(0)}(p-1) - \hat{a}_{2}\hat{x}_{p}^{(1)}(p-1) + \hat{b}], p = 2, 3 \cdots P + 2$$

$$\hat{x}_{p}^{(1)}(p) = \hat{x}_{p}^{(0)}(p) + \hat{x}_{p}^{(1)}(p-1)$$
initial: $\hat{x}_{p}^{(0)}(1) = x^{(0)}(n-1)$ and $\hat{x}_{p}^{(1)}(1) = \hat{x}^{(1)}(n-1)$
where $\hat{x}^{(1)}(n-1) = \frac{(1 + \hat{a}_{1} + 0.5\hat{a}_{2})}{-\hat{a}_{2}} \cdot x^{(0)}(n) + \frac{1}{\hat{a}_{2}}x^{(0)}(n-1) + \frac{\hat{b}}{\hat{a}_{2}}$

$$(4.64)$$

The *P*-step ahead prediction $x^{(0)}(k+P) = x_p^{(0)}(P) = \hat{x}_p^{(0)}(p-2)$. The relationship of $x_p^{(0)}(P)$ and $x_p^{(0)}(p)$ is shown in Figure 4.2.



Figure 4.2: Relationship of Raw Data and Prediction in Grey Model

D. Weighted Decisions of Grey Model Prediction

The GM(1,1) is suitable for modeling and predicting the data sequence which shown as monotonic trend. GM(1,1) shows robust performance in noise environment, but it is low accurate for modeling non-monotonic data sequence. While the GM(2,1) which is considered with second order structure is suitable for modeling and predicting the data which shown as non-monotonic trend. GM(2,1) can be used in process with wavelike trend, but it is sensitive for the noise in the data sequence. To combine the advantages of the GM(1,1) and GM(2,1), the decision making tool for weighting prediction calculation is proposed. The combined grey model prediction can be more accurate and more adaptable for process variable prediction.

In grey model theory, $-\hat{\theta}_{GM11}(\hat{a})$ called development coefficient is produced from GM(1,1). If process variables show large fluctuations or non-monotonic, the development coefficient will have obvious changes. To describe easily, the development coefficient $-\hat{\theta}_{GM11}(\hat{a})$ is represented as $\gamma_{dco}(k)$. Leveraging $\gamma_{dco}(k)$, the trend of the collected data can be indicated, and GM(1,1) and GM(2,1) can be selected based on different trend.

To balance and combine the advantages of the GM(1,1) and GM(2,1), the weighting algorithms utilizing $\gamma_{dco}(k)$ are developed in this thesis as grey model prediction decision making tool. A computed weight W_{GM} is used for weighting the prediction value between the prediction from GM(1,1) and GM(2,1). The weight W_{GM} ($0 \le W_{GM} \le 1$) represents

change amplitude of the development coefficient $\gamma_{dco}(k)$, which is calculated in equation (4.65)

$$\begin{split} \gamma_{diff}(k) &= abs[\gamma_{dco}(k) - \gamma_{avg-dco}(k)] \\ \gamma_{dev}(k) &= sqrt(\gamma_{var-dco}(k)) \\ if \gamma_{diff}(k) &> \gamma_{dev}(k) \\ W_{GM} &= (\gamma_{diff}(k) - \gamma_{dev}(k)) / \gamma_{diff}(k) \\ else \\ W_{GM} &= 0; \end{split}$$

$$(4.65)$$

where the mean value of $\gamma_{dco}(k)$ is $\gamma_{avg-dco}(k)$ calculated via (4.25), the variance of $\gamma_{dco}(k)$ is $\gamma_{var-dco}(k)$ calculated via (4.28).

From equation (4.65), W_{GM} will increase if $\gamma_{dco}(k)$ is changed dramatically when the process values fluctuate dramatically and show non-monotonic. Contrarily, W_{GM} will decrease even to 0 if $\gamma_{dco}(k)$ is changed small when the process values show monotonic.

The proposed grey model prediction decision making tool for final prediction value are weighted from GM(1,1) and GM(2,1) by W_{GM} . The final P-step ahead prediction is calculated via equation (4.66)

$$\hat{x}_{GMComp}^{(0)}(P) = (1 - W_{GM}) \cdot \hat{x}_{GM11p}^{(0)}(P) + W_{GM} \cdot \hat{x}_{GM21p}^{(0)}(P)$$
(4.66)

In default status, the combined prediction is equal to the predicted value from GM(1,1). GM(1,1) can overcome the noise in the process value. If process values fluctuate dramatically and show non-monotonic, the predicted value from GM(2,1) will be more weighted in combined prediction. While the system variables are monotonic changes, the weight W_{GM} may equal to 0, and GM(1,1) will be more weighted. Therefore, the advantages of the GM(1,1) and GM(2,1) can be balanced and the more accurate prediction value can be provided.

4.4 Summary

This chapter presents the specifications of integrated process fault detection and variable prediction functions. To achieve the two functions, the algorithms for process fault detection and variable prediction functions which are investigated and summarized in Table 4.4. To detect faults in process system, ARX model and EWRLS parameter estimation are combined to generate residuals. Two-sided CUSUM is utilized to evaluate residuals for fault detection. Prediction has two methods. If external inputs of the monitored system are unknown, the grey models, least squares parameters estimation as well as multistep iteration are used for further step prediction of system response. If external inputs of the monitored for the system, and Kalman predictor as well as multi-step iteration are combined for further steps prediction of system response. Both groups of algorithms needs to be implemented into NCS-TT105.

	Algorithms				
	Process Fault Detection	Variable Prediction without Exogenous Variables	Variable Prediction with Exogenous Variables		
Model	ARX	GM(1,1) and GM(1,2)	ARX		
Parameter Estimation	EWRLS (residual generation)	Least Squate (LS)	EWRLS		
Functions	Two-sided CUSUM (residual evaluation)	Multi-step iteration for further steps prediction	Kalman predictor for one-step prediction & Multi-step iteration for further steps prediction		

Table 4.4: Algorithms Summary of Process Fault Detection and Variable Prediction

Chapter 5

5 Implementation of Smart Functions in Real-time Operating System

To implement process fault detection and variable prediction functions in the smart transmitter, the algorithms are grouped into process fault detection task and variable prediction task and embedded into the Nucleus RTOS of NCS-TT105. Specific parameters, procedures, and equations are summarized and collated to better clarify the two groups of algorithms. These summaries offer clear guidance for programming the code of the algorithms using either Matlab or C programming. Furthermore, to organize and schedule those algorithms in RTOS, the design of parameters for scheduling three tasks which are bidirectional communication task, process fault detection task, and variable prediction task in Nucleus RTOS is discussed.

5.1 The Process Fault Detection Task

Heaters, chillers, and pipeline are widely used equipment in industrial plants and factory production lines. The faults in such equipment may result in the production line shutdown or decrease productivity. The targets of the fault detection task in the smart transmitter are to detect faults in the plant process system which is shown in Figure 5.1. Such faults may be heater tripping, chiller shut down, pipeline leak, pipeline plugged, or sensing elements unplugged.

To implement the algorihtms in fault detection task and scheduled by Nucleus RTOS in NCS-TT105 transmitter, design and configuration of the fault detection task are disscussed in details. The overview diagram of the algorithms, including EWRLS and CUSUM for process fault detection functions, is shown in Figure 5.1.



Figure 5.1: Implementation of Process Fault Detection in RTOS on NCS-TT105

5.1.1 EWRLS for Residual Generation

The ARX model and EWRLS algorithm involve three data sources. $U_1(k)$ and $U_2(k)$ are the input data sources of the ARX model and EWRLS and the inputs of the process system, respectively. Y(k) which is the response of the process system is also the input source of the EWRLS.

Because many algorithms use recursive methods, the initial value and parameters of the algorithms are essential. The delay parameters of two inputs for ARX model are denoted as d_1 and d_2 . These two parameters need to be set based on the dead-time of the process response corresponding to the two respective inputs. The dead-time of the process response can be tested using the step response from each input, and d_1 and d_2 can be calculated via each dead-time divided by sample time of the algorithms. The *m* dimension

is set as 5, which means that the orders of the model to describe the system is 5. The model with 5 orders can describe vast majority of systems. Furthermore, the forgetting factor λ_{avg} of a recursive exponential forgetting mean value equation (4.25) for an online computing operating point is set as 0.99, which means that it is used for an on-line computing operating point that is changing slowly. The forgetting factor in variance equation (4.28) is set to 0.98. To balance the tracking speed of the parameter changes and overcome the disturbance from system noise, the EWRLS's forgetting factor λ_w is set as 0.98. Moreover, the initial value of the inverse correlation matrix of EWRLS can be set as $P_w(0) = \alpha I_{3m}$, $\alpha = 10000$, while the initial parameter vector of EWRLS is preset as $\hat{\theta}(0) = 0$. These parameters are estimated through EWRLS equations from (4.10) to (4.13). Finally, the parameter vector $\hat{\theta}(k)$ and residual e(k) are the EWRLS's outputs. Those outputs are computed in every recursive step. Furthermore, the parameters in $\hat{\theta}(k)$ and ARX models represent the model of the estimated process, and can be transferred into state-spaces model for the Kalman predictor being used. The generated residual e(k) is evaluated using a two-sided CUSUM for fault detection.

To further analyze execution time, the computational complexity of the core procedure is shown on the right-hand side. The computational complexity can be used for estimating execution time in the corresponding microprocessor, as well as for comparing it with further optimal algorithms. Moreover, the inputs need to be monitored to prevent EWRLS from becoming sensitive or even unstable when the inputs are not exciting enough. In cases where the excitation of the inputs is very small, the forgetting factor λ_w needs to be time-variant and adjusted to a value close to 1.

The implementation procedures for EWRLS parameters estimation algorithms which are used for residuals generation are shown in Table 5.1.

Residual generation method	×&÷	+&
ARX with two inputs		
EWRLS methods for parameter estimation		
Basic:		
ARX model with 2 inputs and <i>m</i> dimension with each input $y_u(k) + a_1 y_u(k-1) + + a_m y_u(k-m)$		
$=b_1u_1(k-d_1-1)++b_mu_1(k-d_1-m)$		
$+c_1u_2(k-d_2-1)++c_2u_2(k-d_2-m)$		
The noise in the monitored systems is assumed as white noise.		
$\psi^{T}(k) = [-y_{u}(k-1),, -y_{u}(k-m)],$		
$u_1(k-d_1-1),,u_1(k-d_1-m),$		
$u_{2}(k-d_{2}-1),,u_{2}(k-d_{2}-m)$		
$\hat{\theta} = [\hat{a}_1 \dots \hat{a}_m \hat{b}_1 \dots \hat{b}_m \hat{c}_1 \dots \hat{c}_m]^T$		
Dimension:		
Vector $\hat{\theta}$ is $3m \times 1$; Vector $\psi^T(k)$ is $1 \times 3m$;		
Vector $\boldsymbol{\gamma}_{w}(k)$ is $3m \times 1$;		
Matrix $P_w(k)$ is $3m \times 3m$.		
$M_{R} = 3m = 15$		
Inputs:		
$U_1(k)$: The input I of the system		
$U_2(k)$: The input 2 of the system		
Y(k): The response of the system		
Initialization:		
$\lambda_{avg} = 0.99$, $\lambda_{var} = 0.98$		
$P_{w}(0) = \alpha I_{3m}, \alpha = 10000, \qquad \hat{\theta}(0) = 0$		
$\lambda_w = 0.98$		
$d_1 = 8s / sample time, d_2 = 4s / sample time$		

Table 5.1: The procedure of EWRLS Parameter Estimation for Residual Generation

E.

5.1.2 CUSUM for Residual Evaluation

Residual e(k), which is generated from EWRLS is the input of the two-sided CUSUM algorithm. The dynamic reference values of drift and threshold, taken from equations

(4.20) and (4.21), are automatically adjusted using online calculating $Avg(Abs(e_k))$ and $Sqrt(Var(e_k))$ via equation (4.25) and (4.28) respectively. The output of the two-sided CUSUM is the alarm signal CUAlarm(k). Once the cumulative values $g_1(k)$ or $g_2(k)$ are greater than the alarm threshold, the CUAlarm(k) is set to 1, which means the process fault is detected. After the alarm indication is generated, the cumulative values $g_1(k)$, $g_2(k)$ and CUAlarm(k) are reset to 0 for continuous accumulative calculation.

The initial cumulative values are set as $g_1(0) = 0$, $g_2(0) = 0$. The mean value is calculated via the recursive exponential forgetting method from equation (4.25). The mean forgetting factor λ_{avg} is set as $\lambda_{avg} = 0.99$, which is suitable for checking variation from the slowly changing residual e_k . The variance value is calculated using equation (4.28) with a recursive exponential forgetting method. The variance forgetting factor λ_{var} is set as $\lambda_{var} = 0.98$. The drift threshold ratio K_v is set at 0.5, while the alarm threshold ratio K_h is set at 4.0. These two value are determined by the online testing to avoid the false indications or missing indications.

The procedures for residual evaluation by CUSUM algorithms are presented in Table 5.2.

Residual evaluation method	×&÷	+&
Two-sided CUSUM test		
for fault detection alarm indication		
Basic:		
$s_{k}^{(1)} = e_{k}, s_{k}^{(2)} = -e_{k}$ $g_{k}^{(1)} = \max(g_{k-1}^{(1)} + s_{k}^{(1)} - \nu_{k}, 0)$ $g_{k}^{(2)} = \max(g_{k-1}^{(2)} + s_{k}^{(2)} - \nu_{k}, 0)$		
if $g_k^{(1)} > h \text{ or } g_k^{(2)} > h$		
Alarm Indication = 1		
reset $g_k^{(1)} = 0$, $g_k^{(2)} = 0$		

Table 5.2: The Procedure of Two-sided CUSUM for Residual Evaluation

$K \sim C \operatorname{rest}(V_{res}(z))$		
$U_k = K_{v} \times Sqrt(var(e_k))$		
$n_k = \mathbf{K}_h \times Avg(Abs(e_k))$		
e(k)		
Initialization:		
$g_1(0) = 0; g_2(0) = 0$		
$\lambda_{avg}=0.99$, $\lambda_{var}=0.98$		
$K_{\nu} = 0.5$ for drift; $K_{h} = 4$ for alarm		
$e_{avg}(0) = e(k); e_{var}(0) = 0$		
CUAlarm(k) = 0		
M .		
Main program:		
<i>For k=1,2</i>		
$S_1(k) = e(k); S_2(k) = -e(k)$		
$e_{avg}(k) = \lambda_{avg} e_{avg}(k-1) + (1-\lambda_{avg})e(k)$	2	1
$2\lambda_{yar} - 1$ (<i>b</i> 1) + (1 - 2) [-(<i>b</i>) (<i>b</i> - 1)] ²	3	1
$e_{var}(k) = \frac{1}{\lambda_{var}} e_{var}(k-1) + (1 - \lambda_{var})[e(k) - e_{avg}(k-1)]$	1	
$\upsilon(k) = K_{\upsilon} \times Sqrt(e_{var}(k))$	1	
$g_1(k) = \max(g_1(k-1) + s_1(k) - \upsilon(k), 0)$		
$g_2(k) = \max(g_2(k-1) + s_2(k) - \upsilon(k), 0)$		6
$h(k) - h(k) - K \times Abs(a(k))$		
$n_1(\kappa) - n_2(\kappa) - n_h \wedge nos(e_{avg}(\kappa))$	1	
if $g_1(k) > h_1(k)$ or $g_2(k) > h_2(k)$	1	
CUAlarm(k) = 1; Alarm for fault dection		2
reset $g_1(k) = 0, g_2(k) = 0$		
else		
CUAlarm(k) = 0		
Output:		
CUAlarm(k) 1: Alarm for fault dection		
Computational Complexity		
\overline{x}		
I. 10		

Referencing the implementation procedures shown in Table 5.1 and Table 5.2, the algorithms are programmed in Matlab and C code. The embedded C code of fault detection is programmed the fault detection task in Nucleus RTOS. The inputs of the fault detection task have three channels including $U_1(k)$, $U_2(k)$, and Y(k). The time period for the fault detection task can be set at 400ms, which not only produces a fast sample time for data acquisition, but a balanced workload for microprocessor running as well. Furthermore, based on the computational complexity in Table 5.1 and Table 5.2, the estimated execution time of the 40MHz ARM7 MCU in NCS-TT105 transmitter for executing the fault detection algorithms is estimated as 70ms.

5.2 The Variable Prediction Task

The variable prediction information which can forsee the variable changes further than the dead-time of the process system is the valuable information which is extracted from the collected data by variable prediction task in NCS-TT105 transmitter. For instance, the dead-time of the system response according to the inputs of the system is 8s, the 10s system response prediction results can forsee the system response after 10s later, enabling impacts of the dead-time can be eleminated or reduced. The protection actions against to system abnormal changes can be taken timely.

The overview diagram of the algorithms including the Kalman predictor and grey models are shown in Figure 5.2. The initial values, inputs, outputs, and detailed equations of the prediction task are summarized in this section.



Figure 5.2: Implementation of Variable Prediction in RTOS on NCS-TT105

5.2.1 Kalman Predictor

The main procedures used to implement prediction via Kalman predictor include four steps. Firstly, the structure of the system models A(k), B(k) are described by the ARX model. As shown in Figure 5.2, the inputs of the ARX model are $U_1(k)$ $U_2(k)$ and Y(k). Next, the parameters of the ARX model are estimated using EWRLS. The procedure of EWRLS for parameter estimation can be the same or similar as in section 5.1.1. However, the parameters and initial value of the ARX and EWRLS are not the same as the setting in section 5.1.1. For instance, the λ_w is 0.92 to enhance the tracking performance of the EWRLS parameter estimation, while the dimension *m* is set to 3 to use model with 3 orders for the prediction. Secondly, Kalman predictor computes onestep states prediction using the multiple inputs Companion-form of the state-space matrix, which is transferred from ARX mode and combined with the system input data and output response. Thirdly, system model and the one-step predicted states are used to iteratively compute predictions for further steps. To predict the P_{KP} -step of the process variables, the one-step states prediction is used for the $(P_{KP} - 1)$ -step further states prediction via iterative computation. Finally, the $(P_{KP} - 1)$ -step ahead system response prediction can be obtained by $Cx_p(P_{KP})$, where $x_p(P_{KP})$ is the P_{KP} -step predicted states.

As an adaptive algorithms, the initial value and parameters of the Kalman Predictor are essential. The initial value of the states $\hat{x}(0)$ is set as 0. The initial value of the correlation matrix is set as the identity matrix with the dimension $M_{KF} \times M_{KF}$. Regarding the two noise correlation matrices, the correlation matrix of measurement noise R is set to 0.3^2 , while the correlation matrix of process noise Q is set to $0.2^2 I_{M_{KF}}$ with $M_{KF} \times M_{KF}$ dimension. R is referred from the specification of NCS-TT105 using RTD PT100, which is shown in Table 3.2. Q can be determined using various statistical methods according to different scenarios. The prediction step is set as $P_{KP} = 10$, which can also be adjusted for different prediction requirements.

The implementation procedure of Kalman predictor algorithm is shown in Table 5.3. The computational complexity of core procedures is shown on the right-hand side of Table 5.3. Since the matrix calculations involved in the Kalman predictor procedure, the computational complexity needs to be carefully considered. Given the characteristic of the Companion-form of A(k), the matrix calculation can be optimized and simplified.

One-step Kalman Predictor	×&÷	+&
Variable prediction with inputs of the system		
Basic:		
Using <i>m</i> dimension ARX and EWRLS to estimate $\hat{\theta}$		
$\hat{\theta} = [\hat{a}_1 \dots \hat{a}_m \hat{b}_1 \dots \hat{b}_m \hat{c}_1 \dots \hat{c}_m]^T$		
$u_1(k) = U_1(k) - U_{1op}(k)$		
$u_2(k) = U_2(k) - U_{2op}(k)$		
$y_u(k) = Y(k) - Y_{op}(k)$		
$u(k) = [u_1(k - d_1 - 1)u_1(k - d_1 - m)]$		
$ u_2(k-d_2-1)u_2(k-d_2-m)]^T$		
ARX model:		
$y_u(k) = -a_1 y_u(k-1) - \dots - a_m y_u(k-m)$		
$+b_1u_1(k-d_1-1)++b_mu_1(k-d_1-m)$		
$+c_1u_2(k-d_2-1)++c_mu_2(k-d_2-m)$		
The noise in the monitored systems is assumed as white noise.		
Matrix:		
$\begin{bmatrix} -a_1 & -a_2 & -a_3 \cdots & -a_{m-1} & -a_m \end{bmatrix}$		
$1 0 0 \cdots 0 0$		
$A(k) = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 & 0 \\ m \times m \end{bmatrix}$		
$\begin{array}{c ccccccccccccccccccccccccccccccccccc$		
$\begin{bmatrix} b_1 & b_2 & \cdots & b_m & c_1 & c_2 & \cdots & c_m \end{bmatrix}$		
$B(k) = \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ \vdots & m \times 2m \end{bmatrix}$		
$C = \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix} 1 \times m$		
$x(k) = \begin{bmatrix} x_1 & x_2 & x_3 & \cdots & x_{m-1} & x_m \end{bmatrix}^T$		

Table 5.3: The Procedure of the Kalman Predictor and Multistep Iteration

x(k+1) = A(k)x(k) + B(k)u(k) + w(k)	
z(k) = Cx(k) + v(k)	
Dimension:	
Vector $x(k)$: $M_{KF} \times 1 = m \times 1$	
Vector $u(k)$: $M_{KFp} \times 1 = 2m \times 1$	
Matrix $A(k)$: $M_{KF} \times M_{KF} = m \times m$	
Matrix $B(k)$: $M_{KF} \times M_{KFp} = m \times 2m$	
Matrix C : $1 \times M_{KF} = 1 \times m$	
Vector $\overline{K}(k)$: $M_{KF} \times 1 = m \times 1$	
Matrix $P^{-}(k)$: $M_{KF} \times M_{KF} = m \times m$	
Correlation matrix:	
$E\{w(k)w^{T}(k)\} = Q, E\{v(k)v^{T}(k)\} = R,$	
Correlation matrix of measurement noise R is 1×1	
Correlation matrix of process noise Q is $M_{KF} \times M_{KF} = m \times m$	
$M_{KF} = m, \ M_{KFp} = 2m, \ m = 3$	
Inputs:	
$U_1(k)$: The input 1 of the system	
$U_2(k)$: The input 2 of the system	
Y(k): The response of the system	
Initialization:	
$\lambda_w = 0.92$ for EWRLS parameter estimation	
$\hat{\theta} = [\hat{a}_1 \dots \hat{a}_m \hat{b}_1 \dots \hat{b}_m \hat{c}_1 \dots \hat{c}_m]^T$ has been determined	
$\hat{x}(0) = 0, P^{-}(0) = I_{M_{KF}}$	
$Q = 0.2^2 I_{M_{\rm eff}}, R = 0.3^2$	
The steps of prediction $P_{KP} = 10$	
Main program:	
For k=1.2	
Kalman Predictor:	

$\overline{K}(k) = P^{-}(k)C^{T}[CP^{-}(k)C^{T} + R]^{-1}$	M _{KF}	1
$\frac{A\overline{K}(k)[y_u(k) - C\hat{x}(k \mid k-1)]}{2}$	$M_{\rm KF}^2$ + $M_{\rm KF}$	$M_{_{KF}}^2 + 1$
$\hat{x}(k+1 k) = A\hat{x}(k k-1) + Bu(k) + \underline{A\overline{K}(k)[y_u(k) - C\hat{x}(k k-1)]}$	M_{KF}^2 +2 M_{KF}	$M_{_{KF}}^2$
$P^{-}(k+1) = AP^{-}(k)A^{T} - AP^{-}(k)C^{T}[CP^{-}(k)C^{T} + R]^{-1}CP^{-}(k)A^{T} + Q$ $= AP^{-}(k)A^{T} - A\overline{K}(k)CP^{-}(k)A^{T} + Q$	3 <i>M</i> ³	$+4M_{_{KF}}$
$= A[I - K(k)C]P(k)A^{*} + Q$ $x_{KPIst} = \hat{x}(k+1 k) \text{ is the one-step states prediction}$	KF	$3M_{KF}^{3}$ $+2M_{KF}$
 Further prediction for $P_{KP} - 1$ steps initialization: $x_p(1) = x_{KPIst}$		
For $k=1: (P_{KP}-1)$ $x_p(k+1) = A(k)x_p(k) + B(k)u(k)$ end	$(P_{kp}-1)$ $\times 3M_{KF}$	$(P_{kp} - 1)$ ×3 M_{KF}
The prediction Y_p after P_{KP} steps is $Y_{KPp}(P_{KP}) = Y_{op}(k) + Cx_p(P_{KP})$		1
Output: $Y_{KPp}(P_{kp})$		
The optimized computational complexity $\times \div \approx 3M_{KF}^{3} + 2M_{KF}^{2} + 3P_{kp}M_{KF} + M_{KF}$ $\pm \approx 3M_{KF}^{3} + 2M_{KF}^{2} + 3P_{kp}M_{KF} + 3M_{KF} + 2$		

5.2.2 Grey Model

The main procedures used to implement grey model prediction include three steps. Firstly, the parameters of the GM(1,1) and GM(2,1) are estimated using the LS method in every sampling step. Secondly, the multiple steps of predictions are calculated iteratively using the GM(1,1) and GM(2,1) models. Thirdly, the decision-making tool is used for weighting the prediction GM(1,1) and GM(2,1), used for the final combined prediction value.

The length of the data sequence is set at 20, which can be adjusted corresponding with different practical applications. The parameters vector of GM(1,1) is a 2×2 dimension, while the parameters vector of GM(2,1) is a 3×3 dimension. The steps of the prediction are set as $P_{GM} = 10$. v_{psft} is set to 2, which means that the initial values for further prediction are selected from the last two data from the data sequence. Moreover, the $\lambda_{avg} = 0.99$ and $\lambda_{var} = 0.98$ are used as forgetting factors for online mean and variance calculation in decision-making tool. The implementation procedures of the GM(1,1) and GM(2,1) prediction is shown in Table 5.4.

Gray Model for Prediction	× P. ·	Pr
City Model for Trediction	$\wedge \alpha \overline{\cdot}$	$\pm \alpha$
Variable prediction without inputs of the system		
Basic		
$X^{(1)} = (x^{(1)}(1), x^{(1)}(2), \cdots x^{(1)}(n))$		
$(1) \mathbf{x}(0) ((1) (0) (2) (1) (0) (2))$		
$\alpha^{(*)} X^{(*)} = \{\alpha^{(*)} x^{(*)}(2), \dots, \alpha^{(*)} x^{(*)}(n)\}$		
$7^{(1)}$ (-(1)(2) -(1)(2) -(1)())		
$Z^{(n)} = (Z^{(n)}(2), Z^{(n)}(3), \cdots Z^{(n)}(n))$		
$X_{p1}^{(0)} = (\hat{x}_{p1}^{(0)}(1), \hat{x}_{p1}^{(0)}(2), \cdots x_{p1}^{(0)}(P_{GM21} + v_{psft}))$		
$X_{n2}^{(0)} = (\hat{x}_{n2}^{(0)}(1), \hat{x}_{n2}^{(0)}(2), \cdots , x_{n2}^{(0)}(P_{GM21}))$		
r- r- y- y- out-		
$r^{(1)}(i) = \sum_{k=n+i}^{k-n+i} r^{(0)}(i) i=1,2,,n$		
$x (l) = \sum_{k=n+1}^{l} x (l), l = 1, 2, \cdots, ll$		
$\alpha^{(1)} r^{(0)}(i) - r^{(0)}(i) r^{(0)}(i-1) i - 2 \dots n$		
$\alpha x (l) - x (l) - x (l-1), l = 2, \dots, n$		

Table 5.4: The Procedure of Grey Model Prediction

$$\begin{aligned} & For k=n, n+l, n+2, \dots \\ & \text{Acquiring data into data sequence} \\ For i=l:n \\ x^{(0)}(k-n+i) = Measurement(k-n+i) \\ & end \\ & x^{(1)}(i) = \sum_{k=s+i}^{k=s+i} x^{i(0)}(i), i = 1, 2, \cdots, n \\ & z^{(1)}(i) = \frac{1}{2} (x^{(1)}(i) + x^{(1)}(i-1)), i = 2, 3, \cdots, n \\ & GM(1,1) \text{ parameters estimation:} \\ & Y_{GM11} = \begin{bmatrix} x^{(0)}(2) \\ x^{(0)}(3) \\ \vdots \\ x^{(0)}(n) \end{bmatrix}, B_{GM11} = \begin{bmatrix} -z^{(1)}(2) & 1 \\ -z^{(1)}(3) & 1 \\ \vdots \\ -z^{(1)}(n) & 1 \end{bmatrix} \\ & \hat{\theta}_{GM11} = \begin{bmatrix} \hat{a} \\ \hat{b} \end{bmatrix} = \begin{bmatrix} B_{GM11}^T B_{GM11} \end{bmatrix}^{-1} B_{GM11}^T Y_{GM11} \\ & GM(2,1) \text{ parameters estimation:} \\ & Y_{GM21} = \begin{bmatrix} x^{(0)}(2) - x^{(0)}(1) \\ \vdots \\ x^{(0)}(n) - x^{(0)}(n-1) \end{bmatrix}, \\ & B_{GM21} = \begin{bmatrix} -x^{(0)}(2) - x^{(0)}(2) \\ \vdots \\ x^{(0)}(n) - x^{(0)}(n-1) \end{bmatrix} \\ & B_{GM21} = \begin{bmatrix} -x^{(0)}(2) - z^{(1)}(2) & 1 \\ \vdots \\ -x^{(0)}(n) - z^{(1)}(n) & 1 \end{bmatrix} \\ & \hat{\theta}_{GM21} = \begin{bmatrix} \hat{a}_{1} \\ \hat{a}_{2} \\ \hat{b} \end{bmatrix} = \begin{bmatrix} B_{GM21}^T B_{GM21}^T Y_{GM21} \\ -B_{GM21}^T Y_{GM21} \end{bmatrix}^{-1} B_{GM21}^T Y_{GM21} \end{aligned}$$

$$\begin{array}{l} \mbox{The prediction of P_{cM} steps by GM(1,1) model: \\ The GM(1,1) Model $x^{(0)}(i) = a(-z^{(1)}(i)) + b \cdot 1$ \\ can be rewritten as \\ \hline $x^{(0)}(i) = \frac{-\hat{a}}{1+0.5\hat{a}}, \hat{x}^{(1)}(i-1) + \frac{\hat{b}}{1+0.5\hat{a}}, i = 2, 3 \cdots n$ \\ $x^{(1)}(i) = \hat{x}^{(0)}(i) + \hat{x}^{(1)}(i-1)$ \\ initialization: \\ using $x^{(0)}(n - v_{pqh} + 2)$ to estimate $\hat{x}^{(1)}(n - v_{pqh} + 1)$ \\ \hline $x^{(1)}(n - v_{pqh} + 1) = \frac{(1+0.5\hat{a})}{-\hat{a}} \cdot x^{(0)}(n - v_{pqh} + 2) + \frac{\hat{b}}{\hat{a}}$ \\ \hline $x_{GM11p1}^{(0)}(1) = \hat{x}^{(0)}(n - v_{pqh} + 2)$ and $\\ \hline $x_{GM11p1}^{(0)}(1) = \hat{x}^{(1)}(n - v_{pqh} + 1)$ \\ are used as initial value for GM(1,1) prediction $ \\ Main program for prediction iteration: $ \\ For $p = v_{pqh}: P_{GM} + v_{pqh}$ \\ \hline $x_{GM11p1}^{(0)}(p) = \frac{-\hat{a}}{1+0.5\hat{a}}, \hat{x}_{GM11p1}^{(0)}(p-1) + \frac{\hat{b}}{1+0.5\hat{a}}$ \\ \hline $x_{GM11p1}^{(0)}(p) = \hat{x}_{GM11p1}^{(0)}(p) + \hat{x}_{GM11p1}^{(0)}(p-1)$ \\ \hline $x_{GM11p1}^{(0)}(p] = \hat{x}_{GM11p1}^{(0)}(p) + \hat{x}_{GM11p1}^{(0)}(p-1)$ \\ \hline $x_{GM11p1}^{(0)}(p] = \hat{x}_{GM11p1}^{(0)}(p + v_{pqh})$ \\ end $ \\ For $p = 1: P_{GM}$ \\ \hline $x_{GM11p2}^{(0)}(P] = \hat{x}_{GM11p1}^{(0)}(p + v_{pqh})$ \\ end $ \\ \hline $x_{GM11p2}^{(0)}(P_{GM})$ is the prediction from GM(1,1) $ \\ The prediction of P_{GM} steps by GM(2,1) model: $ \\ The GM(2,1) Model $a^{(1)}x^{(0)}(i) + a_{1}x^{(0)}(i) + a_{2}z^{(1)}(i) = b$ \\ can be rewritten as $ \\ \hline \end{aligned} \end{al$$

$\gamma_{avg-dco}(k) = \lambda_{avg}\gamma_{avg-dco}(k-1) + (1-\lambda_{avg})\gamma_{dco}(k)$ $\gamma_{avg-dco}(k) = \frac{2\lambda_{var}-1}{\gamma}\gamma_{avg-dco}(k-1) + (1-\lambda_{avg})[\gamma_{co}(k) - \gamma_{co}(k)]^{2}$	3	2
$\gamma_{var-dco}(k) = \lambda_{var} \gamma_{var-dco}(k) - \gamma_{avg-dco}(k)]$ $\gamma_{diff}(k) = abs[\gamma_{dco}(k) - \gamma_{avg-dco}(k)]$ $\gamma_{dev}(k) = sqrt(\gamma_{var-dco}(k))$	1	1
Start:		
$W_{GM}=0;$		
$if \gamma_{diff}(k) > \gamma_{dev}(k)$		
$W_{GM} = \left(\gamma_{diff}\left(k\right) - \gamma_{dev}\left(k\right)\right) / \gamma_{diff}\left(k\right)$	1	2
end		
$\hat{x}_{GMComp}^{(0)}(P_{GM}) = (1 - W_{GM}) \cdot \hat{x}_{GM11p2}^{(0)}(P_{GM}) + W_{GM} \cdot \hat{x}_{GM21p2}^{(0)}(P_{GM})$	2	2
Output:		
P_{GM} steps ahead prediction based on every sampling step k:		
is $\hat{x}_{GMComp}^{(0)}(P_{GM})$		
Computational Complexity		
$\times \div : GM(1,1) \approx 7M_{G11} + P_{GM} - 4$		
$GM(2,1) \approx 10M_{G21} + P_{GM} + 90$		
$\pm : GM(1,1) \approx 8M_{G11} + 2P_{GM} + 1$		
$GM(2,1) \approx 12M_{G11} + 3P_{GM} + 25$		

Following the implementation procedures shown in Table 5.3 and Table 5.4, the embedded algorithms can be programmed and embedded into the prediction task program. The inputs of the prediction task have four channels; three channels, including $U_1(k)$, $U_2(k)$, and Y(k), are used for Kalman predictor algorithms, one channel of *Measurement(k)* is used for the input for grey model prediction. The outputs of the prediction task are predicted variables from two groups of prediction algorithms. The period time for the prediction task can be set to 1000ms, allowing the number of prediction steps can be the same as the proceeding seconds of prediction. Furthermore, based on the computational complexity in Table 5.3 and Table 5.4, the estimated

execution time of the 40MHz ARM7 MCU in NCS-TT105 transmitter for executing the prediciton algorithms is estimated as 90ms.

5.3 The Bidirectional Communication Task

It can be seen from the previous sections that $U_1(k)$ and $U_2(k)$, which are the inputs of the observed system, are needed for the tasks of both fault detection and prediction. This means that besides measuring system response, the designed smart sensor also needs to acquire external data from other sources. The smart transmitter therefore needs to be equipped with bidirectional communication function in order to realize exchanging data and information.

The original PROFIBUS-PA Fieldbus communication function in NCS-TT105 transmitter only transmits two channel measurements to upper layer systems. Implementing bidirectional communication in this smart transmitter requires modifying its inside PROFIBUS-PA profile and the GSD file used in upper layer controllers. The modified communication profile programs are grouped in the bidirectional communication task.

As introduced in section 3.3.1, Function Blocks (FBs) of the PA device profile in a NCS-TT105 are the interface blocks for exchanging data with the network [101]. Analog Input FBs, and Analog Output FBs, shown in Figure 3.4, are commonly used FBs for this purpose. Each communication data channel corresponds to a particular FB. The original NCS-TT105 only transmits two channels of measured data using two Analog Input FBs. To transmitter more data and information from NCS-TT105 to upper-level systems, Analog Input FBs need to be added to transfer the outputs of algorithms such as alarm indication and prediction results. While, in order to collect data from other devices and systems, Analog Output FBs must be established in PA devices for receiving data from the Fieldbus. The received data can be the initial parameters and the input data sources of the embedded algorithms.

These modifications of the PA devices profile are implemented in the bidirectional communication task. The increased Analog Input FBs and Analog Output FBs are

programmed in this task, with the communication interface including $16 \times AI$ and $8 \times AO$ channels (a list of these communication channels is shown in Table 5.5). Because the communication task is the foundational task for the NCS-TT105, it should have the highest priority, with a period time of 200ms, which is the shortest of all the tasks. Since modification of the communication task is not the core of this research, this thesis will not show the details of the implementation procedures, though a more detailed introduction of the PROFIBUS-PA profile is summarized in the Appendix B.

Data Direction	Channel Number	Name	Format	Comments	Unit
	0	Channels 1 measurement	Float	The 1st channel of sensor measurement	°C
	1	Channels 2 measurement	Float	The 2nd channel of sensor measurement	°C
	2	EWRLS_u1	Float	1st input of EWRLS	%
	3	EWRLS_u2	Float	2nd input of EWRLS	L/min
Survey Courses	4	EWRLS_y	Float	Reference value of EWRLS	°C
-> DCS	5	EWRLS_FD_Residuals	Float	Residuals generation from EWRLS	°C
Analog Inputs	6	CUSUM_G1	Float	CUSUM positive cumulation	
FBs' channels	7	CUSUM_G2	Float	CUSUM nagative cumulation	
	8	CUSUM_Threshold_h	Float	CUSUM dynamic threshold	
	9	Fault Indication	Float	Fault detection indication	0/1
	10	EWRLS_KF_Residuals	Float	Residuals from ARX model for Kalman predictor	°C
	11	GM_Prediction	Float	Prediction of combined GM(1,1) and GM(2,1)	°C
	12	KF_Prediction	Float	Prediction of Kalman predictor and multi-steps iteration	°C
	13	Combined_Prediction	Float	Combined Prediction from Kalman predictor and Grey model	°C
	14	Prediction_Validation	Float	Validation(%) of combined prediction	%
	15	GM_Prediction_Allocation	Float	Weight W _{GM} (%) of Grey model prediction	%
	0	External Inputs 1	Float	1st input of obserered system	%
	1	External Inputs 2	Float	2nd input of obserered system	L/min
DCS ->	2	Delay_In1	Integer	Delay parameter of Input1	Number
Smart Sensor	3	Delay_In2	Integer	Delay parameter of Input2	Number
Analog Outputs	4	CUSUM_Kh	Float	CUSUM threshold ratio	Number
TBS channels	5	ForgettingFactor_EWRLS	Float	Forgetting Factor of EWRLS	Number
	6	Prediction Steps	Integer	The steps of number for prediction	Number
-	7	GM_N	Integer	The data length of Grey model sequence	Number

Table 5.5: List of Communication Interface of PROFIBUS-PA in NCS-TT105

To allow the upper layer controller to identify the modified NCS-TT105, the GSD file must also be modified corresponding with the increased input and output channels. Once the updated file is configured into the Fieldbus controller, the controller can identify and communicate with the NCS-TT105 via 16 × AI channels and 8 × AO channels.
As shown above, the bidirectional communication task is implemented via the modification of the PROFIBUS-PA profile and GSD file. The original interface of twochannels PROFIBUS-PA communication in NCS-TT105 is extended to include both $16 \times AI$ and $8 \times AO$ channels, enabling the embedded algorithms in NCS-TT105 can exchange data and information with other systems.

5.4 Implementation of Multitasking in the Real-time Operating System

To execute fault detection, prediction, and bidirectional communication all in real-time, the embedded Nucleus RTOS in an NCS-TT105 plays the key role of guaranteeing the event response time and period time of the tasks. To achieve the requirements of the period time for the three tasks which are 200ms for bidirectional communication task, 400ms for process fault detection task, and 1000ms for variable prediction task (as shown in Figure 5.3), the priority, period time, and quantity of the tasks, as well as an RTOS kernel tick, must be designed carefully (some terminologies related to Nucleus RTOS, including kernel tick, task state machine, preemptive scheduling, and task priority are explained in Appendix C). All the algorithms are scheduled in the Nucleus RTOS, which is assigned in the networking and application unit (NAAU) and processed by the ARM7 MCU in NCS-TT105 transmitter.

Profibus PA Fieldbus Network



Figure 5.3: Multitasking Implementation of Three Tasks in Nucleus RTOS

5.4.1 Configuration of Multitasking Scheduling with RMS Rules

According to the review in section 2.4, RMS offers optimum scheduling solutions among other priority-based preemptive static scheduling policies. RMS policies for scheduling multitasks are followed in configuring the parameters of the Nucleus RTOS.

The RMS, which is known as the rate-monotonic priority assignment, indicates that the tasks with shorter period time are assigned higher priorities [87]. The relationship between the task period time and the priorities in RMS can be expressed as equation(5.1).

$$Periority = \frac{1}{PeriodTime}$$
(5.1)

Moreover, all the assumptions mentioned in [87] for using the RMS rules can be guaranteed during the implementation of NCS-TT105. The cost in time of RTOS kernel switching and hardware interruption are ignored. Some timing terminologies and key words should be clarified.

The terminologies of timing and key words [20]

- T_e : Execution time of a task
- T_{v} : Period time of a task
- T_r : Response time of a task
- T_d : Deadline of a task
- T_{int} : Serving interrupts time of a task
- T_{ever} : Event response time of a task
- T_{Tick} : Kernel tick time of RTOS
- N_{Tick} : Numbers of kernel tick
- WL: Microprocessor workload for running tasks

The execution time T_e , which is a dynamic time for online running, can either be obtained from a load-test or estimated from computational analysis. The period constant time T_p refers to the request rate or invocation interval of a task. The response time T_r of a task is the time between the starting point of the current task period and the starting point of the task execution. The event response time T_{ever} is the time between the occurrence of an event and the response time of the task. The deadline T_d is the time from a task's ending point to the end of the current period. The relationship of all these timings is shown in equation (5.2), while Figure 5.4 shows more clearly the timing terminologies.

$$T_p = T_r + T_e + T_{int} + T_d \tag{5.2}$$



Figure 5.4: Timing Terminologies of a Task in an RTOS System Three essential rules using RMS are discussed as follows:

Task event response time: The event response time T_{ever} of a task is an important criterion for analyzing whether a task can achieve the application requirements. Because the starting point of a task may be delayed by an interruption from higher priority executing tasks, the response time of a task is dynamic variable. Therefore, the worst T_{ever} is commonly used in evaluation criterion. If the worst T_{ever} can be accepted, then the design of multitasking can achieve the requirements of the application.

The worst event response time T_{ever} of the task is combined with both the worst task response time T_r of a task and the period time T_p of a task. Thus the worst event response time T_{ever} of the *n*th task is expressed as (5.3)

$$T_{n ever} \leq worst \ T_{n ever}$$
where worst $T_{n ever} = T_{np} + T_{nr} = T_{np} + \sum_{i=n-1}^{i=n-1} [T_{ie} + T_{i int}]$
(5.3)

where the worst task response time T_{nr} of *n*th task is the sum of the execution time and interruption time from the numbers of *i* higher priority executing tasks. If the event response time T_{ever} of a task is longer than the requirement, the priority, execution time and the period time of the task have to be reconsidered and optimized, sometimes resulting in a redesign of the parameters of the RTOS or even an upgrade of the microprocessors. The event response time T_{ever} of the three implemented tasks are analyzed in the next subsection.

<u>RTOS kernel tick setting:</u> The kernel tick T_{Tick} is the heartbeat of an RTOS kernel which is introduced in Appendix C. The parameter of the kernel tick has a close relationship with the switch period of the task state machine, task period time, and RTOS kernel workload. Following the rules of RMS, there are two configuration scenarios of kernel tick setting.

A. Scenario-1: $\sum T_{ne} < GCD[T_{np}]$

If the greatest common divisor (GCD) of all the tasks' period time is greater than the sum of the execution time of all tasks, the kernel tick can be configured in between these two values[20]. The rule for kernel tick in scenario-1 is shown in (5.4).

$$if \sum [T_{1e}, T_{2e}, \cdots, T_{ne}] \leq GCD[T_{1p}, T_{2p}, \cdots, T_{np}] \\\sum [T_{1e}, T_{2e}, \cdots, T_{ne}] < T_{Tick} \leq GCD[T_{1p}, T_{2p}, \cdots, T_{np}]$$
(5.4)

In kernel tick scenario-1, all tasks are executed completely in a single kernel tick, and the executing task with lower priority will not be interrupted by the task with higher priority. This result in $T_{i int}$ of the low priority task is 0. The worst event response time $T_{n ever}$ can be determined from equation (5.3), since it is only with the execution time T_{ie} and period time T_{np} . Moreover, since $T_{i int}$ of the task with the lower priority is 0, the worst event response time $T_{n ever}$ of the task in this scenario is the minimum.

The kernel tick rules in scenario-1 are recommended due to their shorter and more accurately determined worst event response time, which enable the $T_{n ever}$ of a task to achieve the application requirements probably. Therefore, the kernel tick rules in scenario-1 is the suggested. Moreover, it should be also noted that drawback of these rules is that the condition from equation (5.4) cannot be met in some applications. The designers must either optimize the program to shorten the execution time of the tasks, or reorganize the time period of the tasks.

B. Scenario-2 -
$$\sum T_{ne} > GCD[T_{np}]$$

In some case, the conditions in scenario-1 can not be met. This allows the designer to select these configuration rules in which the kernel tick time T_{Tick} is less than the minimum of the execution time T_e of all tasks, and the remainder from the division of the greatest common divisor (GCD) of all the tasks' period time by kernel tick time needs to be zero. These rules for the kernel tick in scenario-2 are shown in (5.5) and (5.6)

$$T_{\text{Tick}} \le Min[T_{1e}, T_{2e}, \cdots, T_{ne}]$$
(5.5)

$$Mod(GCD[T_{1p}, T_{2p}, \cdots, T_{np}], T_{Tick}) = 0$$
(5.6)

In this scenario, with the small value setting of the kernel tick, the kernel of RTOS will frequently interrupt lower priority tasks to respond to REDAY tasks with higher priority. This is suitable for the scenario the task is required for, with a short time period and the need for a prompt response by the kernel timely. The conditions in (5.5) and (5.6) can also be met at the same time in most of cases. However, because of the tasks with low priority are frequently interrupted by higher priority tasks, the $T_{i \text{ int}}$ of the task with low priority is not zero. Therefore, the worst task response time T_r and the worst event response time $T_{n \text{ ever}}$ of the task with the lower priority may be expended longer. As shown in Figure 5.4, the T_d of the interrupted tasks may overflow in worst case scenario. Meanwhile, since interrupt service routine (ISR) interrupts tasks for switching to kernel scheduling every tick, the workload of the RTOS kernel increases dramatically.

As shown above, the kernel tick needs to be configured, considering the T_e and T_p of the all the tasks in RTOS, as well as the event response time from the application requirements. The different kernel tick parameters have obvious influence on both event response performance and RTOS kernel workload. In order to describe and analyze this more clearly, two examples based on these scenarios are discussed in the Appendix C.

Processor utilization and workload: To guarantee the designed tasks are workable in RTOS and the microprocessor, the upper bound of processor utilization and current task

workload need to be evaluated. From [87], processor utilization must be less than an upper bound for scheduling the m numbers of tasks with RMS, as expressed in (5.7)

$$WL_{processor} \le U_{processor} = m(2^{1/m} - 1)$$
(5.7)

where the workload of the processor is caculated using equation (5.8)

$$WL_{processor} = \frac{T_{1e}}{T_{1p}} + \frac{T_{2e}}{T_{2p}} + \dots + \frac{T_{me}}{T_{mp}}$$
(5.8)

From (5.7), it can be seen that as the number of tasks, *m*, increases, utilization of processor $U_{processor}$ approaches 70% if RMS is employed. It is therefore recommended that the workload $WL_{processor}$ of the MCU in NCS-TT105 transmitter for running all tasks with the RMS in RTOS should be less than 70%. The execution time T_e and period time T_p of all the tasks need to be designed carefully.

In summary, the design rules for multitasking scheduling in RTOS have been studied in detail. Within RMS rules, the task priority, event response time, kernel tick, and processor workload all require careful analysis and design.

5.4.2 Implementation of Three Tasks in Nucleus RTOS

All the tasks discussed in previous sections are scheduled by the Nucleus RTOS. The name, functions, priority, estimated execution time, and period time of the tasks are all assigned, as shown in Table 5.6. The communication task with a period time of 200ms is named Task11 and assigned with the highest priority 11; the fault detection task, with a period time of 400ms is named Task12 and assigned priority 12; the prediction task has a period time of 1000ms is named Task13, assigned the lowest priority of 13. According to the analysis of computational complexity from section 5.1 and 5.2, the estimated execution time of Task12 and Task13 for ARM7 MCU in NCS-TT105 are 70ms and 90ms, respectively. Moreover, since the programs involved in the communication task are not added too much with the original programs of the ARM7 MCU, the estimated execution time of Task11 is referenced from Microcyber, at about 30ms.

	Task11	Task12	Task13
Functions	Bidirectional communication	Process fault detection	Variable prediction
Priority	11	12	13
Estimated Execution Time (ms)	30	70	90
Period Time (ms)	200	400	1000

Table 5.6: Three Tasks for Integration in RTOS

From the estimated execution time shown above, the sum of the execution time of the three tasks is 190ms. The total execution time is less than the GCD of all three tasks' period time, as shown in equation (5.9). According to the equation (5.4) in scenario-1 of the kernel tick setting rules, the kernel tick of Nucleus RTOS is set as 200ms.

$$\sum [\hat{T}_{11e}, \hat{T}_{12e}, \hat{T}_{13e}] = 190 < T_{Tick} \le GCD[T_{11p}, T_{12p}, T_{13p}] = 200ms$$
(5.9)

The task flow of multitasking scheduling with a 200ms kernel tick is shown in Figure 5.5. After the RTOS starts up, the three tasks run independently according to their assigned time periods. Every 200ms, the RTOS kernel executes ISR, and the highest priority READY task is executed (the five task states of Nucleus RTOS, including EXECUTING, READY, SUSPENDED, TERMINATED and FINISHED are introduced in Appendix C). Every 1000ms, the three tasks gather together due to the requirement that period time and needs be processed within a single kernel tick interval. As shown in Figure 5.5, according to the rules of the kernel tick established in scenario-1, the three tasks are scheduled in priority sequence within one kernel tick interval.

Kernel Tick = 200ms, Every slot = 10ms																								
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Task11																								
Task12																								
Task13																								

Figure 5.5: Task Flow of Multitasking Scheduling in RTOS

According to equation (5.7), the upper bound of processor utilization is 77.98%. From (5.8), the workload of the microprocessor is 41.5%, which is calculated from (5.10).

$$WL_{processor} = \frac{30}{200} + \frac{70}{400} + \frac{90}{1000} = 41.5\%$$
 (5.10)

According to equation (5.3), the worst event response time for each of the three tasks is 230ms, 500ms, and 1140ms respectively, which are calculated in (5.11). The event response time of all tasks can clearly be determined.

$$T_{1 ever} \leq T_{1p} + T_{1r} + T_{1 int} = 200 + 0 + 0 = 230$$

$$T_{2 ever} \leq T_{2p} + T_{2r} + T_{2 int} = 400 + 30 + 0 = 430$$

$$T_{3 ever} \leq T_{3p} + T_{3r} + T_{3 int} = 1000 + (30 + 70) + 0 = 1100$$
(5.11)

To implement period time for three tasks, the NU_Sleep() [95] function in Nucleus RTOS is used at the end of each task. The effects of NU_Sleep() for realizing period time of the task is shown in Figure 5.6. The NU_Sleep(N_{Tick}) is used for suspending the calling task for the number of kernel ticks. Each task's period time is configured by the numbers of ticks. When NU_Sleep() is called, the kernel will transfer the task to a SUSPENDED state. After the numbers of the set kernel tick, the task will be transferred to the READY state. Leveraging NU_Sleep(), the accurate period time of a given task can be determined by kernel tick and N_{Tick} .



Figure 5.6: Program Context Refer to NU_Sleep() and Kernel Tick

Finally, as shown in Figure 5.3 and Figure 5.6, the multitasking of the three tasks is implemented in the Nucleus RTOS. The summary of this implementation is shown in Table 5.7. The kernel tick is set to 200ms. NU_Sleep(1) is called at the end of task 11 to implement the 200ms period time. NU_Sleep(2) is called at the end of task 12 to implement the 400ms period time, and NU_Sleep(5) is called at the end of task 13 to implement the 1000ms period time. The program code is programmed in MULTI programming software, and is compiled as a firmware file. To embed the code into NCS-TT105, the firmware is downloaded into the ARM7 MCU in the NCS-TT105. Finally all the algorithms are embedded into the RTOS of the NCS-TT105, allowing NCS-TT105 to realize smart functions in practical industrial systems.

To clarify the different name between the new NCS-TT105 and conventional NCS-TT105, the enhanced NCS-TT105 which is integrated with smart functions is named SMART NCS-TT105. The name SMART NCS-TT105 which means smart algorithms embedded and smart functions integrated is used in the following chapter.

	Task11	Task12	Task13
Functions	Bidirectional communication	Process fault detection	Variable prediction
Priority	11	12	13
Estimated Execution Time (ms)	30	70	90
Period Time (ms)	200	400	1000
Worst Event Response Time (ms)	230	430	1100
Workload (%)	15.0	17.5	9.0
NU_Sleep(N) kernel tick = 200ms	1	2	5

Table 5.7: Implementation Summary of Three Tasks

5.5 Instructions of using the Designed Smart Sensor

Considering the conditions of the embedded algorithms, to apply the designed SMART NCS-TT105 into practical applications, some instructions of using the designed functions in practical process are summarized as following:

Inputs, outputs and initial parameters of two smart functions: Table 5.8 and Table 5.9 are reference lists which could be used to set the data interfaces and parameters of the internal functions in SMART NCS-TT105 for various process system applications. Especially the initial parameters of the algorithms which can be modified by the specific monitored process system are summarized in these two tables.

The discrete dead-time parameters d_1 and d_2 can be obtained via the inputs stepresponse test into the system. If the d_1 and d_2 are dynamic parameters, the advanced methods for determining the dead-time parameters should be utilized and bumpless transfer process of the parameter estimation should be considered.

The correlation matrix of measurement noise R can be referenced from the technical specification in the documents of the applied sensor. Regarding the correlation matrix of

process noise Q, the higher parameters in Q comparing with R, more weight to the measurements and the estimation accuracy is compromised.

	The Smart Function of Process Fault Detection in Task12								
	$U_1(k)$:	The input 1 of the system							
Inputs:	$U_{2}(k)$:	The input 2 of the system							
	Y(k):	The response of the system							
	dimension of ARX $m=5$	\geq the order of the process system							
	$d_1 = 8s / sample time$	The dead-time for input 1							
	$d_2 = 4s / sample time$	The dead-time for input 2							
		Sample time = 400ms (period time of the task)							
	$\lambda_{avg} = 0.99$	Forgetting factor of online computing for average value $(0.9-1.0)$							
Initialization:	$\lambda_{var} = 0.98$	Forgetting factor of online computing for variance value (0.9–1.0)							
	$\lambda_w = 0.98$	Forgetting factor of EWRLS (0.9–1.0)							
	$K_{v} = 0.5$	Drift threshold ratio (0.1–1.0)							
	$K_{h} = 4.0$	Alarm threshold ratio (≥ 1.0)							
Output:	CUAlarm(k)	1: Alarm indication, faults are detected							

Table 5.8 The Summary of Setting for Process Fault Detection Function

	0: No fault is detected
--	-------------------------

	The Smart Function of Variable Prediction in Task13								
	$U_1(k)$:	The input 1 of the system							
Inputs:	$U_{2}(k)$:	The input 2 of the system							
	Y(k):	The predicted response of the system in Kalman predictor							
	Measurement(k):	The predicted response of the system in grey model							
	dimension of	\geq the order of the process system							
	ARX $M_{KF} = 3$								
	$d_1 = 8s / sample time$	The dead-time for input 1							
	$d_2 = 4s / sample time$	The dead-time for input 2							
Initialization:		Sample time = 1000ms (period time of the task)							
	$P_{KP} = 10$	The steps of prediction							
	$\lambda_{avg} = 0.99$	Forgetting factor of online computing for average value (0.9–1.0)							
	$\lambda_{var} = 0.98$	Forgetting factor of online computing for variance value (0.9–1.0)							

Table 5.9 The Summary of Setting for Variable Prediction Function

	$\lambda_w = 0.92$	Forgetting factor of EWRLS (0.9–1.0)
	$R = 0.3^2$	Correlation matrix of measurement noise, 0.3 is referred from the specification of NCS- TT105
	$Q = 0.2^2 I_{M_{KF}}$	Correlation matrix of process noise, 0.2 is determined by statistical methods.
	$M_{G11} = M_{G21} = n = 20$	The length of the input data sequence for $GM(1,1)$ and $GM(2,1)$
	$v_{psft} = 2$	The shifted position of the initial values for further prediction from the original data sequence $(n \ge v_{pst} \ge 1)$
Output:	$Y_{_{\!$	P_{GM} steps ahead prediction of Kalman predictor
	$\hat{x}_{GMComp}^{(0)}(P_{GM})$	P_{GM} steps ahead prediction of grey model

<u>Convergence of the parameter estimation</u>: Because of the model-based parameters estimation and the model which is used by Kalman predictor all rely on the parameter estimation, the identifiability conditions [63] should be guaranteed. Firstly, the system should be excited by the inputs sufficiently. Secondly, the two inputs signals are uncorrelated with the disturbance of the system. Especially, if the parameter estimation is used to estimate the process system in a closed-loop system, the controller order should be large than $m-d_1$ and $m-d_2$, where m is the order of the estimated parameters of the process system, and d_1 and d_2 are the discrete dead-time parameters of the system corresponding to the two respective inputs [63]. Thirdly, the expectation of the system disturbance should be equal to 0.

Normal status of the system: The normal status of the process system should work around operating point, because only the process system works around operating point is suitable to be described by the linear model. Therefore, ARX linear model can be used by parameter estimation and Kalman predictor as system model, if the process works around operating point.

5.6 Summary

This chapter implements three tasks, including fault detection, prediction, and bidirectional communication. Multitasking scheduling designs are presented and implemented following RMS rules. The three tasks are scheduled following their respective time periods and priorities, as shown in Figure 5.7. The SMART NCS-TT105 is to be verified and validated in the next chapter.



Figure 5.7: Overview of the Multitasking Implementation in Nucleus RTOS

Chapter 6

6 Verification and Validation of the Smart Sensor

To validate the designed smart functions and exam if the implemented smart sensor can provide the information of process faults and variable predicted changes, the enhanced SMART NCS-TT105 is tested in a physical bed in real-time. The implemented three tasks including bidirectional communication, fault detection, and variable prediction have been verified. The functionalities of integrated process fault detection and variable prediction are validated.

6.1 Introduction to the Test Environment

To verify and validate if the SMART NCS-TT105 can achieve the designed specifications which are discussed in section 4.1, a test facility and two DCS control systems are used to construct a test environment.

6.1.1 Introduction to the Test Facility

The typical equipment of industrial process system includes pumps, valves, pipelines, chillers, and heaters. To test the SMART NCS-TT105 in a practical process system, a system named the Nuclear Power Control Test Facility (NPCTF) [105] which can demonstrate the practical process systems, is used as evaluation process system. The NPCTF is a simplified process system designed for verification and validation (V&V) test of the research in instrumentation and control field. In this study, the primary water loop subsystem of the NPCTF is used as the evaluation process system. The noise in this evaluation process system is assumed as white noise. Utilizing this system, typical industrial application scenarios can be carried out in the research lab.

The complete schematic diagram of the primary water loop is shown in Figure 6.1. The essential equipment in the primary water loop subsystem of the NPCTF includes pumps, valves, actuators, sensors, heater and cooling system. Various combination of the equipment can emulate typical industrial process systems such as furnaces, cooling towers, steam boilers, and heat exchangers. Therefore, this primary water loop subsystem is qualified as an evaluation process system comparable to a real industrial system.



Figure 6.1: Primary Water Loop Subsystem of the NPCTF

The essential equipment in the primary water loop subsystem of the NPCTF is listed in Table 6.1. They include pumps, valves, actuators, sensors, a heater, and a cooling system. In the initial stage of the process system's evaluation, pump2 supplies water from the lower tank to the water loop through servo valve CV-16. In normal status, pump1 circulates the water loop with constant-velocity. Valves CV-11, CV-14, FV-2 and FV-3 are opened, while valves CV-3, CV-20, FV-1 are closed. The flow rate sensor F1 is used for measuring the water flow rate in the water loop. Valves CV-1 and CV-2 are regulated by proportional integral derivative (PID) controls from a Freelance DCS to control the water flow rate and ensure it stays around the operating point. The heater is controlled by signals from the Freelance DCS PID control via regulating C2. T1 and T2 are water temperature which is measured by RTD sensing element at inlet and outlet of the heater respectively. This two RTDs are connected to the NCS-TT105 smart transmitter. Valve CV-34 is regulating by the PID to control the chilled water used for cooling down the heated water via a heat exchanger. The heating and cooling systems can also be switched to manual mode for shutdown operations. Furthermore, the FV-1 manual valve can be used for operating pipeline leak fault, while the FV-2 and FV-3 manual valves can also be used for operating pipeline plugged faults.

Equipment	Components	Signals/Data	Functions	Unit
Dump	Pump1	C1	Circulate primary water flow	%
rump	Pump2	C5	Supply water in primary loop	%
	F1	F1	Flow rate of the primary water loop	L/m
Sensor	T1	T1	Water temperature at the heater inlet	°C
	T2	T2	Water temperature at the heater outlet	°C
	CV-1	CV-1	Control flow rate F1	%
Value	CV-2	CV-2	Control flow rate F1	%
valve	CV-14	CV-14	Circulate primary water flow	%
	CV-16	CV-16	Control water supply for primary loop	%
Heater	C2	C2	Control signal of the heater power	%
Heater	-	-	Mannual control for stop/start	-
Cooling	CV-34	CV-34	Control chilled water to cool heated water	%
System	-	-	Mannual control for stop/start	-
Monnucl	FV-1	-	Simulate pipeline leak	-
Valvo	FV-2	_	Simulate pipeline plug	_
valve	FV-2	-	Simulate pipeline plug	-

 Table 6.1: Essential Equipment in Primary Water Loop Subsystem of the NPCTF

6.1.2 Introduction to the DCS Systems

Besides the test facilities, ABB Freelance DCS [106] and Siemens PCS 7 DCS [107] also play important roles in this test environment. The ABB Freelance DCS and NPCTF process systems are typical of plant control systems. A Siemens PCS 7 DCS system is used to monitor and archive the online data from smart sensors.

ABB Freelance DCS is used to control and monitor the evaluation process system. The AC700F controller and the S700 I/O are the hardware of the Freelance system, while DigiVis is the operation software system (OS) of the Freelance. The DigiVis OS contains display picture graphics, faceplates operation, report, and alarm functions. The DigiVis OS graphic of NPCTF system is shown in Figure 6.2. Both the control logic program and the OS configuration program have been preconfigured and preprogrammed. This ABB Freelance DCS system runs as an established control system in practical plants.



Figure 6.2: DigiVis HMI Graphic of NPCTF system

A Siemens PCS 7 DCS system is used for connecting and monitoring the SMART NCS-TT105 via a PROFIBUS-PA Fieldbus, as well as for exchanging the data between the Freelance DCS and the SMART NCS-TT105. The S7-400 controller and its integrated PROFIBUS-DP Fieldbus port are the typical compositional configuration of the PCS 7 DCS hardware system. Through this port, the S7-400 controller can connect remote stations and PROFIBUS-PA devices through a DP/PA Link, which is the gateway for transmitting PROFIBUS-PA to the PROFIBUS-DP master system. The SMART NCS-TT105 can therefore be connected with a S7-400 controller using a PROFIBUS-PA Fieldbus. From a software angle, WinCC is used as the OS of the PCS 7 for collecting data, showing graphics and faceplates, archiving data, generating alarms, illustrating trends, and exporting reports [108]. WinCC is a high-performance archiving database system for both short-term and long-term data archiving. The archiving functions and the trend ActiveX tools are the most efficient tools for recording the experimental data and visualizing the change in values. Leveraging the PROFIBUS-DP, PROFIBUS-PA Fieldbus, and trend and data archiving, the Siemens DCS system is utilized as part of the V&V system.

The schematic diagram of the whole V&V system is shown in Figure 6.3. ABB Freelance DCS takes charge of the NPCTF system, running just as a real plant system would. The sensor and control data can be transferred through ABB S700 analog output I/O models as 4-20mA signals, connected to a PROFIBUS DP slave I/O station named DC 705F. The S7-400 controller connects this DC 705F PROFIBUS DP station and communicates with the SMART NCS-TT105 through a DP/PA Link. Through this architecture, the sensor and control data of the evaluation process system can be transferred to the SMART NCS-TT105 via a PROFIBUS PA. The data which is used for the embedded algorithms is both from external data from PROFIBUS-PA and two sensing channels of SMART NCS-TT105.



Figure 6.3: Schematic Diagram of Validation System for SMART NCS-TT105

To validate the test results, the data exported from the WinCC archiving system by the trend ActiveX tool are used for both offline and online validation tests. All the exchanged data from the PROFIBUS-PA interface of the NCS-TT105 is collected into the WinCC archiving database system. The WinCC trend ActiveX tools are used for displaying

various trends in the variables, and for exporting data as Excel files. The variable table can be used for viewing the data values in the trend ActiveX tools. Using all of these tools allows us to efficiently test the designed SMART NCS-TT105.

6.2 Verification of Communication Channels in Smart Sensors

The bidirectional communication task of the SMART NCS-TT105 is evaluated in the test environment. The systems, including SMART NCS-TT105, PCS7 DCS system, Freelance DCS system, and NPCTF primary water loop evaluation process system, all run together. The data exchanged between the SMART NCS-TT105 and the DCSs is shown in a WinCC graphic.

Among the exchanged data, the C2, F1, T1 and T2 which are listed in Table 6.2 are the four essential variables for the embedded algorithms in SMART NCS-TT105, and as such are the focused channels of the bidirectional communication for verification. Other channels among $16 \times AI$ and $8 \times AO$ channels are the same implementation methods as these four channels.

Table 6.2: Four Essential Variables for the Algorithms in SMART NCS-TT105

Name Signal		Function	Path	Scale	Unit
Heater Current C2		Control signal of the heater power	AC 700F -> DC 705F -> S7-400 -> NCS-TT105	0 - 100	%
F1 Sensor F1		Flow rate of primary water loop	AC 700F -> DC 705F -> S7-400 -> NCS-TT105	0 - 9.5	L/m
NCS-TT105 - T1	T1	Water temperature of heater inlet	Sensing elements T1 -> NCS-TT105-> S7-400	0 - 50	°C
NCS-TT105 - T2	T2	Water temperature of heater outlet	Sensing elements T2 -> NCS-TT105-> S7-400	0 - 50	°C

<u>C2: control signals for the heater current</u> C2 is the control signal used to regulate the current powering the heater, with a range of 0.0%-100.0%. It is the exogenous input signal that affects the difference in water temperature between the heater inlet and outlet. Through the DC 705F PROFIBUS-DP Slave station, C2 is acquired by a S7-400 controller, and is transferred to the SMART NCS-TT105 via a PROFIBUS-PA Fieldbus.

F1: flow rate of the primary water loop F1 is the signal of the F1 flow sensor. The range of signal F1 is 0.0 L/min - 9.5 L/min. The water flow rate is controlled by servo valves CV-1, CV-2 and manual valves CV-2 and CV-3. The water flow rate also affects the difference in water temperature between the heater inlet and outlet. The signal F1 is also therefore the exogenous input signal of the heater system. Signal F1 is connected to a current splitter in order to retransmit the measurement signal 4-20mA to both the AC700F controller and the DC 705F PROFIBUS-DP slave station, after which the signal F1 can be transferred to the SMART NCS-TT105 via a PROFIBUS-PA Fieldbus.

<u>T1: water temperature at the heater inlet</u> The range of the T1 temperature measurement is 0.0°C-50.0°C. Since water goes through the whole loop, T1 is affected by many factors, such as the cooling system, water flow rate, pipe length, and heater power. Temperature T1 can therefore be deemed as the response of an input-implicit system.

<u>T2: water temperature at the heater outlet</u> The range of the T2 temperature measurement is 0.0°C-50.0°C. It is the product of the heater, and is the essential variable of the primary water loop system. Moreover, the difference in temperature between T2 and T1 can be described as $\Delta T (\Delta T = T_2 - T_1)$. ΔT temperature is affected both by the power of the heater and flow rate . Since the control signal C2 and flow rate signal F1 are observed, ΔT can be seen as the response of an input-explicit system.

The flow chart of the exchanged data and WinCC graphic with I/O monitoring block icons are shown in Figure 6.4. In Figure 6.4, the four square I/O fields in WinCC graphic show the values of C2, F1, T1, and T2, which are exchanged with the SMART NCS-TT105. This verifies that bidirectional communication between the SMART NCS-TT105 and V&V systems has been achieved. The multiple data channels shown in Table 5.5 can be realized. Meanwhile, the collected data from the evaluation process can be used for the validation of both process fault detection and variable prediction functions.



Figure 6.4: Verification of Communication Channels of SMART NCS-TT105

6.3 Offline Validation of Smart Functions in MATLAB

The previous verification section paves the way for acquiring proper data from the evaluation process system. To guarantee the functionalities and specifications of the designed smart functions are qualified before download them into the NCS-TT105, the algorithms are programmed in the MATLAB and are validated using acquired process data. This so-called offline validation means that the algorithms are in MATLAB rather than running in the NCS-TT105, as shown in Figure 6.5. The data archived and exported from WinCC is imported into MATLAB as the data source for algorithm validation. The data from C2, F1, T1, and T2 are imported as a ".txt" file, with a sample time of the data of 1000ms. The cycle time of the algorithms in MATLAB for each step is 1000ms. Other parameters and initial values are set directly in MATLAB. The graphics of all the test

results from MATLAB can intuitively display the fault detection alarms and the effects of the prediction. The procedure and results of validation for the two functions are shown in the next two subsections.



Figure 6.5: Diagram of Smart Functions Offline Validation

6.3.1 Process Fault Detection

The fault detection algorithms, including EWRLS for parameter estimation and residual generation, and CUSUM for residual evaluation, are programmed in MATLAB. To validate the functionalities of the algorithms, typical faults which frequently occur in practice are considered. These faults, including heater power tripping, abnormal heater operation, chiller shut down, pipeline plug, pipeline leak, flow rate signal loss, and T1 sensing line break, are demonstrated in this primary water loop process system. Those faults and their time stamps are listed in Table 6.3.

Faults	Time	Horizontal Axis	Events	Operation	Objects
	2:30 PM	0	Normal operation		
Α	2:45 PM	908	Heater power tripping	Heater is stopped (C2 is supplied continously)	Heater
В	3:00 PM	1816	Control malfunctioned	C2 is manipulated 69.3% -> 50%	Heater
С	3:15 PM	2705	Chiller shut down	Cooling system is shut down	Cooling system
D	3:31 PM	3660	Pipeline plugged	FV-2 and FV-3 are closed and opened again	Pipeline
Е	3:40 PM	4186	F1 signal loss	Sensor F1 is powered off	F1 Sensor
F	3:50 PM	4800	T1 signal loss	T1 sensing elements is unplugged	NCS-TT105
G	4:00 PM	5440	Pipeline leak	FV-1 is opened and closed again	Pipeline

Table 6.3: Simulated Faults in Primary Water Loop System for Offline Validation

The initial status of the water loop system is running around its normal operating point. At the stable operation point, water temperature T1 at inlet of heater is cooled around 22°C, and water temperature T2 at outlet of heater is controlled by the heater around 28°C. The heater current control signal C2 is regulated around 68% and flow rate F1 is around 8.8 L/min. The archived data is selected from 02:30 PM to 04:05 PM, and the data length of every variable is 5700, which means the duration time of the test is 5700s. The whole process of the variables changes is shown in Figure 6.6.



Overview of Process in Fault Detection Event

Figure 6.6: Overview of Variables in MATLAB for Fault Detection Validation

Fault A: This fault simulates the tripping occurrence in the current power cable of the heater. To demonstrate this fault, the heater is stopped and the control signal C2 is sent to the SMART NCS-TT105 continuously. Figure 6.6 shows that measurement T2 and T1 decrease because of the heater power loss, and are cooled by the chiller in the water loop after tripping fault occurs. Since C2 is held and the water flow rate is not influenced, the inputs signals of EWRLS are unchanged. After a moment, the power of heater is switched back to auto mode, and T1 and T2 rise back to their normal operation points.

Fault B: This fault simulates abnormal operation of the heater current control, which can be caused by the operator misusing the controller. To demonstrate this fault, the control mode of the heater is switched from auto to manual mode, and the C2 is changed from 69.3% to 50% to reduce the current supply to the heater. T1 and T2 drop after this fault occurs and rise again after the heater recovers into auto-control mode.

Fault C: This fault simulates cooling system shut down. To demonstrate this fault, the chiller is manually stopped, resulting in T2 increasing 3°C before controllers return it back to its operating point after 2705 of the horizontal axis in Figure 6.6. The changes of T1, T2, and C2 can be seen in Figure 6.6. The chiller is also promptly restarted.

Fault D: This fault simulates a pipeline plug caused by impurities or dirt. To demonstrate this fault, the manual valve FV-2 and FV-3 are turned down and reopened within 3 s. Since the water pipe is suddenly blocked, the flow rate decrease from 8.8 to 4.2 L/min but recovers instantaneously. Since the water flow fluctuates, small changes in T1 and T2 also influenced.

Fault E: This fault simulates a scenario in which the sensor which communicates with SMART NCS-TT105 loses power or loses sensing signals. The F1 flow rate sensor is powered off to demonstrate this fault. The F1 signal changes caused by this fault can be seen from 4186 of the horizontal axis in Figure 6.6.

Fault F: This fault simulates a sensing element loss or sensing line break of the SMART NCS-TT105. To demonstrate this fault, the T1 RTD sensing line is plugged out. The measurement of T1 jumps to 0 at 4800 of the horizontal axis in Figure 6.6. After 5s, the T1 sensing line has recovered. Moreover, since T1 is not involved in closed-loop control, neither the heater nor the T2 outlet temperature are not affected.

Fault G: This fault simulates a water leak caused by pipeline rupture. To demonstrate this fault, the manual valve FV-1 is opened and reclosed within 5s. Since the water run out, the flow rate F1 signal decreases dramatically which can be seen around 5440 of the horizontal axis in Figure 6.6. Since the water loss is in closed water loop, the water temperature arises obviously after this fault occurs.

To show the internal effects of CUSUM, the fault residual generation and residual evaluation are shown in Figure 6.7. In normal status, the changes in residuals are within about ± 0.02 °C, which means that the estimated parameters can track the actual parameters of the system well. Once faults occur in the process, the residual changes dramatically, which can clearly be seen in Fault A to G occurrence in the top plot of Figure 6.7. In the middle plot of Figure 6.7, the evaluation results $g_k^{(1)}$ and $g_k^{(2)}$ from (4.18) are red and green respectively. The dynamic threshold *h* in (4.19) is the blue line. It can be seen that after the faults occur, once either the $g_k^{(1)}$ or $g_k^{(2)}$ grows greater than *h*, it can be seen that the fault alarm indications are generated promptly. Finally, fault detection alarms with value 1 are indicated in the bottom plot of Figure 6.7. The black lines which refers to faults detection are located in 910, 1820, 2708, 3668, 4190, 4801, and 5455 of the horizontal axis.

Furthermore, since the mean of the residual is almost zero and the variance of the residual is still small value in normal status, therefore if the residual is assumed to be equal to the noise of the system, the assumption of the white noise in the test system can be accepted.



Figure 6.7: Residual Generation and Evaluation for Fault Alarm

Revisiting the faults occurrence record in Table 6.3, the response interval between each fault and its corresponding fault indication are listed in Table 6.4. It can be seen that process faults can be detected within 25 steps. If every step corresponds with 400ms, the response time of fault alerts are all within 10s.

This offline validation shows that fault detection function can detect process faults and generate alerts signals in time, enabling the system to achieve the specifications of fault detection functions can be achieved. The fault detection algorithms can be embedded into the SMART NCS-TT105 for online process fault detection tests.

Faults	Events	Time	Fault Occur	Fault Detection	Response Steps	Response Time(s)	Sepcifi- cation(s)	Validation Result
Α	Heater power tripping	2:45 PM	908	910	2	< 1	< 10	Pass
В	Control abnormal	3:00 PM	1816	1820	4	< 2	< 10	Pass
С	Chiller shut down	3:15 PM	2705	2708	3	< 2	< 10	Pass
D	Pipeline plugged	3:31 PM	3660	3668	8	< 4	< 10	Pass
Е	F1 signal loss	3:40 PM	4186	4190	4	< 2	< 10	Pass
F	T1 signal loss	3:50 PM	4800	4801	1	< 1	< 10	Pass
G	Pipeline leak	4:00 PM	5440	5455	15	< 8	< 10	Pass

Table 6.4: Result of Offline Validation of Fault Detection Test

6.3.2 Variable Prediction

The variable prediction algorithms including GM(1,1), GM(2,1) grey model prediction and Kalman predictor as well as multi-step iteration, are all programmed in MATLAB. To validate the functionalities of the code, the 10-step ahead predicted values of heater outlet water temperature T2 are compared with actual variable. Revisiting the specifications of variable prediction in section 4.1.2, prediction accuracy has two criteria, which are around the normal operating point and abnormal status. Therefore, as shown in Table 6.5, the variable prediction validation is tested under these two scenarios.

Table 6.5: Scenarios of Offline Variable Prediction Test

Scenarios	Status	Start Time	End Time	Start Horizontal Axis	End Horizontal Axis	Events
А	System is around operating point	6:40:01 PM	6:41:00 PM	2401	2460	C2 is regulated
В	System is in abnormal status	6:41:01 PM	6:44:00 PM	2461	2640	Heater is shut down and started again

In the initial stage of the test, the process system is running normally. At the stable operation point, temperature T1 is cooled around 17°C, and temperature T2 is controlled by the heater is around 26°C, while the control signals C2 is regulated around 65% and the flow rate F1 is around 6.4 L/min. To test the prediction results, the archived process values which include both process changes around the normal operating point and changes related to abnormal status are selected between 6:40 PM to 6:44 PM,

corresponding to the horizontal axis 2400 to 2640. The process changes associated with Scenarios A and B are shown in Figure 6.8.

Under Scenario A, the current power of the heater control signal C2 is regulated, result in the T2 decrease from 28°C to 25°C. Since the reduce of T2 temperature, the cooling system decreases the speed of its internal pump. This leads the water flow rate has a little decrease, and F1 flow rate signal changes from 6.4 L/m to 6.2 L/m. Since every equipment in system is normal and the T2 fluctuate around the operating point 26 °C, therefore, the Scenario A is the system works under normal status.

Under Scenario B, the heater is shutdown to demonsrate the system works under abnormal status. To show the influence by the heater only, the flow rate is controlled to be kept stable. Since the heater is shutdown, both the temperature value of T1 and T2 drop dramatically. After about 80s, the heater is started again, and T2 increase gradually following the C2 is increased by the controller.

Under these two scenario, the T2 variable is predicted by the variable prediction algorithms. To predict T2, T1 and $\Delta T (\Delta T = T_2 - T_1)$ are predicted respectively, and the final prediction of T2 is combined from these two parts.



Figure 6.8: Overview of the Process Changes for Prediction Validation

The grey model prediction function is used for predicting heater inlet temperature T1. Figure 6.9 shows the prediction effects of the heater inlet water temperature T1 from three grey model prediction methods which are prediction from GM(1,1), GM(2,1), and combined method with W_{GM} . The actual T1 variable is represented as a dark green line; the 10-step prediction of GM(1,1) is a red line; the 10-step prediction of GM(2,1) is blue line; and the final combined prediction is black line. Since the black line is ahead of the green line, this indicates that the combined grey model prediction accurately foresees the process variables.

To validate the effects of W_{GM} , the percentage value of the W_{GM} is shown as bold red line in the bottom of Figure 6.9. From 2480 to 2530 of the horizontal axis, the value of T1 changes from a flat line to a decline. Because of this non-monotonic change, W_{GM} is greater than 0 from 2497 to 2530 of the horizontal axis. As discussed in equation (4.66), if W_{GM} is greater than 0, the combined prediction is balanced between the prediction from GM(1,1) and GM(2,1). From the zoomed graphic, it can be seen that the black line located between the red line and blue line when W_{GM} is greater than 0. To validate that the combined method's prediction is better than the prediction from GM(1,1) and GM(2,1), the average error rate of the prediction is used for validation. Using equation (6.1) and selecting $2507 \le k \le 2540$, the average error rate of the 10-step prediction from GM(1,1), GM(2,1) are calculated by the equation (6.1), and the combined method are compared in Figure 6.9

$$Prediction \ error \ rate = Avg[Abs(\frac{Prediction(k-10) - Actual(k)}{Actual(k)})] \times 100\%$$
(6.1)

where the prediction steps is 10, Prediction(k-10) is the *p*-step ahead prediction at (k-10) th step. Actual(k) is the actual process response at kth step. It should be noted that the Actual(k) must not equal to 0 when using equation (6.1).

As shown in Table 6.6, the average error rate of the prediction from combined method is less than the prediction from GM(1,1) and GM(2,1). This means that the proposed combined methods can provide better prediction than the independently using GM(1,1) and GM(2,1).



Figure 6.9: Offline T1 Prediction by the Grey Model Prediction

Curve	Prediction Method	Prediction Start	Prediction End	Actual Value Start	Actual Value End	Prediction error rate
Red	GM(1,1)	2497	2530	2507	2540	0.86%
Blue	GM(2,1)	2497	2530	2507	2540	0.87%
Black	Weighted from GM(1,1) and GM(2,1)	2497	2530	2507	2540	0.73%

Table 6.6: Comparison of Prediction by Grey Model and Combined Method

Kalman one-step prediction and multi-step iteration prediction are used for predicting the $\Delta T (\Delta T = T_2 - T_1)$. In contrast to prediction using the Grey model, the impact of inputs have been considered in the Kalman one-step predictor and multi-step iteration prediction. Since ΔT is affected by both heater power and flow rate, C2 and F1 are considered as inputs of the ARX model in the Kalman predictor and multi-step iteration for ΔT prediction.

The prediction effects of the Kalman predictor are shown in Figure 6.10. In Figure 6.10, the brown line is the 10-step ahead prediction; the orange line is actual ΔT ; the dark blue line is the value of C2; and the light blue line is the value of F1. The three stages including heater working around operating point, heater shutdown, and heater restart are shown. It can be seen that the brown line is ahead of the orange line, indicating that the ΔT prediction foresees the ΔT variables. It needs to be noted that the ΔT prediction effects between 2480 and 2560 of the horizontal axis are not good, because the C2 is 0, the water temperature in this interval is far from the normal operating point, and the online estimated parameters of ARX have not tracked the actual parameters of the system in this interval. After 2560 on the horizontal axis, the C2 is increased and the exciting input is increased. After that, the parameters can be estimated well, and the prediction shows normally.


Figure 6.10: Offline (T2-T1) Prediction by Kalman Predictor and Multi-step Iteration

The effects of combined T2 prediction are shown in Figure 6.11. The actual value of T2 is shown as a black line and the 10-step prediction value is shown as a red line. Among the intervals in scenario A, the prediction value of T2 at 2410 of the horizontal axis is 25.70°C. The actual value of T2 after 10s is 25.48°C, and the prediction error is 0.22°C. Among the intervals in scenario B, the prediction value of T2 at 2475 of the axis which is in the right hand zoomed figure is 22.89°C. The actual value of T2 after 10s is 23.01°C. The prediction error is 0.12°C. The actual value of T2 after 10s is 23.01°C, while the prediction error is 0.12°C. These two examples show the prediction errors are small in both scenario A and scenario B of system working conditions. To display the comparison between the prediction value and actual value of T2 more intuitively, the prediction value

 ΔT (T2-T1) Prediction by the Kalman Predictor and Multi-step Iteration

of T2 is shifted by 10 steps to the right and shown in Figure 6.12. This plot presents an overview of the effects of the T2 prediction. The red line of prediction and black line of actual T2 is basically matched in the duration of test. The absolute errors between the prediction and actual value on every sampled point between 2401 and 2640 are less than 1.0 °C, although ARX did not track the actual parameters of the system very well between 2480 and 2560 of the horizontal axis, which is discussed about ΔT prediction in previous paragraph.



Figure 6.11: Offline T2 Prediction Validation for Combined Prediction



Figure 6.12: Offline T2 Prediction Validation for Combined Shifted Prediction

Using equation (6.1), and selecting $2390 \le k \le 2450$, the predicted value of T2 from 2390 to 2450, and the actual value of T2 from 2400 to 2460 of the horizontal axis are compared. Finally, the validation results of the average errors of the T2 10-step prediction in scenario A and B are shown in Table 6.7. The prediction error rate in scenario A is 0.92%, while the prediction error rate in scenario B is 2.18%. Both prediction error rates are less than the specification. The grey model and Kalman predictor can both be implemented into the NCS-TT105 for online prediction tests.

Table 6.7: Validation Results of Offline Prediction Test

Scenarios	Status	Start Time	End Time	Start Horizontal Axis	End Horizontal Axis	Length of Sampled Data	Prediction Error Rate	Specification	Validation Results
А	C2 is regulated System is around operating point	6:40:01 PM	6:41:00 PM	2401	2460	60	0.92%	< 1.2% = 0.3/25°C (Around O.P.)	Pass
В	Heater is shut down and started again System is in abnormal status	6:41:01 PM	6:44:00 PM	2461	2640	180	2.18%	< 6% = 1.5/25°C (Abnormal)	Pass

Furthermore, because of the EWRLS parameter estimation for identifying the parameters of ARX models which are used both in fault detection and prediction, the condition of the EWRLS parameter estimation using in the closed loop should be investigated. As discussed in section 5.5 about the convergence of the parameter estimation, since the orders of PID closed loop controller is 2, the delay parameters $d_1 = 8s/T_s$, $d_2 = 4s/T_s$, and the orders of the ARX m = 5, these meet the identifiability conditions [63] which are the controller order large than $m-d_1$ and $m-d_2$. The convergence criteria of parameter estimation using in this test closed loop system can be achieved. Therefore, the EWRLS parameter estimation can be used to identify the ARX model in this closed loop systems. The EWRLS parameter estimation can be used in smart sensor for on-line identifying the primary water loop system.

6.4 Online Validation

Based on the successful offline validation of algorithms in MATLAB, the functionalities of the implemented algorithms can be guaranteed. To implement the process fault detection and variable prediction functions into NCS-TT105, two groups of algorithms are programmed as embedded C code and organized as embedded tasks in the RTOS of the SMART NCS-TT105. This so-called online validation means that the algorithms are scheduled by Nucleus RTOS and embedded in SMART NCS-TT105, as shown in Figure 6.13.

To validate the internal algorithms of the SMART NCS-TT105, all the data exchange in the PROFIBUS-PA interface are archived and monitored in the trend ActiveX in WinCC OS. Among the exchanged data, four channels of data are the inputs of the embedded algorithms. Heater control signal C2 and flow rate F1 are exogenous inputs of the ARX models in the algorithms. Water temperature T1 and temperature T2 are the known responses of the process system. Regarding the output of the internal functions, alarm indication, which is from 0 and 1, is produced by a fault detection algorithm. The 10s prediction of T2 is provided by an embedded variable prediction algorithm. The sample time of WinCC for data archiving and trend drawing is 1000ms. Furthermore, since three tasks which are bidirectional communication Task11, process fault detection Task12, and variable prediction Task13 are scheduled in the Nucleus RTOS of SMART NCS-TT105, to test the multitasking scheduling of these three tasks, the execution time and period time of the tasks are needed to be validated.



Figure 6.13: Diagram of Smart Functions Online Validation

6.4.1 Multitasking in RTOS

In this section, the execution time and period time of three tasks in Nuclues RTOS are tested. To test the execution time of a task, the service NU_Retrieve_Clock() is used at both the beginning and the end of a task. The result of NU_Retrieve_Clock() is the kernel time. Execution time is computed using a counter to calculate kernel time. The method is shown in Figure 6.14. The counter result "ulExecutionTime" can be the execution time of the task if the kernel tick is set as 1ms.

```
TASK()
{
    ulST1 = NU_Retrieve_Clock(); // The timer at start of the task
    ...
    Algorithms
    ...
    ulET1 = NU_Retrieve_Clock(); // The timer at end of the task
    ulExecutionTime = ulET1 - ulST1; //Calculating the used timer
}
```

Figure 6.14: The Method of Computing Execution Time of a Task

Counters are used in all three trasks to test their time period. The time period is calculated by equation (6.2), while online test method is shown in Figure 6.15. Combining the test duration time and the counter results of each task, the period time of each task can be calculated.

Tested
$$T_P = \frac{Test \ duration}{Counter}$$
 (6.2)



Debugging NCS-TT105 in programming software MULTI

Figure 6.15: The Method of Computing Period Time of Three Tasks

The validation results of execution time and period time are shown in Table 6.8 and Table 6.9 respectively. The execution time of the Task11, Task12, and Task13 are 27ms,

70ms, and 80ms respectively. To avoid counter error, the period time is tested for six times. The averaging period time of the Task11, Task12 and Task13 are 200ms, 400ms, and 1000ms respectively. The processor workload is 39.5% which is analyzed in (6.3). Since the workload sum of the three tasks is less than 70%, the integrated tasks are suitable for running in NCS-TT105.

$$WL_{processor} = \frac{27}{200} + \frac{70}{400} + \frac{85}{1000} = 39.5\%$$
(6.3)

From the test results, we can conclude that the execution time and period time of three task achieve the application requirement, and the designed parameters of Nucleus RTOS are correct. The real-time multitasking design and the period time of three tasks achieve the required specifications.

Task Number	Tasks Functions	Designed Execution Time	Tested Execution Time	Validation Result
Task 11	PA bidirectional communication	<30ms	27ms	Pass
Task 12	Process fault detection	<100ms	70ms	Pass
Task 13	Variable prediction	<150ms	85ms	Pass

Table 6.8: Validation Results of Execution Time of Three Tasks

Table 6.9: Validation Results of Period Time of Three Tasks

Multitasking Validation	Group1 Execution Time Validation Test Duration 40s			Group1 Execution	Group2 Ex Te	ecution Time est Duration S	Group2 Execution	Validation Begults	
Test	Test1 (Counter)	Test2 (Counter)	Test3 (Counter)	Time	Test4 (Counter)	Test5 (Counter)	Test6 (Counter)	Time	Results
Task 11	200	200	200	200ms	400	400	400	200ms	Pass
Task 12	100	100	100	400ms	200	200	200	400ms	Pass
Task 13	40	40	40	1000ms	80	80	80	1000ms	Pass

6.4.2 Process Fault Detection

In online validation, the simulated faults emulating the faults occurring in real systems are demonstrated in the evaluation process system. These faults include heater power tripping, chiller shut down, pipeline plug and leak, and sensor signal loss. The architecture of process fault detection for online test is shown in Figure 6.16. The exchanged data and alarm indication of the SMART NCS-TT105 is shown in the trend ActiveX tool of WinCC.



Figure 6.16: Validation of Process Fault Detection in SMART NCS-TT105

The overview of the fault detection graphic is shown in Figure 6.17. The light blue line is the trend of value F1, the dark blue line is the trend of value C2, the green line is the trend of value T1, and the red line is the trend of value T2. The black line, which represents the alarm alert, is changed from 0 to 1 when the NCS-TT105 generates an alarm indication. According to the specification designed in section 4.1, the process fault detection validation test can pass if the time interval between process fault occurrence and black alert is less than 10s.

10/03/17 14:19:15.819 0 SSAI/Fax	ultAlarm Fau	Its Detected in Sensor Loo	p	CG X	3/10/2017 2:20:02 PM
FaultDetection	Prediction		Process	NPTCF_SYS	Western Engineering
T1-Health Milet T2-Health Outlet T2-T1 19.75 23.18 3.39 EVIR.3_input EVIR.3_input EVIR.3_input 0.205 -0.02 4.62	EWRLS_Residuals	CP → Etternal 70.75 b 70.75 b	T1/T2 Fault Injection	Picture Tree	Fault Alarm with message & time stamp
CUSUM_G1 CUSUM_G2 CUSUM_Threshol	5 Fault Detection Alarm	4.00 unknown		Trend window	
	i 🔛 🖽 🛈 🔛 💥 🔳 🖨 🕷	1 4 63 M			
T1,T2 F1 C2 Alam CUSUM 30 75 0.060			*****		******************
28 5 70 1.6 0.055					D 1 1 70
26 4 0.005 3 65 1.2 0.045 24 60 0.040 0.033 22 255 0.8 0.030	Light blue - (0-9.5)	F1	Dark blu (0-100)	e - C2 Fault alarm 0->1	Dark red - 12
20 1 50 0.6 0.025 18 0 45 0.4 0.020 16 1 40 0.2 0.010 14 - 2 7 0 0.005		Dark	green - T	1	
Ready Value Axis	2:17:20 PM 2:17:30 PM 2:17:40 PM 3/10/2017 3/10/2017 3/10/2017	2:17:50 PM 2:18:00 PM 2:18:1 3/10/2017 3/10/2017 3/10/2	0 PM 2:18:20 PM 2:18: 017 3/10/2017 3/10	20 PM 2:18:40 PM 2:18:50 PM 2:19:00 PM 2:19:10 PM 2:19:20 2017 31(0)2017 31	0 PM 2:19:20 PM 2:19:40 PM 2:19:50 PM 2:20:00 017 3/10/2017 3/10/2017 3/10/2017 3/10/20 ₩
	× 1 t		⇒ [

Figure 6.17: Validation Picture of Fault Detection Graphic in WinCC

Before the online process fault detection test, the primary water loop system of evaluation process system needs to run around its normal operating point. Under normal operation, water temperature T1 is cooled to around 20°C, water temperature T2 is heated to around 30°C, while the control signal C2 is regulated around 70%, and the flow rate F1 is around 6.5 L/min. The online test is from 02:00 PM to 17:30 PM. During this online test, the demonstrated six types of faults are shown in Table 6.10.

Table 6.10: Simulated Faults in Primary Water Loop System for Online Validation

Faults	Events	Fault Occurs	Operation	Object
Α	Heater tripping	2:19:12 PM	Heater is stoped (C2 is supplied continously)	Heater
В	Chiller shut down	2:37:40 PM	Cooling system is shut down	Cooling system
С	Pipeline plugged	3:07:29 PM	FV-2 and FV-3 are closed and opened again	Pipeline
D	F1 signal loss	4:11:06 PM	Sensor F1 is powered off	Sensor F1
Е	T1 signal loss	4:15:34 PM	T1 sensing element is plugged out	NCS-TT105
F	Pipeline leak	5:19:48 PM	FV-1 is opened and closed again	Pipeline

Fault A: This fault simulates the tripping occurrence in the heater's power cable. To demonstrate this fault, the heater is stopped manually, and the control signal C2 is sent to SMART NCS-TT105 continuously. In Figure 6.18, it shows that the fault occurs at 02:19:12 PM. After the heater losses power, water temperatures T2 and T1 decrease due to water being cooled by the cooling system. It can be seen that fault alert shown in position@ is archived at 02:19:16 PM. The response time of fault detection for the fault of heater tripping is 4s, which is less than 10s.



Figure 6.18: Fault Detection Validation of Heater Tripping Fault

Fault B: This fault simulates cooling system shut down, demonstrated by manually stopping the chiller. In Figure 6.19, it shows that the cooling system is shut down at 02:37:40PM. After this fault occurs, water temperature T1 and T2 both increase. Because of the loop control, control signal C2 is also decreased to reduce the heater power. The fault alert shown in the position@ is archived at 02:37:46PM. The response time of fault detection for the fault of cooling system shut down is 6s, which is less than 10s. Furthermore, T1 and T2 increases dramatically even though the loop controller takes actions for regulating C2. After 2:37:50 PM, The system approaches the deteriorated status and C2 is regulated distinctly, therefore, the fault is detected in second twice because of this serious faults occur.



Figure 6.19: Fault Detection Validation of Cooling System Fault

Fault C: This fault simulates pipe plug by either impurities or dirt. To demonstrate this fault, the manual valve FV-2 and FV-3 are turned down and reopened within 10s. As shown in Figure 6.20, after the pipeline is plugged at 03:07:29 PM, the flow rate disturbances range from 6.3 to 4.4 L/min. The water temperature of T1 and T2 also undergo small changes due to this fault. It can be seen that the fault alert shown in the position@ is achieved at 03:07:36 PM. The response time of the fault detection for the fault of pipeline plug is 7s, which is less than 10s.



Figure 6.20: Fault Detection Validation of Pipeline Plug Fault

Fault D: This fault simulates a scenario involving either a power loss to the sensor which communicates with the SMART NCS-TT105 or a loss of sensing signals. To demonstrate this fault, the F1 flow rate sensor is powered off. In Figure 6.21, the signal F1 is 0 after the F1 sensor was powered off at 04:11:06 PM. The alarm alert shown in the position@ is archived at 04:11:11 PM. The response time of fault detection for the fault for F1 signal loss is 5s, which is less than 10s. Furthermore, because of the F1 changes dramatically, the SMART NCS-TT105 indicates consecutive fault alarm with 3 times, allowing the black line to keep in the state of value 1 for 3s.



Figure 6.21: Fault Detection Validation of F1 Loss Fault

Fault E: This fault simulates a sensing element loss or break. To demonstrate this fault, the T1 RTD sensing line is plugged out. In Figure 6.22, it shows that the value of T1 jumps from 20.18°C to 0°C at 04:15:34 PM. Because of the T1 value is not involved in closed-loop control, control signal C2 and temperature T2 are not affected. The alarm alert shown in the position@ is achieved at 04:15:36 PM. The response time of fault detection for T1 signal loss is 2s, which is less than 10s. Furthermore, because of the T1 changes dramatically, the SMART NCS-TT105 indicates consecutive fault alarm with 2 times, allowing the black line to keep in the state of value 1 for 2s.



Figure 6.22: Fault Detection Validation of T1 Loss Fault

Fault F: This fault simulates the pipeline leak caused by pipeline rupture. To demonstrate this fault, the manual valve FV-1 is opened and reclosed within 10s. Figure 6.23 shows the fault occurring at 05:19:48 PM, at which point the flow rate changes from 6.87 to 6.65 L/min and then decreases dramatically due to the manipulation of manual valve FV-1. The alarm alert shown in the position@ is archived at 05:19:52 PM. The response time of process fault detection for the fault of pipeline leak is 4s, which is less than 10s.



Figure 6.23: Fault Detection Validation of Pipeline Leak Fault

<u>**CUSUM Residual Evaluation:**</u> The internal effects of two-sided CUSUM residual evaluation is shown in Figure 6.24. The yellow line shows the changes in residuals generated from EWRLS. The evaluation results $g_k^{(1)}$ and $g_k^{(2)}$ from equation (4.18) are red and green trends, respectively. Threshold *h* in equation (4.19) is the blue line. Since *h* is changed dynamically, the blue line fluctuates rather remaining constant. Under normal operation, the red and green lines are under the blue line. Once the fault occurs, the black line representing the alarm indication changes from 0 to 1 if either the $g_k^{(1)}$ or $g_k^{(2)}$ is greater than *h*. Figure 6.24 shows the alarm indication is 1, because the green line $g_k^{(2)}$ is 0.0341, which is higher than the blue line threshold 0.0248 at 2:37:46PM. This means that these faults can be effectively evaluated using the two-sided CUSUM algorithm.



Figure 6.24: Effects of Alarm Indication by CUSUM

As shown above, the faults including heater tripping, chiller shut down, pipeline plug and leak, F1 signal loss, and T1 signal loss are all demonstrated for fault detection validation. The validation results are summarized in Table 6.11. Those online validation tests show that the embedded fault detection algorithms can detect the process faults and generate

fault indications in time. The response times of fault alerts are all within 10s. Thus, the designed fault detection function in SMART NCS-TT105 achieves the desired specifications, and the process faults can be detected in incipient period.

Faults	Events	Fault Occurs	Fault is Detected	Response time (s)	Specification (s)	Validation Result
Α	Heater tripping	2:19:12 PM	2:19:16 PM	4	< 10	Pass
В	Chiller shut down	2:37:40 PM	2:37:46 PM	6	< 10	Pass
С	Pipeline plugged	3:07:29 PM	3:07:36 PM	7	< 10	Pass
D	F1 signal loss	4:11:06 PM	4:11:11 PM	5	< 10	Pass
Е	T1 signal loss	4:15:34 PM	4:15:36 PM	2	< 10	Pass
F	Pipeline leak	5:19:48 PM	5:19:52 PM	4	< 10	Pass

Table 6.11: Online Validation Results of Fault Detection Test

6.4.3 Variable Prediction

To validate the integrated variable prediction functions with an online test, the actual T2 and the predicted T2 are compared, and the average error between the predicted and actual T2 is validated. Since the periodic time of task13 is 1s, the 10-step ahead prediction represents a 10s prediction.

To predict the T2 value, the prediction of T1 computed using the grey model, and prediction $\Delta T (\Delta T = T_2 - T_1)$ computed using the Kalman predictor and repeated iteration are combined. The organization of two parts of prediction and corresponding inputs and outputs are shown in Figure 6.25. The combined prediction of T2 values and the actual T2 value are compared to validate prediction accuracy. The trend and variable table are used as tools for examining the prediction results in the WinCC. Equation (6.1) is used for computing error as prediction error rate validation approach.



Figure 6.25: Validation of Variable Prediction in SMART NCS-TT105

The overview of the prediction graphic is shown in Figure 6.26. The dark blue line is the trend of the C2 value, and pink line is the trend of F1 value. The dark green line is the trend of the T1 value, while the light green line is the trend of the 10s prediction of the T1 value. The dark red line is the trend of the T2 value, whereas the light red line is the trend of 10s prediction of the T2 value. To better demonstrate the prediction results, the variables table is used and shown at the bottom of every figure. According to the specifications designed in section 4.1, the variable prediction validation test can pass if the prediction error rate is less than 1.2% around a normal operating point, and 6% under an abnormal system status.



Figure 6.26: Validation Picture of Prediction Graphic in WinCC

Before the online variable prediction test, the primary water loop system in the evaluation process system runs around the normal operating point. Under the normal operation of the water loop system, water temperature T1 is cooled around 17°C, water temperature T2 is controlled by a heater around 26°C, while the heater control signal C2 is regulated around 65%, and flow rate F1 is around 6.5 L/min. The duration time of this online test is from 05:00 PM to 07:30 PM. The prediction performance in four scenarios of process changes are listed in Table 6.12.

Table 6.12: Scenarios of Online Variable Prediction Test

Scenarios	Operation	Start Time	End Time	Status
Α	F1 changes (Flow changes)	5:13:05 PM	5:14:05 PM	System works around operating point
В	C2 changes (Power of heater changes)	6:07:45 PM	6:08:45 PM	System works around operating point
С	Heater shutdown	6:45:00 PM	6:48:00 PM	System is in abnormal status
D	Chiller shutdown	7:10:10 PM	7:13:10 PM	System is in abnormal status

Scenario A: This scenario demonstrates the system changes around the operating point due to water flow rate changes. Figure 6.27 shows that the all the variable trends changes after the flow rate F1 changes from 7.5 to 6.5 L/min at 05:13:10 PM. Using the variable table, variables at 05:13:30 PM and 05:13:30 PM are shown in detail. The T2 prediction at 05:13:30 PM is 25.31°C, and the actual value of T2 after 10s is 25.46°C. The error between the prediction and actual value is 0.15. To validate the prediction error rate under this scenario, the actual values of T2 from 05:13:05 PM to 05:14:05 PM, which is the main parts of the T2 value fluctuation are selected and compared with the corresponding prediction results. The average error rate of prediction calculated by the equation (6.1) is 0.43%. This average error rate is also shown in a validation summary in Table 6.13.

In Figure 6.27, it also shows that the response of T2 actual value can be described approximately as a second orders underdamped systems corresponding with the water flow rate changes. Therefore the *m* dimension of the ARX model which is set 3 for variable prediction function and 5 for process fault detection function can be used to describe this primary water loop system which works around 25° C and 26° C.



Figure 6.27: T2 Prediction Validation with the Changes of Flow Rate F1

Scenario B: This scenario demonstrates the system changes around the operating point due to power of heater changes. In Figure 6.28, it shows that the all variables trends changing after control signal C2 changes from 65% to 85% and fluctuates after 06:07:46 PM. Using the variable table and selecting the variables at 06:07:52 PM and 06:08:00 PM are shown in the details. It can be seen that the prediction of T2 at 06:07:52 PM is 26.60°C, and the actual value of T2 after 10s is 26.73°C. The error between the prediction and actual value of this example point is 0.13. To validate the prediction error rate under this scenario, the actual values of T2 from 06:07:45 PM to 06:08:45 PM, which form the main parts of the T2 value fluctuation, are selected and compared with the corresponding prediction results. The average error rate of prediction calculated by the equation (6.1) is 0.41%. This average error rate is also shown in the validation summary in Table 6.13.



Figure 6.28: T2 Prediction Validation with the Changes of Heater Control Signal C2

Scenario C: This scenario demonstrates the system changes when the fault being considered heat shutdown. In Figure 6.29, it shows that the all the variables trends change due to this abnormality, which occurs after 06:45:00 PM. Differing from the system changes around the operating point, the heater shutdown causes obvious changes in temperatures T1 and T2, and results in the T2 value moving far away from its normal operating point of 26 °C. The light red line, which is the prediction of T2, undergoes abrupt changes after the heater shuts down near 06:45:10 PM, because the model of the system has an abrupt change in abnormal status. Using the variable table, the variables at 06:45:16 PM and 06:45:26 PM are shown in detail. It can be seen that the T2 prediction at 06:45:16 PM is 19.20°C, and the T2 actual value after 10s is 19.98°C. The error between the prediction and actual value is 0.78°C. To validate the prediction error rate in this scenario, the actual values of T2 from 06:45:00 PM to 06:48:00 PM, which are the main parts of the T2 value fluctuation are selected and compared with the corresponding prediction results. The average error rate of prediction calculated by the equation (6.1) is 3.64%. This average error rate is also shown in the validation summary in Table 6.13.



Figure 6.29: T2 Prediction Validation with Heater Shutdown

Scenario D: This scenario demonstrates the system changes when the fault under consideration is a cooling system shut down. In Figure 6.30, it shows that the all the trends in variables changes due to this abnormal situation after 07:10:10 PM. This abnormality causes temperatures T1 and T2 to increase from 07:10:15 PM onwards. The heater power control signal C2, which is controlled by a loop controller, is decreased. Since the forgetting factor of the EWRLS is set at 0.98, which is suitable for tracking the slowly changing parameters of the system model, these abrupt changes in system parameters are not tracked particularly well by EWRLS from 07:10:15 PM to 07:10:25 PM. Therefore, the light red line of T2 prediction firstly drops from 27.5°C to 22.5°C due to a decrease in C2. From 07:10:25 PM on the prediction become more reasonable, because the changed parameters of the system model in the cooling system shutdown scenario are updated by EWRLS after 15s from the initial occurrence of the abnormality. To overcome this prediction problem, the speed of the parameter estimation needs to be faster. Therefore, the forgetting factor of the EWRLS can be smaller in this particular situation.

Using the variable table and selecting variables at 07:10:27 PM and 07:10:37 PM, the predicted and actual values of T2 are shown in details. The prediction of T2 at 07:10:27 PM is 31.31°C, while the actual value of T2 after 10s is 32.20°C. The error between the prediction and actual value is 0.89°C. To validate the average prediction error rate in this scenario, the actual T2 values from 07:10:10 PM to 07:13:10 PM, which are the main parts of the T2 value fluctuation, are selected and compared with the corresponding prediction. The average error rate of prediction calculated by equation (6.1) is 3.66%. This average error rate is also shown in the validation summary in Table 6.13.



Figure 6.30: T2 Prediction Validation with Chiller Shutdown

As shown above, prediction performance under the four scenarios including flow rate changes, heater power changes, heater shut down and chiller shut down are demonstrated and validated. The validation results are summarized in Table 6.13. It can be seen that the prediction error rates in these four scenarios are all less than the specifications; thus the variable prediction function can predict the process variables successfully when the systems are operating in either normal or abnormal conditions.

Scenarios	Operation	Start Time	End Time	Length of Sampled Data	Prediction Error Rate	Specification	Validation Result
Α	F1 changes (Flow changes)	5:13:05 PM	5:14:05 PM	60	0.43%	< 1.2% = 0.3/25°C (system is normal)	Pass
В	C2 changes (Power of heater changes)	6:07:45 PM	6:08:45 PM	60	0.41%	< 1.2% = 0.3/25°C (system is normal)	Pass
С	Heater shutdown	6:45:00 PM	6:48:00 PM	180	3.64%	< 6% = 1.5/25°C (system is abnormal)	Pass
D	Chiller shutdown	7:10:10 PM	7:13:10 PM	180	3.66%	< 6% = 1.5/25°C (system is abnormal)	Pass

Table 6.13: Online Validation Results of Prediction Test

6.5 Summary of the Designed Smart Sensor

From the V&V test in this chapter, the data analysis functions of process fault detection function and variable prediction function are integrated into smart sensors successfully. The summary of the performance of designed smart functions can be seen in Table 6.14.

Smart Function	Function	Specification	Performance	
	Fault Detection	Response Time		
	Heater tripping	< 10s	Detect fault of heater tripping	
Process	Chiller shut down	< 10s	Detect fault of cooling system	
Fault Detection	Pipeline plugged	< 10s	Detect fault of pipleline plugged	
	F1 signal loss	< 10s	Detect fault of external sensor signal loss	
	T1 signal loss	< 10s	Detect fault of sensing elements loss	
	Pipeline leak	< 10s	Detect fualt of pipleline leak fault	
	10s ahead Prediction	Prediction Error Rate		
Variable Production	System works around operating point	< 1.2% = 0.3/25°C	Predict 10s ahead of system response	
Freulction	System is in abnormal status	< 6% = 1.5/25°C	Predict 10s ahead of system response	

Table 6.14: Performance of Designed Smart Functions in the Smart Sensor

Based on the V&V test in this chapter, the data analysis functions of process fault detection and variable prediction can successfully be integrated into smart transmitters. Revisiting the limitations of existing smart sensors, and the solutions of the improvement discussed in chapter 1, the main highlights of the designed smart sensors can be summarized as:

Data analysis integration: The functions of conventional smart sensors are manufactured to measure precision value, and transmit measurements. There are few integrated functions in smart transmitters for analyzing equipment health and process conditions. The successful implementation of process fault detection and variable prediction functions in the SMART NCS-TT105 shows that data analysis functions which focus on equipment and process conditions can be integrated into smart transmitters. The integrated data analysis functions extend the capabilities of smart sensors compared with the more conventional measurement function.

Information provision: Up to now, the role of the existing smart sensors are still designed to provide measurement data. Through this thesis, the embedded algorithms

enable the smart sensor to detect process incipient faults and predict the approaching process variable changes. The process information of fault detection and prediction can be used for guiding preventive maintenance and taking early actions when abnormal process status occurs. The embedded algorithms evolve the role of the smart sensors to be valuable information provider.

In summary, the successful implementation of these two functions demonstrates that smart sensors can serve as integrated data analysis and information provider devices. The implementation procedures can used to integrate further data analysis functions and algorithms into smart transmitters to further enrich their functionalities.

6.6 Summary

In this chapter, the SMART NCS-TT105 which is used in a practical test environment is verified and validated. The V&V tests which is summarized in Table 6.15 show that the implemented functions and multitasking design achieving the designed specifications. The designed smart sensor can provide information of process faults and critical process variables' prediction.

Tasks in RTOS of Smart Sensor	Function	Specification	V&V Result
		Cycle Time	
Bidirectional Communication	Exchange data between smart sensor and other systems	200ms	Pass
	Fault Detection	Response Time	
	Heater tripping	< 10s	Pass
Process	Chiller shut down	< 10s	Pass
Fault	Pipeline plugged	< 10s	Pass
Detection	F1 signal loss	< 10s	Pass
	T1 signal loss	< 10s	Pass
	Pipeline leak	< 10s	Pass
	10s ahead Prediction	Prediction Error Rate	
Variable Prodiction	System works around operating point	< 1.2% = 0.3/25°C	Pass
rreuicuon	ction System is in abnormal status	< 6% = 1.5/25°C	Pass

Table 6.15: Summary	/ of	V&V	Test
---------------------	------	-----	------

Chapter 7

7 Summary and Conclusions

In this thesis, design, implementation, verification, and validation of the smart sensors with the process fault detection and variable prediction function are performed. The process information of process faults and future changes of variable can be extracted by SMART NCS-TT105 from the collected data.

7.1 Summary

In this research, the process fault detection and variable prediction are selected as smart functions to enhance the data analysis capabilities of the existing industrial transmitter only using its underutilized computational resources without modifying its footprint nor adding extra hardware. The designed embedded algorithms enable the smart transmitter to provide process information beyond the measurement and communication and allow the data analysis functions to be processed at sensor level. The main research tasks in this thesis are summarized as follows:

- A. Through the literature review, the state of the art smart sensors are designed mainly to provide measurement and transmitting measurement via communication functions. To enhance the capabilities of the existing smart sensors, the desirable functions of process fault detection and variable prediction are selected as smart functions to be integrated into an existing transmitter.
- B. To use the underutilized computational resources of the MCU in the NCS-TT105, the feasible algorithms for extracting the information of fault detection and prediction from collected data are investigated. To realize process fault detection, ARX model is used as model structure, EWRLS is used as parameters estimation and residual generation method, and two-side CUSUM test is used as residual evaluation method. While, to realize variable prediction, Kalman predictor, grey models and multi-step iteration methods are utilized.

- C. The designed parameters and algorithms are programmed and implemented into a transmitter hardware platform. To schedule algorithms with the designed period time, the parameters of RTOS in the implementation platform are discussed and designed. Finally, the three tasks in the Nucleus RTOS of the SMART NCS-TT105 including bi-directional communication task, process fault detection task, and variable prediction task are scheduled in real-time execution.
- D. The verification test shows that the designed smart sensor can be workable in a practical process facility. The offline validation tests the functionalities of the algorithms for realizing process fault detection and variable prediction functions. The online test validates that the designed smart sensor with the embedded algorithms can detect the process faults and predict 10s ahead variable prediction.

The procedure of the investigating suitable functions, studying the algorithms base on the computational resources of the implementation platform, implementing the feasible algorithms, and verification and validation in the test facility can be used as a demonstration for further desirable functions to be integrated into smart transmitters to enhance the existing smart sensor.

7.2 Conclusions

According to verification and validation results of the SMART NCS-TT105, conclusions are summarized as follows:

- A. The data analysis functions which are process fault detection and variable prediction functions can be realized in the smart sensors. The capabilities of the designed smart transmitter can be enhanced by embedding desirable functions without modifying its footprint nor adding external hardware.
- B. The information of faults in the process system and variable prediction can be extracted from the collected data by the smart sensors locally.
- C. The process fault detection and variable prediction functions of condition monitoring systems can be realized in the device layers.

7.3 Future Works

Based on the research achievements in this thesis, subsequent implementation and development work can be performed to further expand the capabilities of smart sensors. The future direction of research can be sketched out as follows:

A. Varying forgetting factor of EWRLS

As discussed in the section 6.4.3, the EWRLS with constant forgetting factor is suited for estimating the system with small changed parameters. To timely track the parameters of the system in fault condition when the model is used for prediction, the varying forgetting factor can be applied and can be changed smaller for forgetting the past error quickly. Therefore, with the varying forgetting factor, the EWRLS method can be used flexibly for different applications.

B. Convergence criteria of parameter estimation in closed loop

If the smart sensors are used in closed loop system to identify the parameters of the system, the identifiability conditions should be validated. In this thesis, the test environment achieve the conditions. However, some practical systems may not meet the identifiability conditions. In [63], some solutions are discussed to overcome this problem. To improve the applicability of the smart sensors, those enhanced solutions can be implemented into smart sensors for estimating parameters in closed loop system.

C. Integration of performance analysis functions

Performance analysis functions can provide information on aging and wear and tear of system components. Integrating such functions into smart sensors enables the smart sensor to analyze such process information for preventive maintenance.

D. Implementation of efficient algorithms

With the growth of embedded algorithms, smart sensor processor workload, power consumption, and functional performance must be balanced. More computationally efficient algorithms need to be developed and implemented in smart sensors.

E. Implementation of wireless communication

Wireless communication technologies, such as Low-Power Wide-Area Network (LPWAN) and wireless personal area networks (WPANs), can also be implemented in smart sensors. Wireless functions enable smart sensors to collaborate with other systems and devices. The data collected by smart sensors could thus be more comprehensive, providing a smart edge in the global network.

In summary, based on current technical strength and forward-looking design, smart sensors can contribute even greater value for industrial production and thus further improve human lives.

References

- [1] R. Frank, Understanding smart sensors. Boston: Artech House, 1996.
- [2] G. C. M. Meijer, Smart sensor systems. Chichester, U.K: J. Wiley & Sons, 2008.
- [3] R. Isermann, Fault-diagnosis applications: model-based condition monitoring: actuators, drives, machinery, plants, sensors, and fault-tolerant systems. Berlin ; Heidelberg ; New York: Springer, 2011.
- [4] "Condition Monitoring Emerson." [Online]. Available: http://www.emerson.com/en-us/automation/asset-management/assetmonitoring/condition-monitoring. [Accessed: 28-Feb-2017].
- [5] L. Wang and R. X. Gao, Condition monitoring and control for intelligent manufacturing. London: Springer, 2006.
- [6] H. Song, D. B. Rawat, S. Jeschke, and C. Brecher, Cyber-Physical Systems : Foundations, Principles and Applications. Saint Louis: Elsevier Science, 2016.
- [7] N. Ida, Sensors, Actuators, and their Interfaces. SciTech Publishing Inc, 2013.
- [8] J. Fraden, Handbook of Modern Sensors. Cham: Springer International Publishing, 2016.
- [9] M. Bhuyan, Intelligent instrumentation: principles and applications. Boca Raton, FL: CRC Press, 2011.
- [10] "Sensors and Actuators, 10 (1986) 239 248 239 SMART SENSORS." [Online]. Available: http://www.academia.edu/4443317/Sensors_and_Actuators_10_1986_239_-_248_239_SMART_SENSORS. [Accessed: 21-Sep-2016].
- [11] R. Frank, Understanding smart sensors, 3rd ed. Boston: Artech House, 2013.
- [12] "SITRANS TF Sensor Systems Siemens." [Online]. Available: http://w3.siemens.com/mcms/sensor-systems/en/processinstrumentation/temperature-measurement/field-transmitter/Pages/sitrans-tf.aspx. [Accessed: 02-Oct-2016].
- [13] K. Iniewski, Smart Sensors for Industrial Applications. CRC Press, 2016.
- [14] H. M. Hashemian, Maintenance of process instrumentation in nuclear power plants. Berlin; New York: Springer, 2006.

- [15] C. Jiang, G. Liu, and J. Jiang, "A self-validating algorithm for hot thermistor constant differential temperature air flow sensor," in Proceedings of 2005 IEEE Conference on Control Applications, 2005. CCA 2005., 2005, pp. 669–674.
- [16] "SmartLine Transmitters | Specification." [Online]. Available: https://www.honeywellprocess.com/en-US/online_campaigns/smartline-pressuretransmitters/Pages/specification.html. [Accessed: 02-Oct-2016].
- [17] H. Lasi, P. Fettke, H. Kemper, T. Feld, and M. Hoffmann, "Industry 4.0," Bus. Inf. Syst. Eng., vol. 6, no. 4, pp. 239–242, 2014.
- [18] K. M. GmbH, "OPTISWIRL 4070." [Online]. Available: http://krohne.com/en/products/flow-measurement/vortex-flowmeters/optiswirl-4070/. [Accessed: 03-Oct-2016].
- [19] "Emerson Process Management 3051S Advanced Diagnostics | Rosemount."
 [Online]. Available: http://www2.emersonprocess.com/en-US/brands/rosemount/Pressure/Pressure-Transmitters/3051S-Advanced-Diagnostics/Pages/index.aspx. [Accessed: 03-Oct-2016].
- [20] T. D. Morton, Embedded microcontrollers. Upper Saddle River, N.J: Prentice Hall, 2001.
- [21] M. Colnarič, W. A. Halang, and D. Verber, "Distributed Embedded Control Systems: Improving Dependability with Coherent Design," Springer E-Books, 2008.
- [22] R. Stackowiak, A. Licht, V. Mantha, and L. Nagode, "Big Data and the Internet of Things: Enterprise Information Architecture for a New Age," Springer EBooks, 2015.
- [23] "Emerson Process Management Petroleum Refining Essential Asset Monitoring | Emerson." [Online]. Available: http://www2.emersonprocess.com/en-US/industries/refining/Solutions/Pages/EssentialAssetMonitoring.aspx. [Accessed: 19-Oct-2016].
- [24] "SIMATIC PCS 7 Condition Monitoring Library." [Online]. Available: https://mall.industry.siemens.com/mall/en/WW/Catalog/Products/10223953?tree= CatalogTree. [Accessed: 20-Oct-2016].
- [25] "Process & Equipment Monitoring." [Online]. Available: https://www.honeywellprocess.com/en-US/explore/products/advancedapplications/uniformance/uniformance-asset-sentinel/Pages/process-equipmentmonitoring.aspx. [Accessed: 22-Oct-2016].
- [26] H. Schödel, "Utilization of fuzzy techniques in intelligent sensors," Fuzzy Sets Syst., vol. 63, no. 3, pp. 271–292, 1994.

- [27] J. Rivera, M. Carrillo, M. Chacón, G. Herrera, and G. Bojorquez, "Self-Calibration and Optimal Response in Intelligent Sensors Design Based on Artificial Neural Networks," Sensors, vol. 7, no. 8, pp. 1509–1529, Aug. 2007.
- [28] "16-bit 32-bit MCU | Low-power MCUs | Overview | Microcontrollers (MCU) | TI.com." [Online]. Available: http://www.ti.com/lsds/ti/microcontrollers_16bit_32-bit/msp/overview.page. [Accessed: 24-Sep-2016].
- [29] "LPC4000 32-bit MCUs|ARM Cortex-M4 Cores|NXP." [Online]. Available: http://www.nxp.com/products/microcontrollers-and-processors/armprocessors/lpc-cortex-m-mcus/lpc-cortex-m4/lpc4000-cortexm4:MC_1403790399405. [Accessed: 24-Sep-2016].
- [30] D. C. Swanson, Signal processing for intelligent sensor systems with MATLAB, 2nd ed. Boca Raton, Fla: CRC Press, 2012.
- [31] S. Middelhoek and A. C. Hoogerwerf, "Smart sensors: when and where?," Sens. Actuators, vol. 8, no. 1, pp. 39–48, Sep. 1985.
- [32] J. M. Giachino, "Smart sensors," Sens. Actuators, vol. 10, no. 3, pp. 239–248, Nov. 1986.
- [33] G. Meijer, M. Pertijs, and K. Makinwa, Smart Sensor Systems: Emerging Technologies and Applications, 1st ed. Wiley Publishing, 2014.
- [34] K. Najafi, "Smart sensors," J. Micromechanics Microengineering, vol. 1, no. 2, p. 86, 1991.
- [35] A. Elouardi, S. Bouaziz, A. Dupret, L. Lacassagne, J. O. Klein, and R. Reynaud, "A smart sensor-based vision system: implementation and evaluation," J. Phys. Appl. Phys., vol. 39, no. 8, pp. 1694–1705, 2006.
- [36] W. Badawy and G. A. Julien, System-on-Chip for Real-Time Applications. Springer Science & Business Media, 2002.
- [37] "What is MEMS Technology?" [Online]. Available: https://www.memsnet.org/about/what-is.html. [Accessed: 25-Sep-2016].
- [38] A. Wieferink, H. Meyr, and R. Leupers, "SOC Design Methodologies," in Retargetable Processor System Integration into Multi-Processor System-on-Chip Platforms, Springer Netherlands, 2008, pp. 7–23.
- [39] C. Piguet, R. Reis, and D. Soudris, Eds., VLSI-SoC: Design Methodologies for SoC and SiP, vol. 313. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010.
- [40] J. E. Brignell, "The future of intelligent sensors: A problem of technology or ethics?," Sens. Actuators Phys., vol. 56, no. 1–2, pp. 11–15, 1996.

- [41] "Fieldbus and Networking in Process Automation," CRC Press, 14-May-2014. [Online]. Available: https://www.crcpress.com/Fieldbus-and-Networking-in-Process-Automation/Sen/p/book/9781466586765. [Accessed: 26-Sep-2016].
- [42] V. Ç. Güngör, Ed., Industrial wireless sensor networks: applications, protocols, and standards. Boca Raton, FL: CRC Press, Taylor & Francis Group, 2013.
- [43] Q. Wang and J. Jiang, "Comparative Examination on Architecture and Protocol of Industrial Wireless Sensor Network Standards," IEEE Commun. Surv. Tutor., vol. 18, no. 3, pp. 2197–2219, thirdquarter 2016.
- [44] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," Comput. Netw., vol. 52, no. 12, pp. 2292–2330, 2008.
- [45] H. Labiod, H. Afifi, and C. De Santis, Wi-Fi, Bluetooth, Zigbee and WiMAX. Dordrecht: Springer, 2007.
- [46] S. Soloman, Sensors and control systems in manufacturing, 2nd ed. New York: McGraw-Hill, 2010.
- [47] H. Yamasaki, Ed., Intelligent sensors. Amsterdam; New York: Elsevier, 1996.
- [48] R. Doraiswami and J. Jiang, "An intelligent sensor to monitor power system stability, performance and diagnose failures," Power Syst. IEEE Trans. On, vol. 5, no. 4, pp. 1432–1438, 1990.
- [49] M. M. Gupta, N. K. Sinha, and IEEE Neural Networks Council, Eds., Intelligent control systems: theory and applications. Piscataway, NJ: IEEE Press, 1996.
- [50] "BOA Overview Cameras." [Online]. Available: https://www.teledynedalsa.com/imaging/products/vision-systems/cameras/-boaoverview/. [Accessed: 01-Oct-2016].
- [51] R. Isermann, Fault-diagnosis systems: an introduction from fault detection to fault tolerance. Berlin; New York: Springer, 2006.
- [52] P. M. Frank, "Fault diagnosis in dynamic systems using analytical and knowledgebased redundancy," Automatica, vol. 26, no. 3, pp. 459–474, 1990.
- [53] J. Gertler, Fault detection and diagnosis in engineering systems. New York: Marcel Dekker, 1998.
- [54] R. Isermann, "Model-based fault-detection and diagnosis status and applications," Annu. Rev. Control, vol. 29, no. 1, pp. 71–85, 2005.
- [55] J. Ma and J. Jiang, "Applications of fault detection and diagnosis methods in nuclear power plants: A review," Prog. Nucl. Energy, vol. 53, no. 3, pp. 255–266, 2011.

- [56] P. Kadlec, B. Gabrys, and S. Strandt, "Data-driven Soft Sensors in the process industry," Comput. Chem. Eng., vol. 33, no. 4, pp. 795–814, 2009.
- [57] L. Fortuna, Ed., Soft sensors for monitoring and control of industrial processes. London: Springer, 2007.
- [58] R. Isermann and P. Ballé, "Trends in the application of model-based fault detection and diagnosis of technical processes," Control Eng. Pract., vol. 5, no. 5, pp. 709– 719, 1997.
- [59] S. Simani, C. Fantuzzi, and R. Patton, Model-based fault diagnosis in dynamic systems using identification techniques. London; New York: Springer, 2003.
- [60] I. Hwang, S. Kim, Y. Kim, and C. E. Seah, "A Survey of Fault Detection, Isolation, and Reconfiguration Methods," IEEE Trans. Control Syst. Technol., vol. 18, no. 3, pp. 636–653, 2010.
- [61] L. H. Chiang, R. D. Braatz, and E. Russell, Fault detection and diagnosis in industrial systems. London; New York: Springer, 2001.
- [62] S. S. Haykin, Adaptive filter theory, 4th ed. Upper Saddle River, N.J: Prentice Hall, 2002.
- [63] R. Isermann and M. Münchhof, "Identification of Dynamic Systems: An Introduction with Applications," Springer EBooks, 2011.
- [64] R. C. K. Lee, Optimal estimation, identification, and control. Cambridge, Mass: M.I.T. Press, 1964.
- [65] X. Yu and J. Jiang, "A survey of fault-tolerant controllers based on safety-related issues," Annu. Rev. Control, vol. 39, pp. 46–57, 2015.
- [66] F. Gustafsson, Adaptive filtering and change detection. Chichester: Wiley, 2000.
- [67] M. Basseville and I. V. Nikiforov, Detection of Abrupt Changes: Theory and Application. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1993.
- [68] S. R. Mohanty, A. K. Pradhan, and A. Routray, "A Cumulative Sum-Based Fault Detector for Power System Relaying Application," IEEE Trans. Power Deliv., vol. 23, no. 1, pp. 79–86, Jan. 2008.
- [69] P. P. Kanjilal, Adaptive prediction and predictive control. London, U.K: P. Peregrinus on behalf of Institution of Electrical Engineers, 1995.
- [70] W.C. Cave, Prediction Theory for Control Systems. Spring Lake, NJ: Prediction Systems, Inc., 2014.

- [71] S. V. Vaseghi, Advanced digital signal processing and noise reduction, 4th ed. Chichester, U.K: J. Wiley & Sons, 2008.
- [72] J. G. De Gooijer and R. J. Hyndman, "25 years of time series forecasting," Int. J. Forecast., vol. 22, no. 3, pp. 443–473, 2006.
- [73] J. Deng, "Control problems of grey systems," Syst. Control Lett., vol. 1, no. 5, pp. 288–294, Mar. 1982.
- [74] S. Liu and J. Y.-L. Forrest, Grey Systems: Theory and Applications. Berlin; Heidelberg: Springer, 2011.
- [75] S. Liu, J. Forrest, and Y. Yang, "A brief introduction to grey systems theory," Grey Syst., vol. 2, no. 2, pp. 89–104, 2012.
- [76] S. Liu, J. Forrest, and Y. Yang, "A Summary of The Progress in Grey System Research," in Proceedings of 2013 IEEE International Conference on Grey systems and Intelligent Services (GSIS), 2013, pp. 1–10.
- [77] L. Ljung, System identification: theory for the user, 2nd ed. Upper Saddle River, NJ: Prentice Hall PTR, 1999.
- [78] F. Piltan, S. TayebiHaghighi, and N. B. Sulaiman, "Comparative Study between ARX and ARMAX System Identification," Int. J. Intell. Syst. Appl., vol. 9, no. 2, pp. 25–34, Feb. 2017.
- [79] J. Miklěs and M. Fikar, Process modelling, identification, and control. Berlin; New York: Springer, 2007.
- [80] M. S. Grewal and A. P. Andrews, Kalman filtering: theory and practice using MATLAB, 3rd ed. Hoboken, N.J: Wiley, 2008.
- [81] N. Assimakis and M. Adam, "Kalman Filter Riccati Equation for the Prediction, Estimation, and Smoothing Error Covariance Matrices," ISRN Comput. Math., vol. 2013, pp. 1–7, 2013.
- [82] C. Chen, Linear system theory and design, 3rd ed. New York: Oxford University Press, 1999.
- [83] C. E. Howard, "RTOS for a software-driven world," Mil. Aerosp. Electron. Westborough, vol. 22, no. 3, p. 28,30,32-34,36-37,47, Mar. 2011.
- [84] E. Verhulst, R. T. Boute, J. M. S. Faria, B. H. C. Sputh, and V. Mezhuyev, Formal Development of a Network-Centric RTOS. Boston, MA: Springer US, 2011.
- [85] M. O. Farooq and T. Kunz, "Operating Systems for Wireless Sensor Networks: A Survey," Sens. 14248220, vol. 11, no. 6, pp. 5900–5930, Jun. 2011.
- [86] "Selecting an RTOS for the data acquisition fast track," Electron. Eng. Lond., p. 31, Jan. 1999.
- [87] C. L. Liu and J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment," J ACM, vol. 20, no. 1, pp. 46–61, Jan. 1973.
- [88] "Accelerated Technology's Nucleus RTOS, Instrumental in Development Of Panametrics' Newest Flowmeter," PR Newswire; New York, New York, United States, p. 1, 28-Jan-2002.
- [89] "Mentor Graphics Announces Softing Industrial Automation EtherNet/IP and OPC UA Support in Nucleus RTOS and Mentor Embedded Linux Runtimes | ARC Advisory Group." [Online]. Available: https://www.arcweb.com/blog/mentorgraphics-announces-softing-industrial-automation-ethernetip-and-opc-ua-supportnucleus. [Accessed: 24-Apr-2017].
- [90] R. Toulson and T. Wilmshurst, Fast and effective embedded systems design: applying the ARM mbed, 1st ed. Oxford ; Boston, MA: Elsevier/Newnes, 2012.
- [91] H. Dong and J. Lu, "Application of MicroC/OS in the control system of intelligent robocup," in Fifth World Congress on Intelligent Control and Automation (IEEE Cat. No.04EX788), 2004, vol. 6, p. 4768–4772 Vol.6.
- [92] "FreeRTOS Market leading RTOS (Real Time Operating System) for embedded systems with Internet of Things extensions." [Online]. Available: http://www.freertos.org/. [Accessed: 24-Apr-2017].
- [93] "Microcyber Corporation." [Online]. Available: http://www.microcyber.cn/introduction/. [Accessed: 21-Nov-2016].
- [94] "NCS-TT105 Temperature Transmitter." [Online]. Available: http://www.microcyber.cn/thedocument/.
- [95] "Nucleus PLUS Internals Manual."
- [96] "Nucleus RTOS." [Online]. Available: https://www.mentor.com/embeddedsoftware//nucleus/. [Accessed: 28-Nov-2016].
- [97] "Industrial platinum resistance thermometers and platinum temperature sensors," IEC 607512008, 2008.
- [98] "Thermocouples Part 1: Reference tables," IEC 60584-11995, 1995.
- [99] "PROFIBUS PA New standards in process control Siemens." [Online]. Available: http://w3.siemens.com/mcms/process-control-systems/en/distributedcontrol-system-simatic-pcs-7/simatic-pcs-7-systemcomponents/communication/profibus-pa/pages/profibus-pa.aspx. [Accessed: 09-Nov-2016].

- [100] M. P. Kazmierkowski, "Industrial Communication Technology Handbook, 2nd Ed. [Book News]," IEEE Ind. Electron. Mag., vol. 8, no. 2, pp. 67–68, Jun. 2014.
- [101] "PROFIBUS Profile for Process Control Devices Version 3.02." Apr-2009.
- [102] I. Ramadhani, Jondri, and R. Rismala, "Prediction of multi currency exchange rates using correlation analysis and backpropagation," in 2016 International Conference on ICT For Smart Society (ICISS), 2016, pp. 63–68.
- [103] L. Ljung, System identification: theory for the user, 2nd ed. Upper Saddle River, NJ: Prentice Hall PTR, 1999.
- [104] E. W. Weisstein, "Sample Variance Computation." [Online]. Available: http://mathworld.wolfram.com/SampleVarianceComputation.html. [Accessed: 11-Dec-2016].
- [105] S. Liu and J. Y. Forrest, Grey Information: Theory and Practical Applications. London: Springer, 2006.
- [106] J. Ma, "PTCF Subsystems, London." CAN: Western CIES Research Group, 2013.
- [107] "ABB Freelance DCS." [Online]. Available: http://new.abb.com/controlsystems/essential-automation/freelance. [Accessed: 28-Jan-2017].
- [108] "Distributed Control System SIMATIC PCS 7 New standards in process control-Siemens." [Online]. Available: http://w3.siemens.com/mcms/process-controlsystems/en/distributed-control-system-simatic-pcs-7/pages/distributed-controlsystem-simatic-pcs-7.aspx. [Accessed: 28-Jan-2017].
- [109] "Operator Software New standards in process control Siemens." [Online]. Available: http://w3.siemens.com/mcms/process-control-systems/en/distributedcontrol-system-simatic-pcs-7/simatic-pcs-7-system-components/operatorsystem/os-software/pages/operator-software.aspx. [Accessed: 28-Jan-2017].
- [110] "PROFIBUS PA Simple and Safe in Hazardous Zones PI North America Blog," PI North America (PTO), 29-Jul-2014. [Online]. Available: http://us.profinet.com/profibus-pa-safe-in-hazardous-zones/. [Accessed: 09-Nov-2016].

Appendices

Appendix A: Standards for Sensing in NCS-TT105

A. IEC 60751

IEC 60751 [97] standard specifies the relationship between temperature and industrial platinum resistance. RTD is used base on a phenomenon known as thermal resistivity, which shows a principle that metal resistance increases if its temperature increases. Using this principle, the temperature can be calculated by measuring RTD resistance. The following equations show this relationship [97],

$$R_{t} = R_{0}[1 + At + Bt^{2} + C(t - 100 °C)t^{3}] if -200 °C < t < 0 °C$$

$$R_{t} = R_{0}(1 + At + Bt^{2}) if 0 °C < t < 850 °C$$
where
$$R_{0} : the resistance at t = 0 °C$$

$$R_{t} : the resistance at the temperature t$$

$$A = -3.9083 \times 10^{-3} \Omega \cdot °C^{-1}$$

$$B = -5.775 \times 10^{-7} \Omega \cdot °C^{-2}$$

$$C = -4.183 \times 10^{-12} \Omega \cdot °C^{-4}$$
(A.1)

PT100 and PT 1000 The RTDs are two common used RTDs. They have a range of -200 to 850 °C. To convert resistance measurement into temperature value, the transfer calculation methods can use tables of resistance values to find correspondence or computation equations.

B. IEC 60584-1

IEC 60584-1 [98] standard specifies the relationship between temperature and multiple types of thermocouples using polynomial equations and tables. TC is used base on an effect known as Electromotive Force (EMF) in hot junction refer to cold junction. Generally, the cold junction is on sensors or control cabinet side, while the hot junction is the sensing probe with measuring object. Once the temperature applied in hot junction, the millivolt which refer to the dissimilar of two junctions will be generated. Measuring this millivolt, the temperature difference between hot junction and cold junction can be derived.

The typically used TCs include types E, J, K, T, R, S, B and N. The reference polynomials, which express the relationship between temperature and measured thermocouple EMF corresponding with different types of TCs, are different. The smart sensor can decide to use related equations refer to the TC type configuration. The temperature range of TCs is large. Most of these types are higher than 1000 °C. Therefore, if the temperature of the measured objects is higher than 850 °C, TCs are suitable to be used.

Appendix B: Introduction of PROFIBUS-PA

A. General Introduction of PROFIBUS-PA Fieldbus

PROFIBUS-PA is an industrial Fieldbus used for connecting field level sensors and actuators. It is one of the leading Fieldbus technologies, widely used in automation systems, especially in harsh and potentially explosive environments in the process industries [99]. The physical layer of PROFIBUS-PA is designed according to IEC 61158-2, while its profile uses IEC 61784-1 [100].

The key features of PROFIBUS-PA on the physical layer are a digital communication bus with powering and hazardous environment deployment. According to IEC 61158-2, Manchester Encoded and Bus Powered (MBP) technology is used with PROFIBUS-PA [100]. This technology allows the PROFIBUS-PA to be 31.25 Kbit/s as data transmission rate and a maximum main cable length of 1900m. According to the PROFIBUS-PA standard [101], the numbers of inputs and output bytes must not exceed 244 bytes. Each channel employs 5 bytes, in which sensor measurement uses 4 bytes with float data format, while related channel diagnostic messages use 1 byte. PROFIBUS-PA has been verified by the Fieldbus Intrinsically Safe Concept (FISCO) [110] and IEC 60079-11. The FISCO model validates and standardizes the application of communication networks used in potentially explosive environments. This enables devices which are employed in hazardous areas to communicate with a control system via PROFIBUS-PA Fieldbus. For such hazardous environment scenarios, the maximum length of the PROFIBUS-PA should be less than 1000m, and the numbers of devices on the bus must be decreased with respect to power supply limitations. More detailed specifications of a PROFIBUS-PA can be found in Table B.1.

Technical specifications	PROFIBUS PA
Data transmission	MBP
Transmission rate	31.25 Kbps
Cable	Two-wire shielded
Type of protection	Ex ia/ib/ic
Тороlоду	Line, tree, ring
Safety Integrated	•
Control in the field	-
Interoperability	•
Field devices per segment/coupler	31 (typical 16 20)
Field devices per link	64
Active field distributors per segment/coupler - AFD - AFDISD or combinations of AFDISD and AFD	8 5
Max. total current consumption of all field devices	1 A
Max. cable length per segment	1 900 m

Table B.1: The Specifications of PROFIBUS-PA [99]

B. Introduction of the PA devices Profile

The device profile which is designed to fulfill PROFIBUS-PA standards and functions is called PA devices in [101]. A PA device profile complies with Fieldbus protocols and profile specifications, and defines all functions and parameters for different devices. The internal structure of a PA devices profile is shown in Figure B.1. The current PA device profile released by PROFIBUS & PROFINET International (PI) group is version 3.02 [101]. Function blocks, including the Physical Block (PB), Transducer Block (TB), and Function Block (FB), are described in this profile according to IEC 61804.



Figure B.1: Block Structure of a PA devices Profile [100]

The roles of these three types of blocks are summarized as follows [101][100]:

Physical Block (PB): A PB specifies a group of parameters that describe the characteristic of a particular device. FB is used as an ID card for identification by the upper layer Fieldbus controllers. For instance, the device name, hardware version, software version, manufacturer name, and identification number are all stored in the block. Only one PB needs to be used per device profile.

Transducer Block (TB): A TB contains the measurement data and channel parameters. It transfers ADC raw data to measurement data and provides access from the sensing elements to the corresponding function blocks. Different types of sensing elements, such as temperature, pressure, or level, have their specific TB. For instance, as shown in Figure B.2, a temperature sensor is configured one temperature TB for transferring two channels measurement data. The parameters of the sensing elements such as a 2, 3, and 4 wire connection, and input range for linearization, are configured in TB.

Function Block (FB): A FB is used for setting the parameters of measurement data and exchanging data with Fieldbus controllers. Therefore, FBs are the interface blocks for exchanging the internal data with fieldbus network. There are four types of FBs, such as Analog Input FB, Analog Output FB, Digital Input FB, and Digital Output FB. The



internal diagram of an Analog Input FB for transferring internal data to fieldbus is shown in Figure B.3.

Figure B.2: Internal Diagram of a Temperature Transducer Block [101]



Figure B.3: Internal Diagram of an Analog Input Function Block [101]

Appendix C: Introduction of Nucleus RTOS

A. Terminologies of Nucleus RTOS

To understand the task scheduling and program running context, some important theorems of Nucleus RTOS are need to be introduced. They are shown as follows:

Kernel tick: Nucleus RTOS uses a periodic timer service – a so-called kernel tick. Once the system is compiled and started up, kernel tick will work as the heartbeat of the RTOS. The smaller the value of this timer, the higher resolution of the timing and faster the response of the highest task can be. However, a small value of the kernel tick can results in less efficient of RTOS by frequent interruption in its kernel.

<u>**Task state machine:**</u> The Nucleus RTOS defines five task states, including EXECUTING, READY, SUSPENDED, TERMINATED and FINISHED. When multiple tasks are scheduled by the preemptive kernel of the RTOS, the tasks are transferred among three typical states, which are EXECUTING, READY, and SUSPENDED. Through those states transferring, the system can take advantage of the time spent waiting in one task to execute another task, allowing multitask processing.

Preemptive scheduling: Preemption is the action of suspending a task with lower priority when a higher priority task is ready. Once kernel tick occurs, RTOS kernel calls interrupt service routine (ISR) to find the highest priority task in the READY queue. Then the highest priority task is executed promptly. Other lower priority READY tasks need to wait and the preempted task are transferred to SUSPENDED status. Through preemptive scheduling in Nucleus RTOS, the response time of higher priority tasks can be guaranteed.

Task priority: Based on RMS rules, the task with shorter period time is assigned higher priorities. In Nucleus RTOS, the service named NU_Create_Task() is used for creating a task and assigning priority for a task. The task with higher priority is assigned with lower numeric priority integer value in NU_Create_Task(). It is suggested that the priority value of user tasks in Nucleus RTOS are set from 10 to 255.

B. Configuration of Kernel tick in RTOS

To demonstrate the configuration of kernel tick, three tasks named Task11, Task12 and Task13 are used as examples and need to be scheduled by Nucleus RTOS systems. Following the rules of RMS, the setting of kernel tick is need to be discussed based on different scenarios of T_e and T_p of three tasks. Two examples based on two kernel tick scenarios are discussed. They can be a good guidance for designing kernel tick in RTOS.

1) Scenario-1 -
$$\sum T_{ne} < GCD[T_{np}]$$

In the first scenario, the GCD of all three tasks period time are larger than the sum of all three tasks execution time. The execution time and period time of three tasks are shown in Table C.1. The assumptions of execution time of three tasks are 30ms, 70ms, and 40ms respectively. The corresponding period time, which comes from application requirements, are 200ms, 400ms, and 1000ms. The kernel tick time, event response time, utilization and workload of the processor are analyzed and calculated at bellowing.

Tasks	Execution time(ms)	Period time(ms)
11	30	200
12	70	400
13	40	1000

Table C.1: Execution Time and Period Time of Tasks in Scenario-1

Following RMS rules, the Task11 which is shorter period time is assigned with the highest priority11. The second highest priority is the Task12 with the priority 12. The lowest priority is Task13 with priority 13. From the equation (5.4), the kernel tick can meet the first kernel tick scenario. Therefore, from calculation in the equation (C.1), the kernel tick can be set as 200ms.

$$\sum [T_{1e}, T_{2e}, \cdots, T_{ne}] = 140 < T_{Tick} \le GCD[T_{1p}, T_{2p}, \cdots, T_{np}] = 200$$
(C.1)

The task flow of multitasking scheduling in the first scenario is shown Figure C.1: Task Flow of Multitasking Scheduling in Scenario-1. Every 200ms, the system kernel executes ISR and the highest priority READY task is executed. The three tasks are executed in the sequence of priority and completed within one kernel tick interval. Every 1000ms, the three tasks gather together due to the requirement of period time and needs to be processed within one kernel tick interval. As shown in Figure C.1, three tasks are scheduled in the sequence of priority within one kernel tick interval every 1000ms.

Kernel Tick = 200ms, Every slot = 10ms																								
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Task11																								
Task12																								
Task13																								

Figure C.1: Task Flow of Multitasking Scheduling in Scenario-1

According to equation (5.7), the upper bound of processor utilization is 77.98%. From (5.8), the workload of microprocessor is 36.5%, which is calculated from (5.10).

$$WL_{processor} = \frac{30}{200} + \frac{70}{400} + \frac{40}{1000} = 36.5\%$$
 (C.2)

According to equation (5.3), the worst event response time for each three task is 230ms, 500ms, and 1140ms respectively, which are calculated in (5.11). The event response time of all tasks can be determined clearly.

$$T_{1 ever} \le T_{1p} + T_{1r} + T_{1 int} = 200 + 0 + 0 = 230$$

$$T_{2 ever} \le T_{2p} + T_{2r} + T_{2 int} = 400 + 30 + 0 = 430$$

$$T_{3 ever} \le T_{3p} + T_{3r} + T_{3 int} = 1000 + (30 + 70) + 0 = 1100$$
(C.3)

In summary, kernel tick is set as 200ms in scenario-1.With this configuration, all the parameters of the three tasks are summarized in Table C.2.

Tasks	Execution time (ms)	Period time (ms)	Priority	Worst event response time(ms)	Workload
11	30	200	11	230	15.0%
12	70	400	12	430	17.5%
13	40	1000	13	1100	4.0%

Table C.2: Analysis of Three Tasks in Scenario-1

2) Scenario 2 - $\sum T_{ne} > GCD[T_{np}]$

In the second scenario, the GCD of all three tasks period time are small than the sum of all three tasks execution time. The execution time and period time of three tasks are shown in Table C.3. The assumptions of execution time of three tasks are 40ms, 80ms, and 120ms respectively. The corresponding period times of three task, which are same as previous scenario, are 200ms, 400ms, and 1000ms.

Table C.3: Execution Time and Period Time of Tasks in Scenario-2

Tasks	Execution time(ms)	Period time(ms)
11	40	200
12	80	400
13	120	1000

Because of the period times of the three tasks are not changed, following RMS rules, the priority of Task11, Task12 and Task13 are the same as previous scenario, which are 11, 12, and 13 respectively. Since the sum of the execution time of three tasks is increased, as shown in (C.4), the kernel tick cannot meet the kernel tick scenario-1. Therefore, the kernel tick have to be set following the rule from kernel tick scenario-2 which are discussed in (5.5) and (5.6). The kernel tick is set as 20ms, which is shown in (C.6).

$$T_{Tick} \le GCD[T_{1p}, T_{2p}, \cdots, T_{np}] = 200$$

$$T_{Tick} > \sum [T_{1e}, T_{2e}, \cdots, T_{ne}] = 240$$
 (C.4)

$$T_{Tick} \le GCD[T_{1p}, T_{2p}, \cdots, T_{np}] = 200$$

$$T_{Tick} > \sum [T_{1e}, T_{2e}, \cdots, T_{ne}] = 240$$
 (C.5)

$$T_{Tick} = 20ms < 40ms = Min[T_{1e}, T_{2e}, \dots, T_{ne}]$$

Mod[GCD(T_{1p}, T_{2p}, \dots, T_{np}), T_{Tick}] = Mod(200, 20) = 0 (C.6)

The task flow of multitasking scheduling for the second scenario is shown in Figure C.2. Every 20ms, the system kernel executes ISR. At that time the highest priority READY task is executed and the lower tasks are preempted or interrupted. It needs to be care about that the Task13 is interrupted by Task11 at slot 11 and 12. This means that the low priority tasks could be interrupted by higher priority task several times if the period time of higher priority task is short. Therefore, the worst event response time is extended in some special case in which lower priority tasks are interrupted and preempted frequently.

Kernel Tick = 20ms, Every slot = 20ms																								
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Task11																								
Task12																								
Task13																								

Figure C.2: Task Flow of Multitasking Scheduling in Scenario-2

According to equation (5.7), the upper bound of processor utilization is 77.98%. From (5.8), the workload of the processor is 52.0%, which is calculated from equation (C.7).

$$WL_{processor} = \frac{40}{200} + \frac{80}{400} + \frac{120}{1000} = 52.0\%$$
 (C.7)

According to equation (5.3), the worst event response time for each three task is 200ms, 440ms, and 1160ms respectively, which are calculated in (C.8):

$$T_{1 ever} \le T_{1p} + T_{1r} + T_{1 int} = 200 + 0 + 0 = 200$$

$$T_{2 ever} \le T_{2p} + T_{2r} + T_{2 int} = 400 + 40 + 0 = 440$$

$$T_{3 ever} \le T_{3p} + T_{3r} + T_{3 int} = 1000 + (40 + 80) + \underline{40} = 1160$$
(C.8)

It can be seen that the obvious difference of event response time between the two scenarios is cause by the interrupted time of lower priority task by higher priority tasks. As shown in equation (C.8), the T_{1e} is added twice when calculating T_{3ever} , because the Task13 are interrupted twice by Task11.

As shown above, kernel tick is set as 20ms in scenario-2. With this configuration, all the parameters of the three tasks are summarized in Table C.4. In this scenario, the execution time of tasks is almost no restrictions for setting kernel tick if kernel tick is a small value. The tasks with short period time and higher priority can be guaranteed to be executed by kernel. However, the drawbacks of this scenario show that the event response time of low priority tasks is extended dynamically by interruptions from higher priority tasks. If the higher priority tasks have shorter period time while lower priority tasks have longer of the execution time, the worst event response time of the task with lower priority tasks will be enlarged.

Tasks	Execution time (ms)	Period time (ms)	Priority	Worst event response time(ms)	Workload
11	40	200	11	230	20.0%
12	80	400	12	440	20.0%
13	120	1000	13	1160	12.0%

Table C.4: Analysis of Three Tasks in Scenario-2

Curriculum Vitae

Name:	An He
Post-secondary Education and Degrees:	Beijing Information Technology Institute Beijing, China 2002-2006 B.A.
	North China University of Technology Beijing, China 2011-2014 M.Eng.
	The University of Western Ontario London, Ontario, Canada 2016-2017 M.E.Sc.
Related Work	Technical Application Support in SIEMENS Ltd., China 2008-2015
	HANGXING International Automation Engineering Co. Ltd, Beijing, China 2006-2007
Experience	Industrial Control System Design and Commissioning
Lapononee	Excellent in Siemens PLC, DCS, Network, and HMI systems Experience in Factory-Automation and Process-Automation, and SCADA systems