THESIS APPROVAL

GRADUATE SCHOOL, KASETSART UNIVERSITY

Master of Engineering (Information and Communication Technology for Embedded Systems)
DEGREE

Information and Communication Technology for Embedded Systems Electrical Engineering
FIELD **DEPARTMENT**

TITLE: Embedded Multi Sensors for Hydrological Monitoring System

NAME: Mr. Rattanasak Kasettham

THIS THESIS HAS BEEN ACCEPTED BY

THESIS ADVISOR
(Assistant Professor Dusit Thanapatay, Ph.D.)

THESIS CO-ADVISOR
(Mr. Rachaporn Keinprasit, Ph.D.)

THESIS CO-ADVISOR
(Associate Professor Nobuhiko Sugino, Ph.D.)

DEPARTMENT HEAD
(Assistant Professor Teerasit Kasetkasem, Dr.E.)

APPROVED BY THE GRADUATE SCHOOL ON _____

DEAN
(Associate Professor Gunjana Theeragool, D.Agr.)

THESIS

EMBEDDED MULTI SENSORS FOR HYDROLOGICAL MONITORING SYSTEM

RATTANASAK KASETTHAM

A Thesis Submitted in Partial Fulfillment of
the Requirements for the Degree of
Master of Engineering (Information and Communication Technology for Embedded Systems)
Graduate School, Kasetsart University
2011

Rattanasak Kasettham 2011: Embedded Multi Sensors for Hydrological Monitoring System. Master of Engineering (Information and Communication Technology for Embedded Systems), Major Field: Information and Communication Technology for Embedded Systems, Department of Electrical Engineering. Thesis Advisor: Assistant Professor Dusit Thanapatay, Ph.D. 61 pages.

The purpose of this research is to design and implement the low cost sensor nodes for Hydrological Monitoring System. It consists of low cost flow velocity sensors and low cost flow direction sensor. This sensor node is used to measure the flow velocity in various depths and the flow direction of the river. The flow velocity sensor is designed by using a propeller and a Hall Effect sensor. The rotation of propeller is designed to provide a pulse when it passes Hall Effect sensor. So, the frequency of the pulses is proportional to the velocity of water. The flow direction sensor is implemented by using a compass sensor, magnet, vane and controller (MSP430). It is used to measure the angle between the flow direction of the water and the flow velocity sensor. Then, the angle is used to improve the accuracy of flow velocity.

For experimental results, demonstrate that the low cost flow velocity sensors have a low root mean square error. So, they can measure the flow velocity in general flow-velocity condition. And the low cost flow direction sensor has an acceptable accuracy but it need to calibration before using. Furthermore, the pitch and roll of the flow direction sensor has effect to the accuracy. Then, the sensor node should install in horizontal to prevent the error. Therefore, the low cost sensor nodes which consist of low cost flow velocity sensors and low cost flow direction sensor have efficiency for using in the hydrological monitoring station.

_____ _____ ___ / ___ / ___

Student's signature Thesis Advisor's signature

ACKNOWLEDGEMENTS

I would like to grateful thank and deeply indebted to Dr. Dusit Thanapatay my thesis advisor, Dr. Rachaporn Keinprasit my co-advisors from National Electronics and Computer Technology Center, and Prof. Nobuhiko Sugino my co-advisor from Tokyo Institute of Technology for advice, encouragement and valuable suggestion for completely writing of thesis.

This research is financially supported by Thailand Advanced Institute of Science and Technology - Tokyo Institute of Technology (TAIST-Tokyo Tech), National Science and Technology Development Agency (NSTDA), Tokyo Institute of Technology (Tokyo Tech), National Research Council of Thailand (NRCT) and Kasetsart University (KU).

Rattansasak Kasettham
February 2011

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF ABBREVIATIONS

ADC	=	Analog to Digital Converter
A/D	=	Analog to Digital Converter
ASCII	=	American Standard Code for Information Interchange
D/A	=	Digital to Analog Converter
DCO	=	Digitally Controlled Oscillator
I^2C	=	Inter-Integrated Circuit
USART	=	Universal synchronous/asynchronous receiver/transmitter
PCB	=	Printed Circuit Board
ASIC	=	Application-specific integrated circuit
RISC	=	Reduced Instruction Set Computer
IC	=	Integrated Circuit
SCL	=	Serial Clock
SDA	=	Serial Data
Rp	=	Pull-up Resistance

EMBEDDED MULTI SENSORS FOR HYDROLOGICAL MONITORING SYSTEM

INTRODUCTION

Thailand is the agricultural country which is cultivated area more than 50%. Nowadays, 80% of water in this river is used in agriculture. So flood disaster is significant and growing problems for agriculturalist around Chao Phraya River. To reduce these problems, we would like to know the volume of the river for management. Therefore, improving the measurement system is one of the most important things to obtain data of river.

Recently, flow velocity of river measures in one depth (Lee *et al*., 2008) (Chayon *et al.*, 2007) and used to calculate the volume of the river. But flow velocity is varying in different depths. Thus, the information of flow velocity in any depths is more useful to analysis and management (Hughes *et al.*, 2006) (Freiberger and Sarvestani, 2007). Then we will improve measurement system by applying low cost flow velocity sensors and low cost flow direction sensor to measure and record them at various depth of river. After that we evaluate and send the data to station for the flow rate calculation.

In this research, the low cost flow velocity sensor and the low cost flow direction sensor are developed. We separate our work into two parts: design part and practical field part. In design and develop parts, the low cost flow velocity sensor and the low cost flow direction sensor is developed. The flow velocity sensor is designed by using a propeller and a Hall Effect sensor. The rotation of propeller is designed to provide a pulse when it passes Hall Effect sensor. The frequency of the pulses is proportional to the velocity of water in the river. The circuits that used in this sensor are differential amplifier and Schmitt Trigger. The low cost flow direction sensor is implemented by using a compass sensor, magnet, vane and controller. The position of vane that has attached by magnet is used to detect the flow direction of the water. It

uses converting signed two's complement to decimal and trigonometric relations to translate the data from compass sensor to an angle of the position of vane. In practical field part, the low cost flow velocity sensor and the low cost flow direction sensor are used to collect the data nearly the Mobile Telemetering for Flood Warning in Chao Phraya basin station. Then, we use the result of design and testing parts to compute the flow velocity and the flow direction of the water and compare with the data from the station.

OBJECTIVES

1. To design and develop the low cost flow velocity sensor for hydrological monitoring system.

2. To design and develop the low cost flow direction sensor for hydrological monitoring system

LITERATURE REVIEW

The idea of hydrological monitoring system with sensor networks is not new, Lee *et al*. (2008) described the real time flood monitoring with wireless sensor networks. In sensor node, they had a level sensor, a flow sensor and a rainfall sensor to collect data. Then they send that data via wireless network to a base station. In the base station, they used Ethernet port for connected to internet. Data from each river is stored in the database designed to distinguish the measured data by rivers and sensor nodes. The GUI-based web service providing 3D model, data graph, and other representation materials for better readability for users, and SMS are provided by using received data in real-time.

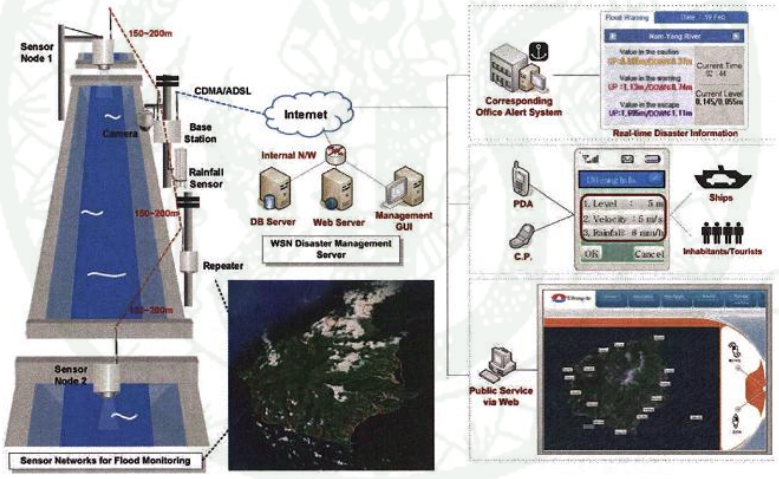


Figure 1 Real time flood monitoring with wireless sensor networks overview.

Chayon *et al*.(2007) described sensor network with Bluetooth. The sensor node consists of Bluetooth devices, water level sensor and pressure sensor to collect data and send it through relay node. Due to Bluetooth devices have a short range wireless then they use relay node to improve the area of network for base station to sensor node. In relay node there is only a microprocessor and a Bluetooth device to communicate with others. So there are another node which are only used for transmit the data to the next node. The Gate Way Node has the data from the sensors and transmits them to the base station. The Gate Way node is very important for the whole

section of the network. If this Gate Way node will fail the whole part of the network will fail. To solve this problem, they use the Double Gate Way Node.

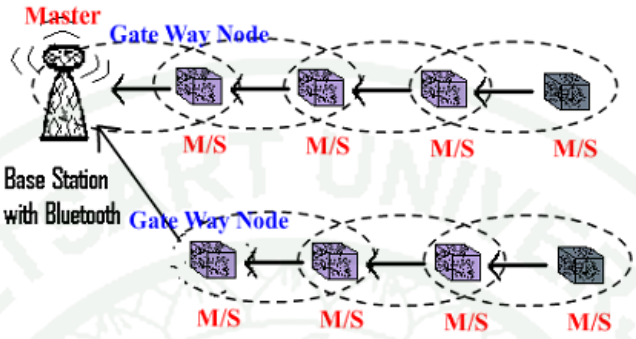


Figure 2 Sensor network with double Gate Way nodes.

Freiberger and Sarvestani (2007) researched in measure soil properties at various depths. Each sensor took measurements based on a schedule, or when prompted by an event. Events could be a measurement value exceeding a threshold, or simply at the user's request. The measurements must be stored and transmitted back over the existing GSM cellular infrastructure.

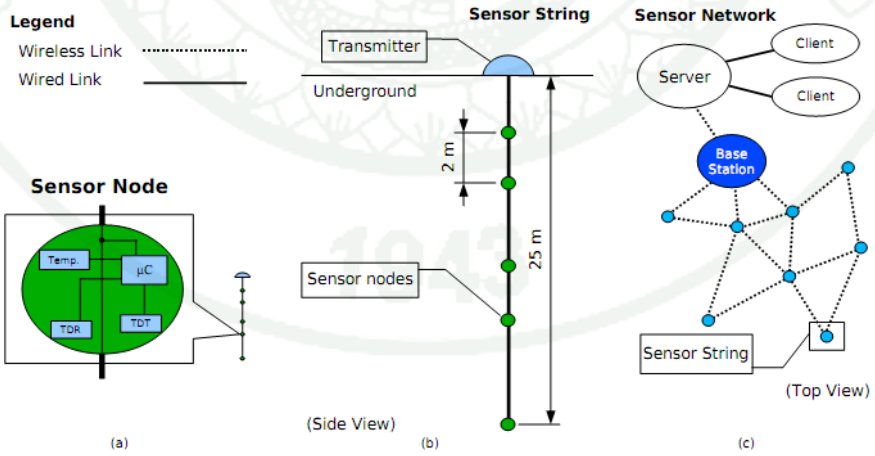


Figure 3 Proposed sensor network and its components.

However, many researchers have attempted to improve accuracy, resolution, scalability and power consumption. But to improve those things, we need to create low cost measurement systems, along with the ability to leave the equipment in the

field for extended periods of time, make it possible to increase the spatial resolution, or alternatively, cover a larger area. So, the main sensor that needs to develop is the flow velocity sensor and the direction sensor.

MATERIALS AND METHODS

Materials

1. Computer
2. CircuitMaker 2000
3. Microsoft Excel software
4. Microsoft Visual C++ 2008 Express Edition software
5. Altium Designer Winter 09
6. MSP430F169 microcontroller
7. IAR Embedded Workbench for MSP430
8. ET-MSP430 FET Debugger (ETT)
9. RS232 Cable
10. Hall Effect sensor (A1302)
11. Compass sensor (HMC5843)
12. Propeller

Methods

In this research, we divided into two parts: design part and practical field part. First, we will introduce the main components. Then, the low cost flow velocity sensor and the low cost flow direction sensor are developed at the design part. After that, they will test for finding the characteristic before using in the practical field. In the practical field part, those sensors will test in the real conditions around Mobile Telemetering for Flood Warning in Chao Phraya basin (C35).

1. Hardware description

In this experiment, we separate the explanation of hardware description into two parts. First, the main hardware on low cost flow velocity sensor. Second, the main hardware on low cost flow direction sensor.

1.1 Low cost flow velocity sensor

The low cost flow velocity sensor is designed by using a propeller and a Hall Effect sensor. The propeller as shown in Figure 4 is 3 inch propeller which is selling in the electronics market.

Figure 4 3 inch propeller

The Hall Effect sensor that use in this research is Continuous-Time Ratiometric Linear Hall Effect Sensor (A1302) as shown in Figure 5 from Allegro MicroSystems. It is optimized to accurately provide a voltage output that is proportional to an applied magnetic field. This device has a quiescent output voltage that is 50% of the supply voltage (4.5 to 6.0 V operations).

Figure 5 Hall Effect sensor (A1302)

1.2 low cost flow direction sensor

The low cost flow direction sensor is implemented by using a compass sensor, magnet, vane and controller. The compass sensor is 3-Axis Digital Compass IC (HMC5843) as shown in Figure 6 from Honeywell that provides advantages over other magnetic sensor technologies. The device features 3-Axis Magnetoresistive sensors and ASIC in a Single Package, Low Voltage Operations (2.5 to 3.3V) and I^2C Digital Interface. The HMC5843 uses a simple protocol with the interface protocol defined by the I^2C bus specification. The data rate is at the standard-mode 100kbps or 400kbps rates as defined in the I^2C bus specifications. The bus bit format is an 8-bit Data/Address send and a 1-bit acknowledge bit. The format of the data bytes (payload) shall be case sensitive ASCII characters or binary data to the HMC5843 slave, and binary data returned. Negative binary values will be in two's complement form. The default (factory) HMC5843 7-bit slave address is 0x3C for write operations, or 0x3D for read operations. The HMC5843 Serial Clock (SCL) and Serial Data (SDA) lines have optional internal pull-up resistors, but require resistive pull-ups (Rp) between the master device (usually a host microprocessor) and the HMC5843. Pull-up resistance values of about 10k ohms are recommended with a nominal 1.8-volt digital supply voltage. Other values may be used as defined in the I^2C bus sspecifications or with the internal 50k ohm pull-up resistors that can be tied to digital supply voltage.

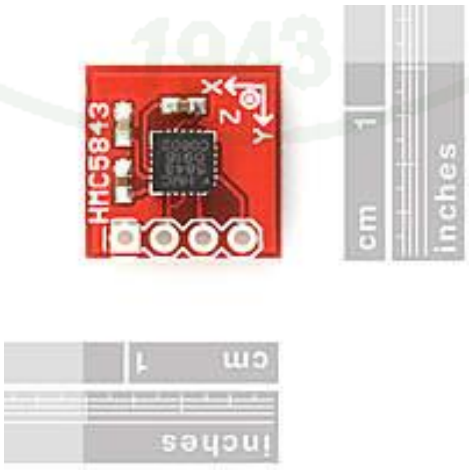


Figure 6 3-Axis Digital Compass IC (HMC5843)

The compass sensor architecture, combined with five modes which can be used to minimize the total power consumption in applications. The data from each axis is 16-bit value in 2's complement form, whose range is 0xF800 to 0x07FF. The recommended application circuit for this experiment is shown in Figure 7.

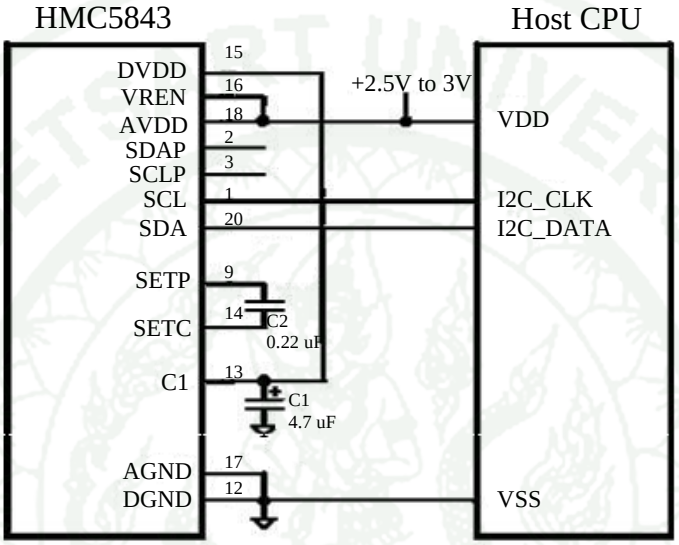


Figure 7 Application circuit of HMC5843

The microcontroller that use in this research is a MSP430F169 as shown in Figure 8. This microcontroller is a mixed-signal microcontroller family from Texas Instruments. Built around a 16-bit CPU, the MSP430 is designed for low cost, and specifically, low power consumption embedded applications. The microcontroller has six different low-power modes, which can disable unneeded clocks and CPU to minimize the total power consumption. The device features a powerful 16-bit RISC CPU, 16-bit registers, and constant generators that attribute to maximum code efficiency. The digitally controlled oscillator (DCO) allows wake-up from low-power modes to active mode in less than 6μs. MSP430F169 is microcontroller configurations with two built-in 16-bit timers, a fast 12-bit A/D converter, dual 12-bit D/A converter, two universal serial synchronous/asynchronous communication interfaces (USART), I2C, DMA, and 48 I/O pins.

Figure 8 MSP430F169

2. Design part

In this part, the general design and testing of low cost flow velocity sensor and low cost flow direction sensor is described. The proposed system shows in Figure 9. It has two microcontrollers. The main function of the main microcontroller is to communicate to computer via USB port and receive the data from the sensors.

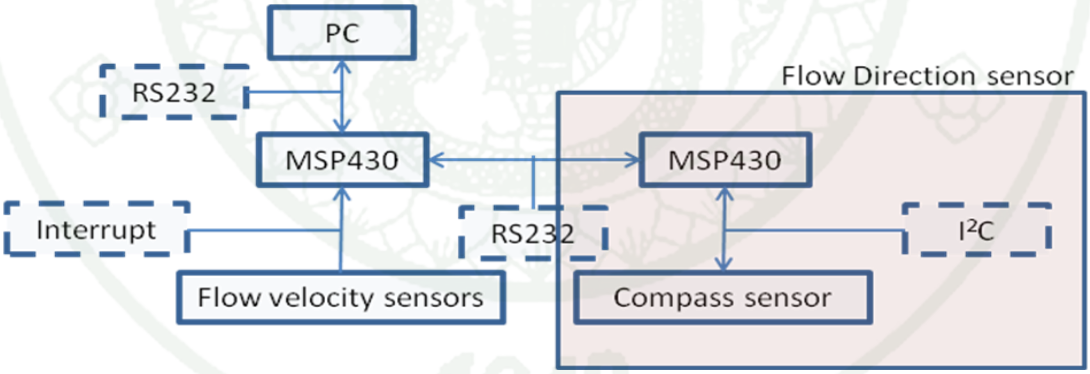


Figure 9 A block diagram of our proposed system.

2.1 low cost flow velocity sensor

The low cost flow velocity sensor, as shown in Figure 10, is designed by using a propeller and a Hall Effect sensor. The rotation of propeller is designed to provide a pulse when it passes Hall Effect sensor. The frequency of the pulses is proportional to the velocity of water in the river. The cost of our flow velocity sensor is about 500 Baht.

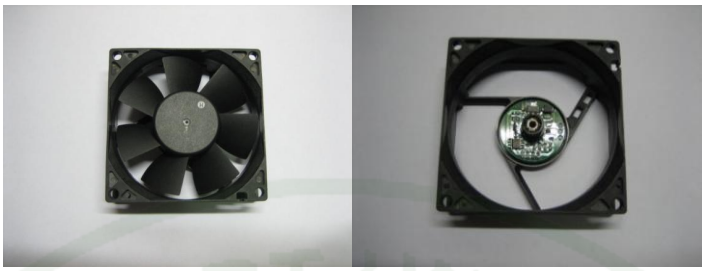


Figure 10 Flow velocity sensor

The circuits that used in flow velocity sensor are differential amplifier and Schmitt Trigger as shown in Figure 11. The supply voltage is 5 V. So, the quiescent output voltage of A1302 is about 2.5 V at none magnetic field. Then we need to cut off the offset by using differential amplifier and Schmitt Trigger to reduce the noise. The components that use in this sensor is shown in Table 1.

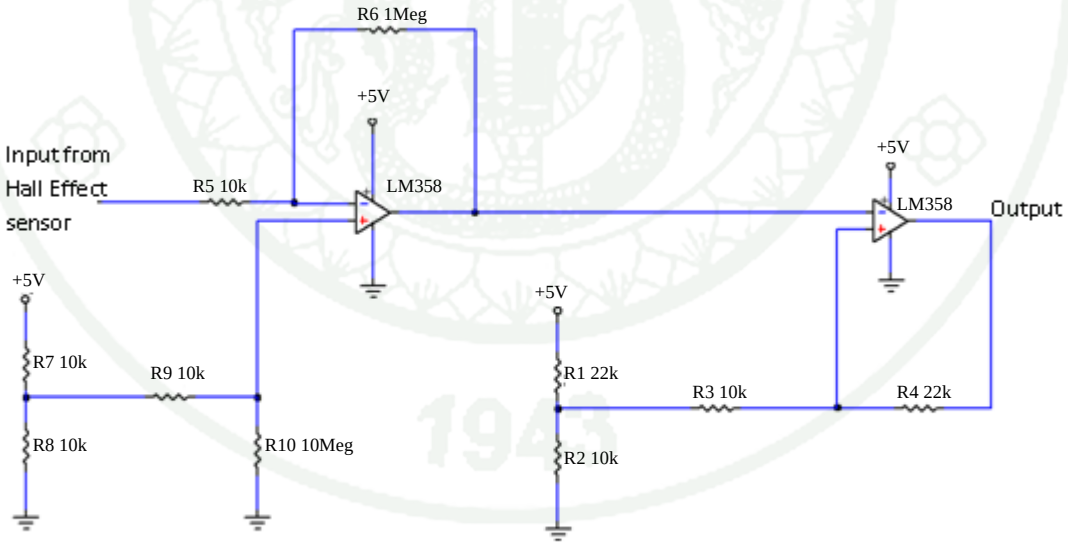


Figure 11 The schematic of low cost flow velocity sensor

Table 1 Bill of Materials for low cost flow velocity sensor circuit

Designator	Comment	Description	Quantity
C1	C 1uF	Capacitor SMD 1uF 16V	2
D1	MBRS120T3	Schottky Diode	1
R1, R3, R4, R5, R8, R9	R 10 K	Resistor SMD 1/10W 5%	6
R7, R10	R 22 K	Resistor SMD 1/10W 5%	2
R2, R6	R 1 M	Resistor SMD	2
U1	LM1117IMPX-5.0	LDO regulator ; Voltage 5	1
U2	A1302	Hall Effect Sensor	1
U3	LM358DR	Op-Amp SMD	1

In the testing of low cost flow velocity sensor, as shown in Figure 12, there are two barrels and pipe connected to the upper barrel. The flow velocity sensor was installed into the pipe. Then water was drained from the upper barrel to the lower barrel and measured the time and volume of the water. Therefore flow rate is obtained, and the velocity is calculated with this equation

$$V = Q/A \quad (1)$$

where, V is flow velocity of the water (m/s)

Q is flow rate of the water (m^3/s)

A is cross sectional area of the pipe (m^2)

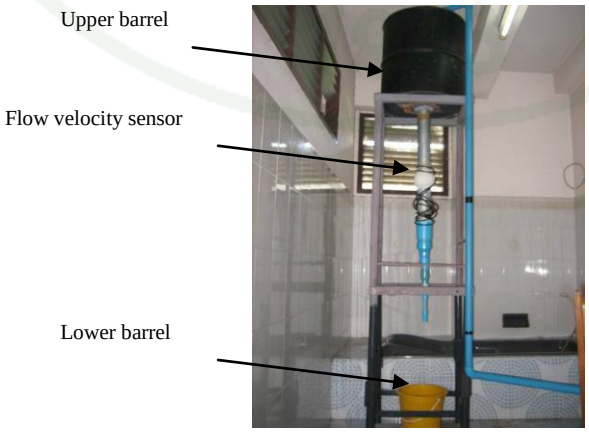


Figure 12 Over all of testing system of flow velocity sensor

The flow velocity sensor generates voltage pulses when the water passes. The relationship between number of pulse and the flow velocity is then found out by using trend analysis. The flow velocity sensors connect to interrupt channels of main microcontroller. Figure 13 shows the front-end window of the developed software using Visual C++ builder program. The application software has an auto sampling to get data in the period of time and save those data into a file. The obtained data consist of date, time and number of pulse from the flow velocity sensors. The flow chart of the system shows in the Figure 14, the pulses from the flow velocity sensors are sent to main microcontroller for counting. The PC then requests the data from main microcontroller. After that, counter is reset and main microcontroller then become waiting status for interrupt.

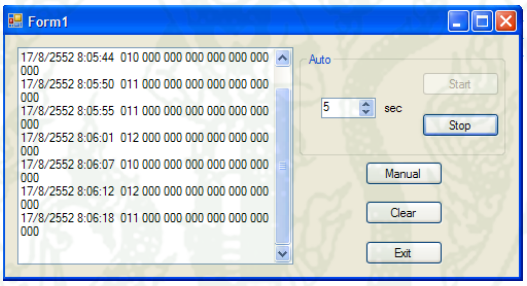


Figure 13 Front-end window of the data logger

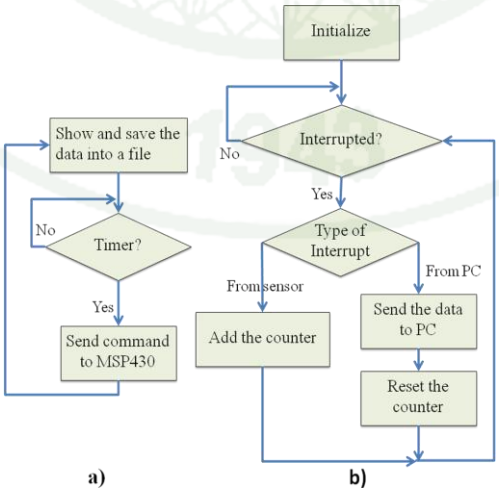


Figure 14 Flow chart of the system
a) Flow chart of programming in the PC
b) Flow chart of programming in main microcontroller

2.2 low cost flow direction sensor

The low cost flow direction sensor, as shown in Figure 15, is implemented by using a HMC5843, magnet, vane and controller (MSP430). The PCB inside box as shown in Figure 16, has the main components such as microcontroller (MSP430), compass sensor (HMC5843), True RS-232 transceivers (MAX3232) and low-dropout linear regulator (LM1117). The MSP430 is used to collect the data from HMC5843 when the microcontroller receives the signal from main microcontroller. The communication between main microcontroller and the microcontroller uses RS232 channel. The power supply is used in MSP430 and HMC5843 is 3.3 V. The components are used in this sensor is shown in Table 2. The cost of this sensor is about 3000 Baht.

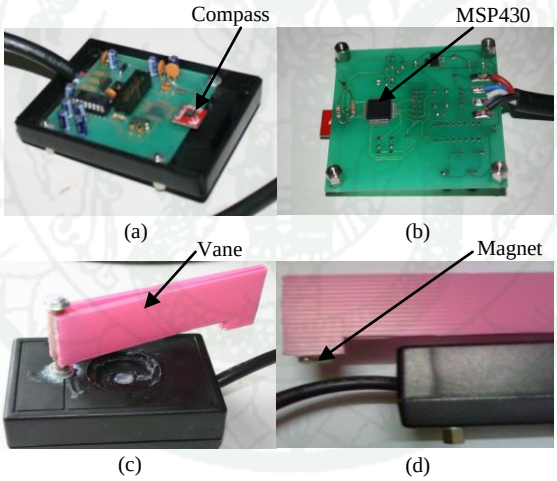


Figure 15 Flow direction sensor
(a) Inside of flow direction sensor
(b) Bottom view of PCB
(c) Top view of flow direction sensor
(d) Side view of flow direction sensor

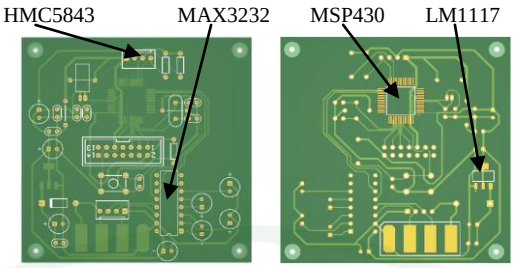


Figure 16 The PCB of flow direction sensor
(a) Top view
(b) Bottom view

Table 2 Bill of Materials for low cost flow direction sensor circuit

Designator	Comment	Description	Quantity
C1, C2, C3, C4, C6	Electrolytic Capacitor	0.1uF	5
C10	Electrolytic Capacitor	10uF	1
C15, C16	Electrolytic Capacitor	47uF	2
C5	Ceramic Capacitor	10 nF	1
C7, C8, C11, C12	Ceramic Capacitor	10 pF	4
C9, C13, C14	Ceramic Capacitor	0.1uF	3
Crystal1	XTAL(HC49S-7.3728M-LF)	7.3728MHz	1
Crystal2	XTAL (DT-26-LF)	32.768KHz	1
D1	Diode 1N4001	1 Amp General Purpose Rectifier	1
H1	Connector	Header 1x4 , 100 mil	1
H2	J-tag	Header 2x7 , 100 mil	1
H3	Compass Sensor	Header 1x4 , 100 mil	1
MSP1	MSP430f169	MSP430f169	1
R1	Resistor	Resistor 47k	1
R2, R3	Resistor	Resistor 10k	2
SW1	Reset (TC-0103-X-ROHS)	Tact Switch	1
U1	MAX3232CPE+	+-15KV ESD-PROTECTED, +5V RS-232 TRANSCEIVERS	1
U2	LM1117MPX-3.3	LDO regulator ; Voltage 3.3	1

The XYZ-axis data from the HMC5843, which is signed two's complement representation from 0xF800 to 0x07FF, can be collected by the application processor, and software routine written to interpret the data as required for heading output. The heading is computed by using these following trigonometric relations.

Where X > 0 and Y > 0

$$\text{Angle} = \text{arctangent}(Y/X) \quad (2)$$

Where X < 0 and Y > 0

$$\text{Angle} = 180 + \text{arctangent}(Y/X) \quad (3)$$

Where X < 0 and Y < 0

$$\text{Angle} = 270 - \text{arctangent}(Y/X) \quad (4)$$

Where X > 0 and Y < 0

$$\text{Angle} = 360 + \text{arctangent}(Y/X) \quad (5)$$

Where X = 0 and Y > 0

$$\text{Angle} = 90 \quad (6)$$

Where X = 0 and Y < 0

$$\text{Angle} = 270 \quad (7)$$

Where X > 0 and Y = 0

$$\text{Angle} = 0 \quad (8)$$

Where X < 0 and Y = 0

$$\text{Angle} = 180 \quad (9)$$

HMC5843 is connected through I2C of microcontroller. The software is used to interpret the data is developed by Visual C++ builder program as shown in Figure 17. The flowchart of the software, is shown in Figure 18, has an auto sampling to get data in the period of time and record those data into a file. The obtained data consist of date, time and raw data from sensor. Then, the program converts the raw data to the position of vane in XY-axis by converting signed two's complement to decimal. Afterwards, the angle is calculated by using the trigonometric relations of equation (2)-(9).

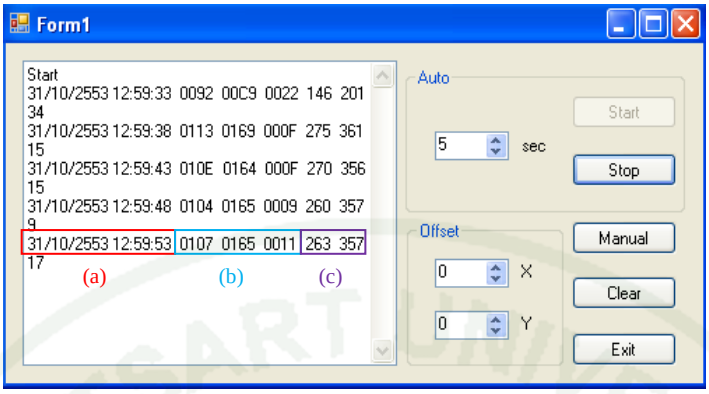


Figure 17 Front-end window of the data logger
(a) Date and time (b) Raw data (c) Position of vane in XY-axis

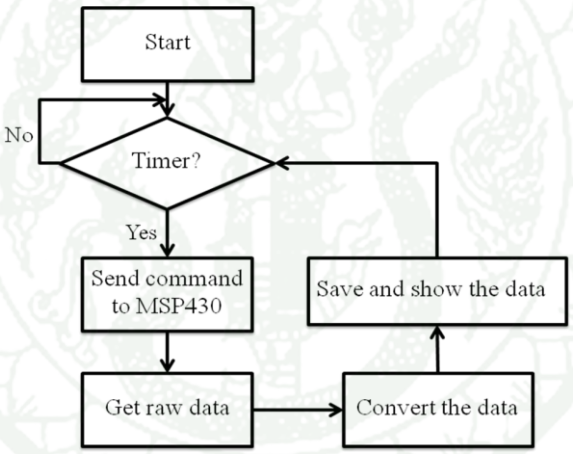


Figure 18 A flowchart of the main program

The vane that has attached by magnet is used to collect data for this experiment. Here, the vane is rotated to 4 reference positions as shown in Figure 19. The first position is on 0° of X-axis, the second position is on 180° of X-axis, the third position is on 90° of X-axis and the fourth position is on 270° of X-axis. For each reference positions, the data is collected 10 times. After that, the average data for each reference positions are calculated. The minimum and maximum values of XY-axis are found. Then, determine the mean values as offsets for each axis. By subtracting these offsets to the average 4 references positions, the calibrated compass heading is resolved.

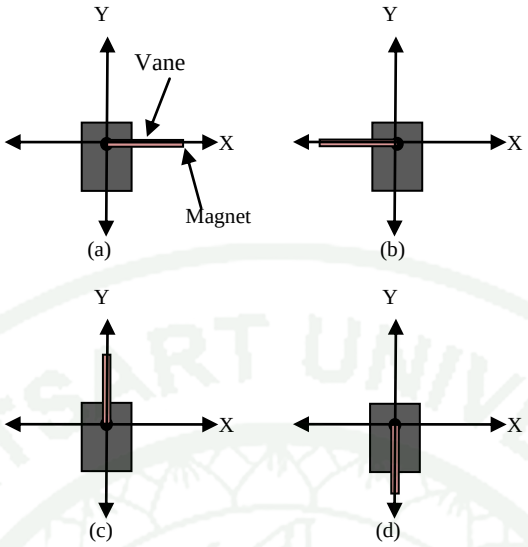


Figure 19 Reference positions
(a) First position (b) Second position
(c) Third position (d) Fourth position

Figure 20 shows the average 4 reference positions offset being translated back to the correct axis for compass heading computation. Xmax represents the average first reference position, Xmin represents the average second reference position, Ymax represents the average third reference position and Ymin represents the average fourth reference position. The dash-dot line represents the trace of magnet that is obtained from HMC5843. The solid line represents the trace of vane that is subtracted by offsets.

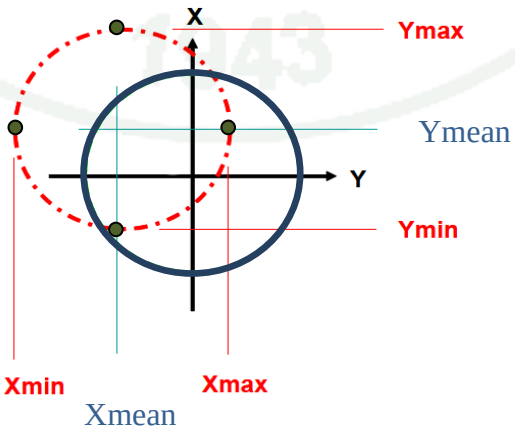


Figure 20 Offset translation of HMC5843

3. Practical field part

The practical field that use to test the sensors is Mobile Telemetering for Flood Warning in Chao Phraya basin station (C35) as shown in Figure 21.

Figure 21 Mobile Telemetering for Flood Warning in Chao Phraya basin station(C35)
Amphoe Bang Ban Phra Nakhon Si Ayutthaya
Latitude: 14.36907 Longitude: 100.52751

Distance between the low cost flow direction sensor and the ground is 0.50 m. And the distance between sensor to sensor is 0.50 m.

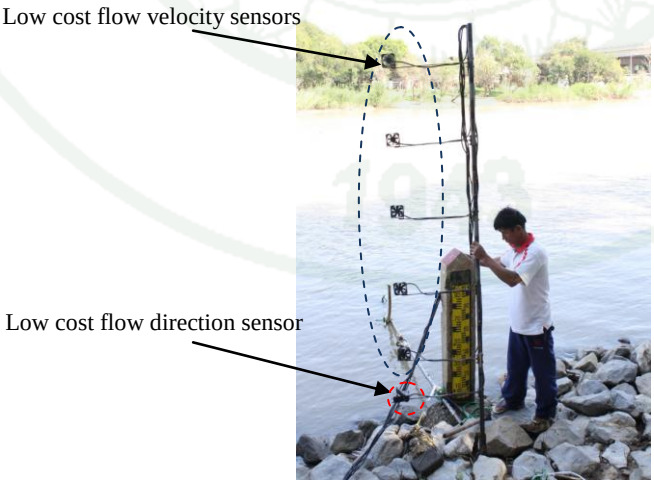


Figure 22 The setting sensors

RESULTS AND DISCUSSION

Our experiments are designed based on our system development. Two phases of experiments are performed. The first phase is to validate our low cost flow velocity sensor and low cost flow direction sensor. In this phase, our sensors are tested in the setting conditions for finding characteristic. The second phase experiment is to validate our sensors node. Our sensors are experimented under real condition.

1. Sensors

1.1 low cost flow velocity sensor

The goal of testing the low cost flow velocity sensor is to find the characteristic before using in the practical field. The sensor is tested in the setting conditions. First, we collect the data in six points from the low velocity to high velocity by changing the size of pipe. For each point, the data will collect in 30 times and average them as shown in Table 3. The flow velocity can obtained by draining the water from the upper barrel to the lower barrel and measured the time and volume of the water. We denote that the capacity of lower barrel is 0.01 m^3 so the flow rate is obtained by equation (10). And the flow velocity is calculated by equation (1). So, the average data for each point will pot in to the graph.

$$Q = 0.01/T \tag{10}$$

where, Q is flow rate (m^3/s) T is time (s)

Table 3 The average data from six points.

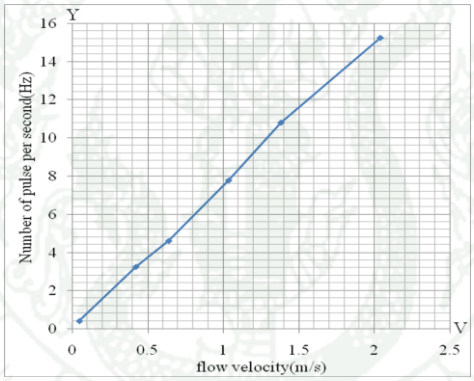
Time	velocity(m/s)	Number of pulse per second(Hz)
1	0.046670804	0.400983639
2	0.420690649	3.23879846
3	0.637617256	4.59463978
4	1.033772724	7.765845719
5	1.380021131	10.78627316
6	2.037147617	15.22725145

Result of testing the low cost flow velocity sensor shows in the Figure 23. It shows relationship between number of pulse per second and the flow velocity. The minimum of flow velocity that the low cost flow velocity sensor can be detected is 0.04 m/s. And approximate velocity is calculated by using trend analysis. After calculating, polynomial of degree 2 is used in the trend line because it has the least root mean square error as shown in Table 4. The coefficients in this line is given by

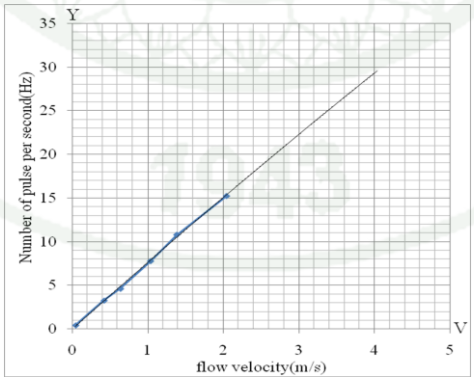
$$Y = -0.104V^2 + 7.761V - 0.050 \quad (11)$$

where, V is flow velocity (m/s)

Y is number of the pulse per second (Hz)



(a)



(b)

Figure 23 Relationship between number of pulse per second and the velocity
(a) The result form testing
(b) Add the trend line

Table 4 Root mean square of each trend line

Type of trend line	Root mean square error
Polynomial of degree 2	0.183782166
Linearity	0.188551778

The experimental results showed that the low cost flow velocity sensor which used a propeller and a Hall Effect sensor has a low root mean square error so it can measure the flow velocity in general flow-velocity condition. Then the low cost flow velocity sensor can be used in the sensor node to improve measurement system by obtaining the flow velocity in various depth.

1.2 low cost flow direction sensor

The goal of this experiment is to calibrate the low cost flow velocity sensor and find the characteristic before using in the hydrological monitoring station around riverside.

Table 5 to Table 8 show the data that collect 10 times for each reference positions. After that, the average data for each reference positions are calculated as shown in Table 9.

Table 5 The data from first reference position

X axis	Y axis
145	436
150	423
147	432
142	429
148	426
151	426
148	436
147	429
141	426
144	425

Table 6 The data from second reference position

X axis	Y axis
-334	434
-323	436
-331	431
-332	430
-331	436
-329	435
-320	438
-323	436
-322	434
-329	423

Table 7 The data from third reference position

X axis	Y axis
-82	679
-75	680
-81	673
-82	676
-76	674
-84	680
-85	678
-78	676
-78	680
-82	678

Table 8 The data form fourth reference position

X axis	Y axis
-91	214
-90	213
-90	208
-89	211
-95	215
-93	214
-84	218
-94	212
-94	221
-93	216

Table 9 Average data of 4 reference positions

Reference position	The average on X-axis ($\bar{X}$)	The average on Y-axis ($\bar{Y}$)
First	146.3	428.8
Second	-327.4	433.3
Third	-80.3	677.4
Fourth	-91.3	214.2

The graph of average 4 reference positions is shown in Figure 24. The error of displacement from the correct axis can be found on the graph. Therefore, the offset is required to adjust the error of HMC5843.

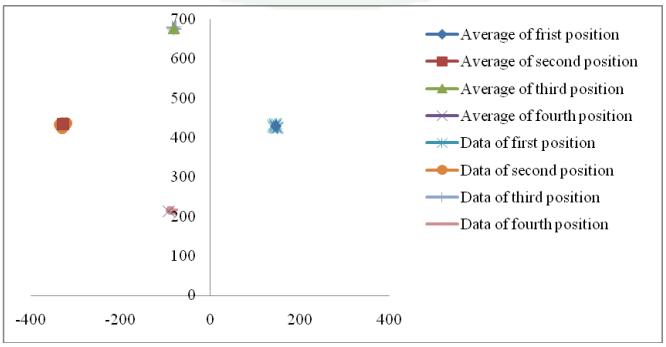


Figure 24 Average position of 4 reference points

Figure 25 shows the plot of 4 reference positions after subtracting the offsets. For X-axis, the offset is determined by

$$\text{Offset X} = (\bar{X}_{First} + \bar{X}_{Second})/2 \quad (12)$$

where, $\bar{X}_{First}$ is the average of first reference position.

$\bar{X}_{Second}$ is the average of second reference position.

For Y-axis, the offset is determined by

$$\text{Offset Y} = (\bar{Y}_{Third} + \bar{Y}_{Fourth})/2 \quad (13)$$

where, $\bar{Y}_{Third}$ is the average of third reference position.

$\bar{Y}_{Fourth}$ is the average of fourth reference position.

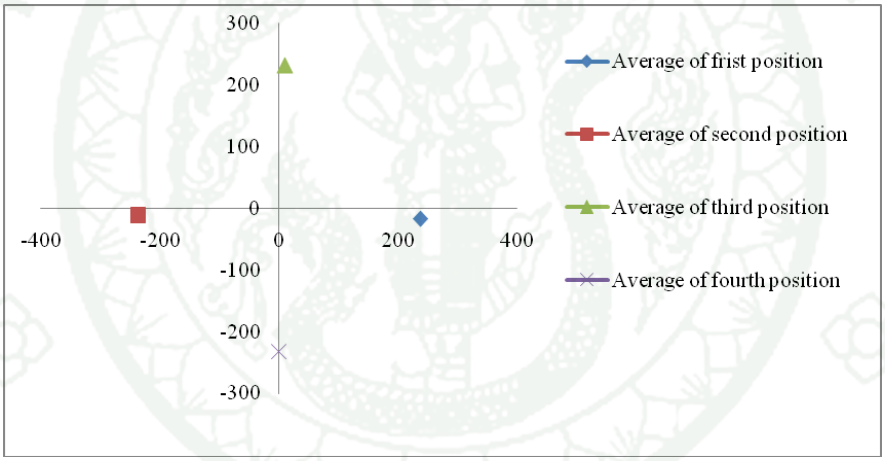


Figure 25 Position of 4 reference points after subtracting offset

Table 10 shows the data which is collected from first reference position. The angle is calculated by using the trigonometric relations of equation (2)-(9). Here, the maximum ADC quantization error of low cost flow direction sensor is 5.41°.

Table 10 The data of first reference position

Data from first reference position				Angle (Degree)
Raw data		Subtracting offsets		
X	Y	X	Y	
145	436	235.55	-9.8	357.62
150	423	240.55	-22.8	354.59
147	432	237.55	-13.8	356.68
142	429	232.55	-16.8	355.87
148	426	238.55	-19.8	355.26
151	426	241.55	-19.8	355.31
148	436	238.55	-9.8	357.65
147	429	237.55	-16.8	355.95
141	426	231.55	-19.8	355.11
144	425	234.55	-20.8	354.93

Table 11 shows the data that is collected from first reference position with pitch. The error of pitch at 10° is 4.72° and the error of pitch at 20° is 8.94°. So, the error is increased due to the angle changed. The second, the third and the fourth reference position has the result as the first reference position.

Table 11 The data of first reference position with pitch

Average data from first reference position					Angle (Degree)
	Raw data		Subtracting offsets		
Pitch	X	Y	X	Y	
0°	146.30	428.80	236.85	-17.00	355.89
10°	143.90	448.30	234.45	2.50	0.61
20°	144.00	465.60	234.55	19.80	4.83
-10°	145.20	412.20	235.75	-33.60	351.89
-20°	143.50	387.60	234.05	-58.20	346.04

Our experimental results showed that the compass sensor (HMC5843) need to calibration before using. Furthermore, the pitch of the sensor has effect to the accuracy. Then, the sensor node should install in horizontal to prevent the error.

2. Practical Field

This section presents results of testing our proposed systems on a practical field. Our practical field is nearly Mobile Telemetering for Flood Warning in Chao Phraya basin (C35) shown in Figure 26. This part is the hardest part in our thesis. Because we need to calibrate the flow direction sensor in the water before collecting the data on the field. After calibration, we collect the data every 15 minutes in one day and compare the results with the Mobile Telemetering for Flood Warning.

Figure 26 Our practical field

Due to the power supply of the sensors and the main controller is different; we have to re-do the experiment setup before running on the field. The connection ports board as shown in Figure 27 is used to connect the sensors to the main controller. The circuit that shows in Figure 28 is used to convert the output voltage of low cost flow velocity sensor before connecting to main controller.

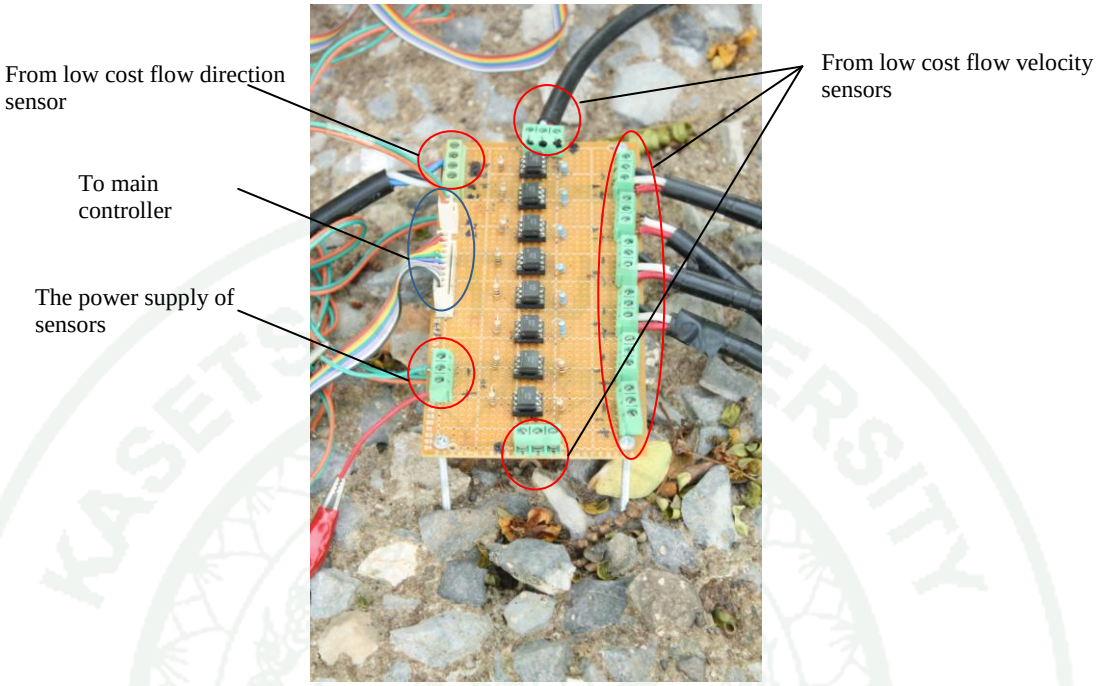


Figure 27 Connection ports board

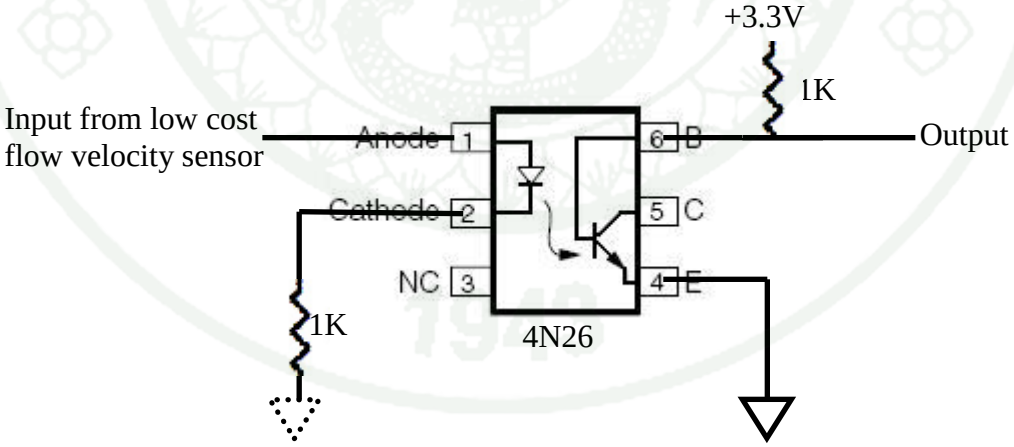


Figure 28 The converting circuit

This experiment is conducted in order to test the accuracy of the proposed system. During process, our systems collected the flow velocity at three different depths and a flow direction. First, the calibration of flow direction sensor is determined by using the raw data of 4 positions as shown in Table 12. So, the offset

can be found by using equation (12)-(13). The offset of X is 236.37 and the offset of Y is -393.67. Then, the data will collect in one day. The collected data have combined with date, time, flow velocity at three depths and the flow direction as shown in Figure 29.

Table 12 The raw data of reference positions

The raw data of reference positions							
First		Second		Third		Fourth	
X	Y	X	Y	X	Y	X	Y
422	-342	-53	-335	230	-104	228	-566
427	-363	-46	-360	240	-123	229	-593
436	-380	-32	-381	242	-141	249	-621
433	-430	-27	-380	228	-139	238	-617
433	-377	-47	-370	227	-140	227	-592
432	-442	-43	-402	226	-177	220	-642
433	-439	-35	-444	222	-216	245	-665
439	-426	-33	-426	230	-187	237	-650
442	-409	-39	-409	235	-169	235	-639
439	-414	-31	-414	241	-169	244	-649
437	-419	-35	-419	240	-192	230	-650
433	-378	-32	-390	237	-168	237	-627
432	-375	-36	-375	239	-147	249	-613
442	-394	-42	-421	235	-186	231	-661
429	-383	-51	-391	245	-154	245	-613

	A	B	C	J	K	L	M	N
1	Date&Time	X	Y	V2	V1	V0	x	y
2	28/12/2553 12:23	01A6	FEAA	75	68	44	422	-342
3	28/12/2553 12:23	01AB	FE95	19	19	18	427	-363
4	28/12/2553 12:23	01B4	FE84	20	20	19	436	-380
5	28/12/2553 12:23	01B1	FE52	18	17	17	433	-430
6	28/12/2553 12:23	01B1	FE87	21	19	13	433	-377
7	28/12/2553 12:23	01B0	FE46	18	15	11	432	-442
8	28/12/2553 12:23	01B1	FE49	18	12	11	433	-439
9	28/12/2553 12:23	01B7	FE56	20	16	8	439	-426
10	28/12/2553 12:23	01BA	FE67	20	11	7	442	-409
11	28/12/2553 12:24	01B7	FE62	20	12	9	439	-414
12	28/12/2553 12:24	01B5	FE5D	19	16	11	437	-419
13	28/12/2553 12:24	01B1	FE86	22	17	11	433	-378
14	28/12/2553 12:24	01B0	FE89	21	17	17	432	-375

Figure 29 The data logger

Figure 30 shows the data that collect from Chao Phraya basin (C35) at 28 December to 29 December 2010 without change some parameter in trend line of our flow velocity sensor. Vmean is the data from Mobile Telemetering for Flood Warning. V2, V1, V0 are the data from our flow velocity sensors. The level of station's sensor is the same level as V1. The Error of sensor V0 and V1 come from the leaving in the river such as garbage, water hyacinth.

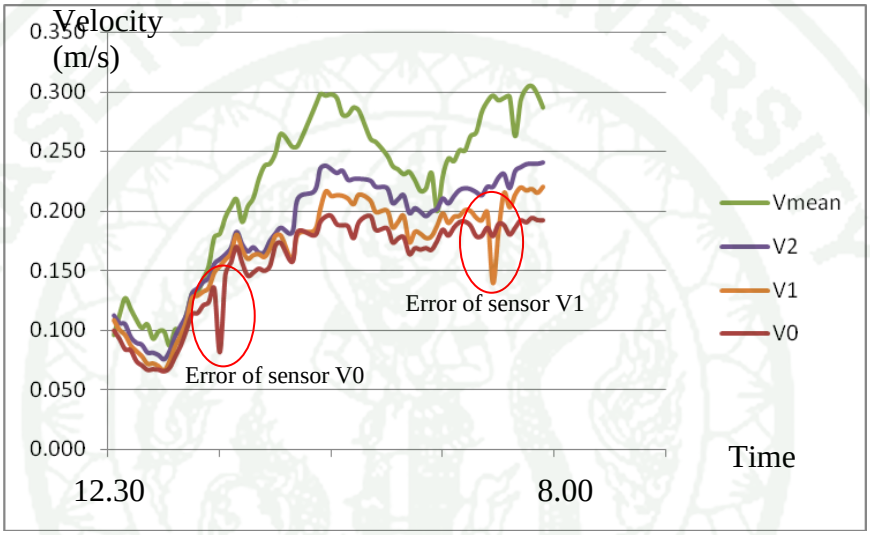
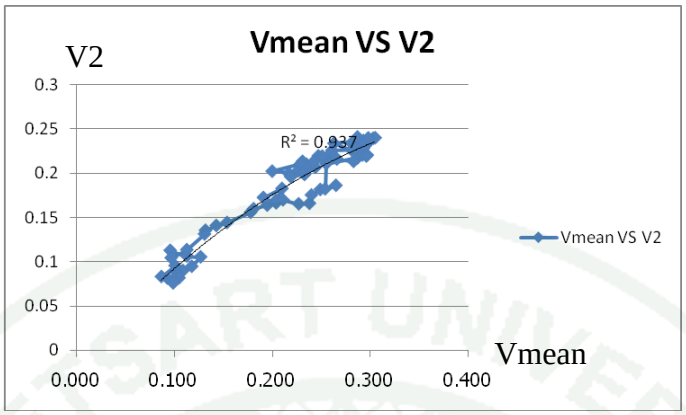
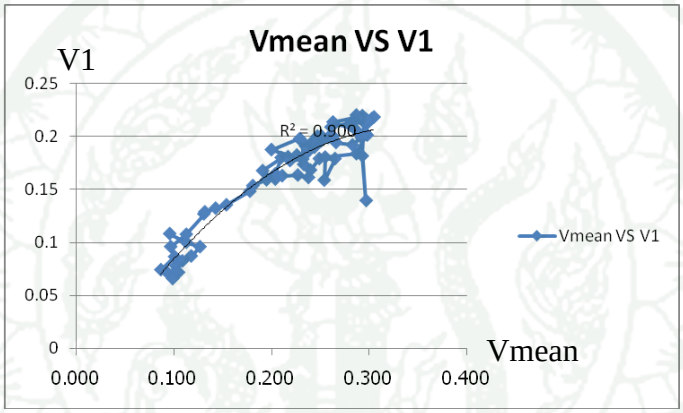


Figure 30 The data of flow velocities without change some parameter

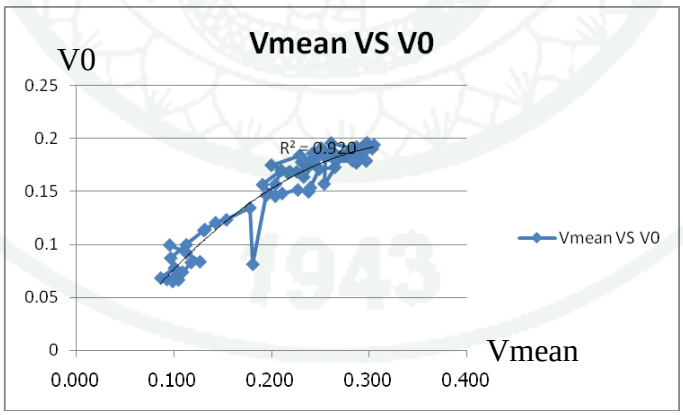
The accuracy of our flow velocity sensors compare with the station's sensor are shows in Figure 31.the accuracy of them have more than 90%. Figure 32 shows the data of flow velocities with change some parameter in trend line of our flow velocity sensor.



a)



b)



c)

Figure 31 The accuracy from low cost flow velocity sensors compare with station's sensor
a) Vmean VS V2
b) Vmean VS V1
c) Vmean VS V0

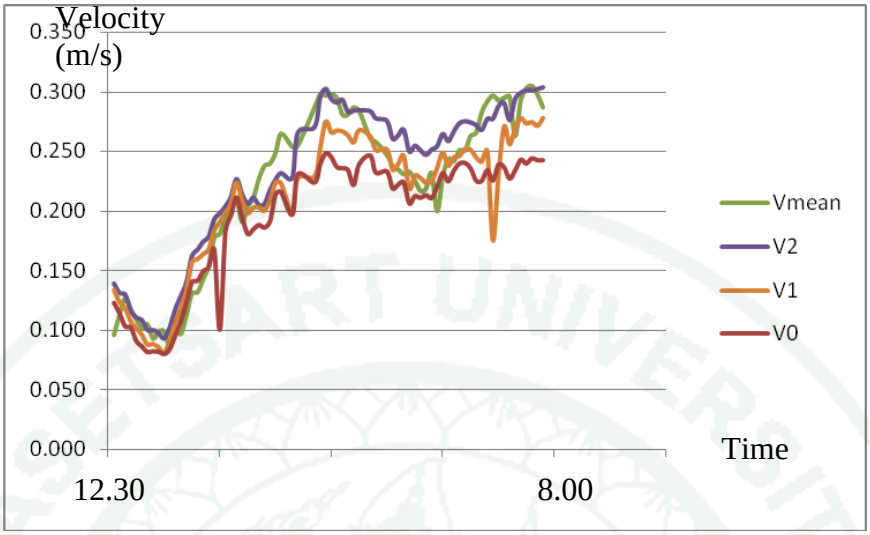


Figure 32 The data of flow velocities with change some parameter

CONCLUSION AND RECOMMENDATION

Conclusion

In this thesis, the low cost flow velocity sensor and the low cost flow direction sensor for hydrological monitoring system is developed. Those sensors used to measure the flow velocity in various depths and the flow direction of the river.

Our experimental results show that the low cost flow velocity sensors have an acceptable measure range and a low root mean square error. So, they can measure the flow velocity in general flow-velocity condition. Furthermore, the measure in various depths can indicate the error of the sensors because of leaving in the river. In addition, they need to change some parameter to improve the accuracy. And the low cost flow direction sensor has an acceptable accuracy but it need to calibration before using. Furthermore, the pitch and roll of the flow direction sensor has effect to the accuracy. About the cost, the existing station is 400,000 Baht but our system is about 10,000 Baht. So, our system can spread out for covering a larger area.

Therefore, the low cost sensor nodes which consist of low cost flow velocity sensors and low cost flow direction sensor have efficiency for using in the hydrological monitoring station.

Recommendation

1. The low cost flow velocity sensor has a problem with the leaving from the river. Therefore the appropriate shield is required to avoid the problem. Setup the net forward the sensor can reduce the problem but it will effect to the accuracy. So, we need to modify the parameter of low cost flow velocity sensor after setup the net.

2. The calibration of low cost flow direction sensor in practical field is hard to handle. So, the self calibration programming is required. About the pitch and roll, it needs another sensor to detect such as accelerometer.

LITERATURE CITED

Danny Hughes, Phil Greenwood, Geoff Coulson and Gordon Blair. 2006. GridStix supporting flood prediction using embedded hardware and next generation grid middleware. **Proceedings of the 2006 International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM'06)**. 0-7695-2593-8/06 2006.

David Harris, David Money Harris, Sarah L. Harris., 2007. "Digital Design and Computer Architecture"

Jong-uk Lee, Jae-Eon Kim, Daeyoung Kim and Poh Kit Chong. 2008. RFMS: Real-time Flood Monitoring System with Wireless Sensor Networks. **National Science & Technology Development Agency.** 1-4244-2575-4/08 2008. 527-528

Md. Hasibur Rashid Chayon, Tawhidur Rahman, Md. Forhad Rabbi and Md. Masum. 2007. Automated River Monitoring System for Bangladesh using Wireless Sensor Network. **IEEE**. 1-4244-1550-0/07 2007.

Mikolaj Sawicki, "Myths about Gravity and Tides", "The Physics Teacher" 37, October 1999. 438 - 441.

Mobile Telemetering for Flood Warning in Chao Phraya basin. http://122.155.12.58/index.php?page=3&bid=10&sid=C35

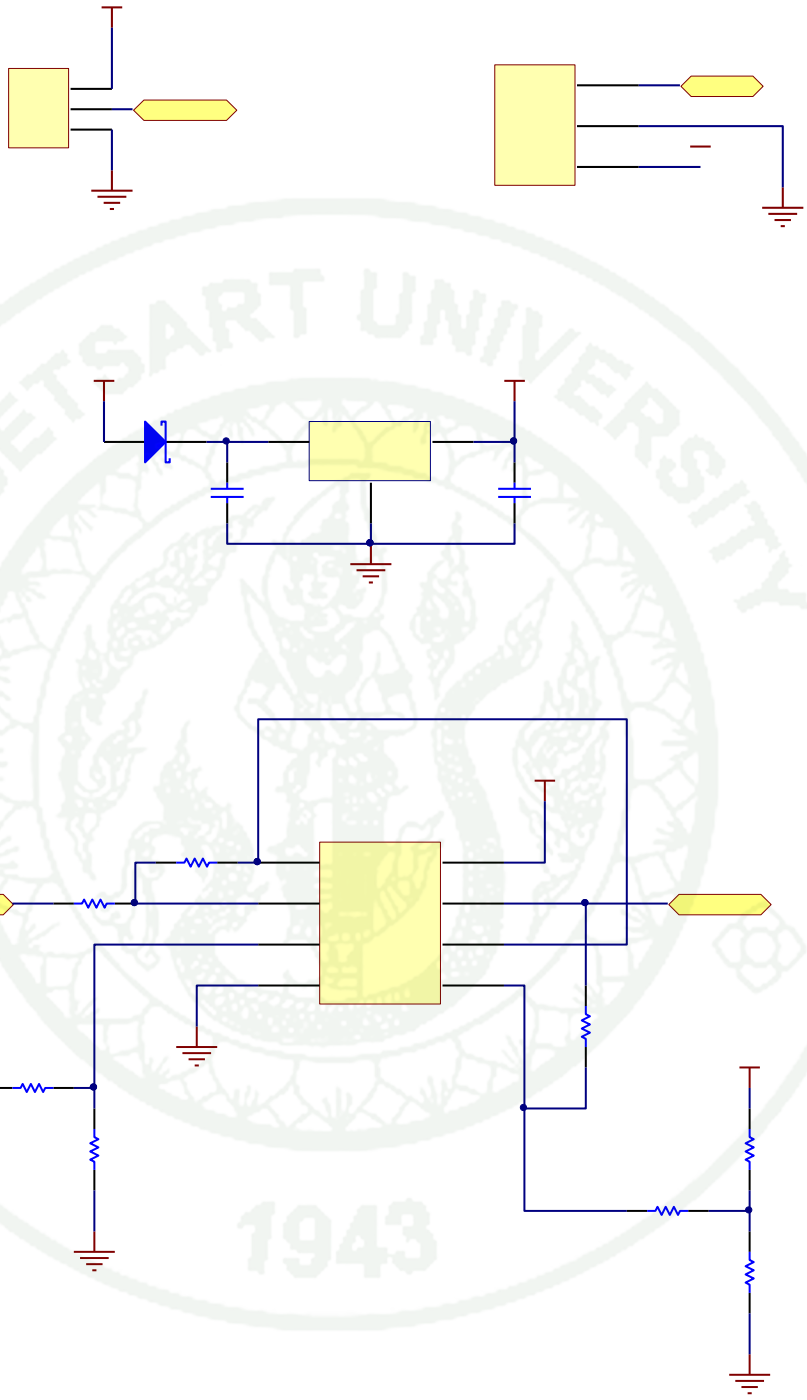
The information of HMC5843. http://www.ssec.honeywell.com/

The Texas Instruments. **MSP430 family of ultralow power microcontrollers.** http://focus.ti.com/docs/prod/folders/print/msp430f169.html.

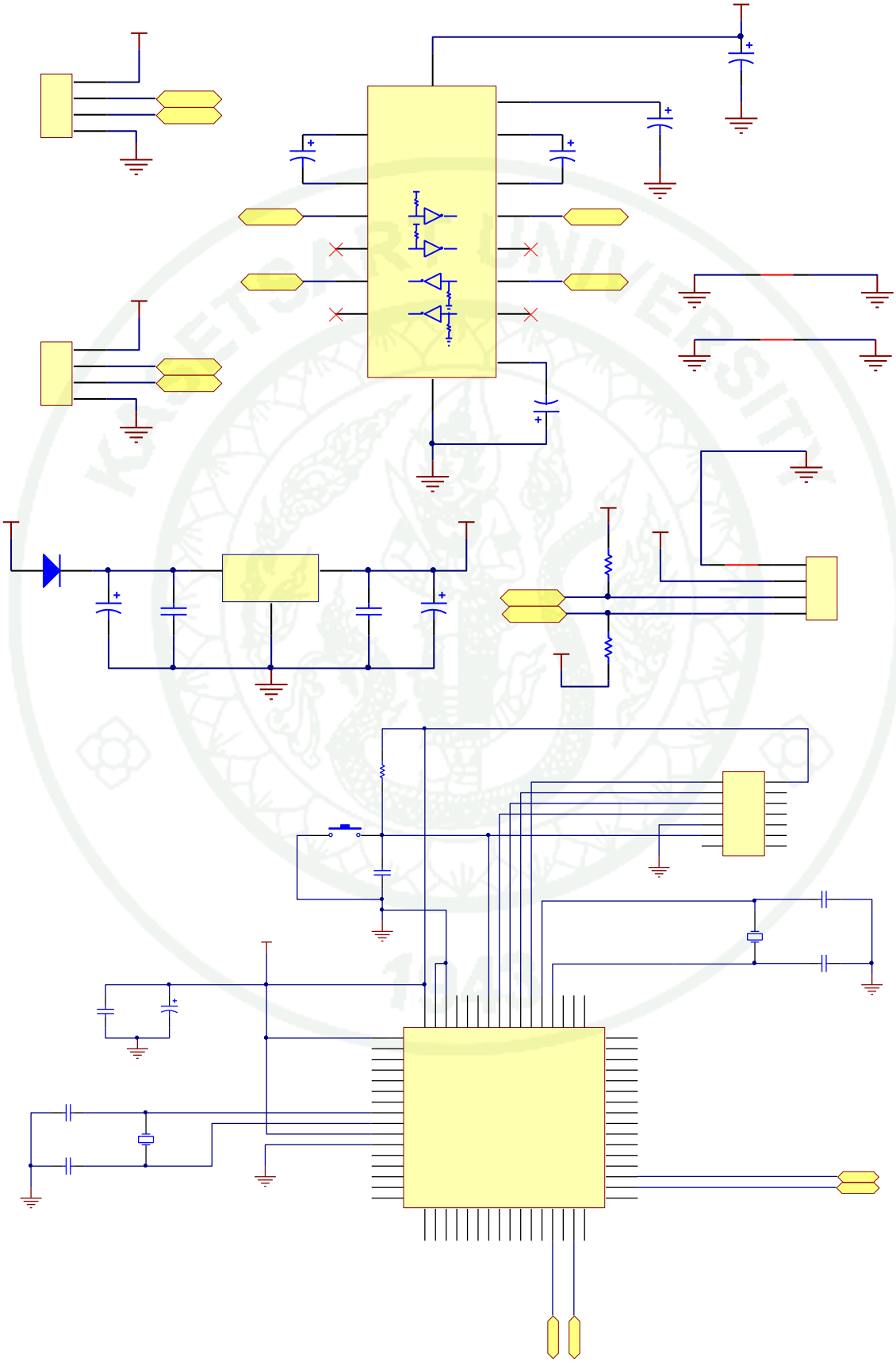
Thomas V. Freiberger and Sahra Sedigh Sarvestani. 2007. Hydrological Monitoring with Hybrid Sensor Networks. **International Conference on Sensor Technologies and Applications**. 0-7695-2988-7/07 2007. 484-489

APPENDICES

Appendix A
The circuit of low cost flow velocity sensor



Appendix B
The circuit of low cost flow velocity sensor



Appendix C
Source code of main microcontroller

```
#include "msp430x16x.h"

void InitData();
void InitPort1();
void InitUART1();
void InitUART0();

int SEN_V[8];                   // for count pulse of velocity sensor
int i,m;
unsigned char XH_DATA1, XH_DATA0,
              XL_DATA1, XL_DATA0,
              YH_DATA1, YH_DATA0,
              YL_DATA1, YL_DATA0,
              ZH_DATA1, ZH_DATA0,
              ZL_DATA1, ZL_DATA0;

void main()
{
 WDTCTL = WDTPW + WDTHOLD  ; // Stop watchdog timer

 InitData()              ; // Initialized data
 InitPort1()             ; //Initialized P1 interrupt
 InitUART1()               ; //Initialized USART1
 InitUART0()             ; //Initialized USART0

 _BIS_SR(LPM1_bits+GIE)    ; // LPM3, enable all interrupt
 while(1)
 {
 }
}
```

```
//======================== SUB
==================================


//----------- InitData ------------
void InitData()
{
  for(i=0;i<8;i++) SEN_V[i] = 0;
  XH_DATA1 = 0x00; XH_DATA0 = 0x00;
  XL_DATA1 = 0x00; XL_DATA0 = 0x00;
  YH_DATA1 = 0x00; YH_DATA0 = 0x00;
  YL_DATA1 = 0x00; YL_DATA0 = 0x00;
  ZH_DATA1 = 0x00; ZH_DATA0 = 0x00;
  ZL_DATA1 = 0x00; ZL_DATA0 = 0x00;
}

//----------- InitPort1 Interrupt ------------
void InitPort1()
{
  P1IE   = 0xFF             ; // P1 interrupt enabled
  P1IES  = 0x00             ; // P1 Hi/lo edge
  P1IFG &= ~0xFF            ; // P1 IFG cleared
}

//----------- InitUART1 Interrupt ------------
void InitUART1()
{
  P3SEL |= 0xC0             ; // P3.6,7 = USART1 TXD/RXD
  ME2 |= UTXE1 + URXE1      ; // Enable USART1 TXD/RXD
  UCTL1 |= CHAR             ; // 8-bit character
  UTCTL1 |= SSEL0           ; // UCLK = ACLK
  UBR01 = 0x03              ; // 32k/9600 - 3.41
```

```
  UBR11 = 0x00              ;
  UMCTL1 = 0x4A             ; // Modulation
  UCTL1 &= ~SWRST           ; // Initialize USART state machine
  IE2 |= URXIE1             ; // Enable USART1 RX interrupt
}

//----------- InitUART0 Interrupt ------------
void InitUART0()
{
  P3SEL |= 0x30             ; // P3.4,5 = USART0 TXD/RXD
  ME1 |= UTXE0 + URXE0      ; // Enable USART0 TXD/RXD
  UCTL0 |= CHAR             ; // 8-bit character
  UTCTL0 |= SSEL0           ; // UCLK = ACLK
  UBR00 = 0x03              ; // 32k/9600 - 3.41
  UBR10 = 0x00              ;
  UMCTL0 = 0x4A             ; // Modulation
  UCTL0 &= ~SWRST           ; // Initialize USART state machine
}

// Port 1 interrupt service routine
#pragma vector=PORT1_VECTOR
__interrupt void Port_1int()
{
  int buffer                ;
  buffer = P1IFG            ;
  for(int m =0;m<8;m++)
  {
    if((buffer&0x01)==0x01)
      SEN_V[m] += 1         ;
    buffer >>=1             ; // Shift right 1 bit
  }
  P1IFG &= ~0xFF            ; // P1 IFG cleared
```

```
}

//USART1 interrupt service routine
#pragma vector=USART1RX_VECTOR
__interrupt void usart1_rx (void)
{
  while (!(IFG1 & UTXIFG0));             // USART0 TX buffer ready?
  TXBUF0 = 0x20;                    // Sent Space
  while (!(IFG1 & URXIFG0));             // USART0 RX buffer ready?
  XH_DATA1 = RXBUF0;
  while (!(IFG1 & URXIFG0));             // USART0 RX buffer ready?
  XH_DATA0 = RXBUF0;
  while (!(IFG1 & URXIFG0));             // USART0 RX buffer ready?
  XL_DATA1 = RXBUF0;
  while (!(IFG1 & URXIFG0));             // USART0 RX buffer ready?
  XL_DATA0 = RXBUF0;
  while (!(IFG1 & URXIFG0));             // USART0 RX buffer ready?
  YH_DATA1 = RXBUF0;
  while (!(IFG1 & URXIFG0));             // USART0 RX buffer ready?
  YH_DATA0 = RXBUF0;
  while (!(IFG1 & URXIFG0));             // USART0 RX buffer ready?
  YL_DATA1 = RXBUF0;
  while (!(IFG1 & URXIFG0));             // USART0 RX buffer ready?
  YL_DATA0 = RXBUF0;
  while (!(IFG1 & URXIFG0));             // USART0 RX buffer ready?
  ZH_DATA1 = RXBUF0;
  while (!(IFG1 & URXIFG0));             // USART0 RX buffer ready?
  ZH_DATA0 = RXBUF0;
  while (!(IFG1 & URXIFG0));             // USART0 RX buffer ready?
  ZL_DATA1 = RXBUF0;
  while (!(IFG1 & URXIFG0));             // USART0 RX buffer ready?
  ZL_DATA0 = RXBUF0;
```

```
while (!(IFG2 & UTXIFG1));              // USART1 TX buffer ready?
TXBUF1 = XH_DATA1;
while (!(IFG2 & UTXIFG1));              // USART1 TX buffer ready?
TXBUF1 = XH_DATA0;
while (!(IFG2 & UTXIFG1));              // USART1 TX buffer ready?
TXBUF1 = XL_DATA1;
while (!(IFG2 & UTXIFG1));              // USART1 TX buffer ready?
TXBUF1 = XL_DATA0;
while (!(IFG2 & UTXIFG1));              // USART1 TX buffer ready?
TXBUF1 = 0x20;                  // Sent Space
while (!(IFG2 & UTXIFG1));              // USART1 TX buffer ready?
TXBUF1 = YH_DATA1;
while (!(IFG2 & UTXIFG1));              // USART1 TX buffer ready?
TXBUF1 = YH_DATA0;
while (!(IFG2 & UTXIFG1));              // USART1 TX buffer ready?
TXBUF1 = YL_DATA1;
while (!(IFG2 & UTXIFG1));              // USART1 TX buffer ready?
TXBUF1 = YL_DATA0;
while (!(IFG2 & UTXIFG1));              // USART1 TX buffer ready?
TXBUF1 = 0x20;                  // Sent Space
while (!(IFG2 & UTXIFG1));              // USART1 TX buffer ready?
TXBUF1 = ZH_DATA1;
while (!(IFG2 & UTXIFG1));              // USART1 TX buffer ready?
TXBUF1 = ZH_DATA0;
while (!(IFG2 & UTXIFG1));              // USART1 TX buffer ready?
TXBUF1 = ZL_DATA1;
while (!(IFG2 & UTXIFG1));              // USART1 TX buffer ready?
TXBUF1 = ZL_DATA0;
while (!(IFG2 & UTXIFG1));              // USART1 TX buffer ready?
TXBUF1 = 0x20;                  // Sent Space

int av7=0x30,bv7=0x30,cv7=0x30,dv7=0x30;
```

```
av7=av7+(SEN_V[7] % 10);
bv7=bv7+((SEN_V[7]/10)%10);
cv7=cv7+((SEN_V[7]/100)%10);
dv7=dv7+((SEN_V[7]/1000)%10);
while (!(IFG2 & UTXIFG1));              // USART1 TX buffer ready?
TXBUF1 = dv7;                           //1000
while (!(IFG2 & UTXIFG1));              // USART1 TX buffer ready?
TXBUF1 = cv7;                           //100
while (!(IFG2 & UTXIFG1));              // USART1 TX buffer ready?
TXBUF1 = bv7;                           //10
while (!(IFG2 & UTXIFG1));              // USART1 TX buffer ready?
TXBUF1 = av7;                           //1
while (!(IFG2 & UTXIFG1));              // USART1 TX buffer ready?
TXBUF1 = 0x20;                          // Sent Space

int av6=0x30,bv6=0x30,cv6=0x30,dv6=0x30;
av6=av6+(SEN_V[6] % 10);
bv6=bv6+((SEN_V[6]/10)%10);
cv6=cv6+((SEN_V[6]/100)%10);
dv6=dv6+((SEN_V[6]/1000)%10);
while (!(IFG2 & UTXIFG1));              // USART1 TX buffer ready?
TXBUF1 = dv6;                           //1000
while (!(IFG2 & UTXIFG1));              // USART1 TX buffer ready?
TXBUF1 = cv6;                           //100
while (!(IFG2 & UTXIFG1));              // USART1 TX buffer ready?
TXBUF1 = bv6;                           //10
while (!(IFG2 & UTXIFG1));              // USART1 TX buffer ready?
TXBUF1 = av6;                           //1
while (!(IFG2 & UTXIFG1));              // USART1 TX buffer ready?
TXBUF1 = 0x20;                          // Sent Space

int av5=0x30,bv5=0x30,cv5=0x30,dv5=0x30;
```

```
av5=av5+(SEN_V[5] % 10);
bv5=bv5+((SEN_V[5]/10)%10);
cv5=cv5+((SEN_V[5]/100)%10);
dv5=dv5+((SEN_V[5]/1000)%10);
while (!(IFG2 & UTXIFG1));              // USART1 TX buffer ready?
TXBUF1 = dv5;                           //1000
while (!(IFG2 & UTXIFG1));              // USART1 TX buffer ready?
TXBUF1 = cv5;                           //100
while (!(IFG2 & UTXIFG1));              // USART1 TX buffer ready?
TXBUF1 = bv5;                           //10
while (!(IFG2 & UTXIFG1));              // USART1 TX buffer ready?
TXBUF1 = av5;                           //1
while (!(IFG2 & UTXIFG1));              // USART1 TX buffer ready?
TXBUF1 = 0x20;                          // Sent Space

int av4=0x30,bv4=0x30,cv4=0x30,dv4=0x30;
av4=av4+(SEN_V[4] % 10);
bv4=bv4+((SEN_V[4]/10)%10);
cv4=cv4+((SEN_V[4]/100)%10);
dv4=dv4+((SEN_V[4]/1000)%10);
while (!(IFG2 & UTXIFG1));              // USART1 TX buffer ready?
TXBUF1 = dv4;                           //1000
while (!(IFG2 & UTXIFG1));              // USART1 TX buffer ready?
TXBUF1 = cv4;                           //100
while (!(IFG2 & UTXIFG1));              // USART1 TX buffer ready?
TXBUF1 = bv4;                           //10
while (!(IFG2 & UTXIFG1));              // USART1 TX buffer ready?
TXBUF1 = av4;                           //1
while (!(IFG2 & UTXIFG1));              // USART1 TX buffer ready?
TXBUF1 = 0x20;                          // Sent Space

int av3=0x30,bv3=0x30,cv3=0x30,dv3=0x30;
```

```
av3=av3+(SEN_V[3] % 10);
bv3=bv3+((SEN_V[3]/10)%10);
cv3=cv3+((SEN_V[3]/100)%10);
dv3=dv3+((SEN_V[3]/1000)%10);
while (!(IFG2 & UTXIFG1));              // USART1 TX buffer ready?
TXBUF1 = dv3;                           //1000
while (!(IFG2 & UTXIFG1));              // USART1 TX buffer ready?
TXBUF1 = cv3;                           //100
while (!(IFG2 & UTXIFG1));              // USART1 TX buffer ready?
TXBUF1 = bv3;                           //10
while (!(IFG2 & UTXIFG1));              // USART1 TX buffer ready?
TXBUF1 = av3;                           //1
while (!(IFG2 & UTXIFG1));              // USART1 TX buffer ready?
TXBUF1 = 0x20;                          // Sent Space

int av2=0x30,bv2=0x30,cv2=0x30,dv2=0x30;
av2=av2+(SEN_V[2] % 10);
bv2=bv2+((SEN_V[2]/10)%10);
cv2=cv2+((SEN_V[2]/100)%10);
dv2=dv2+((SEN_V[2]/1000)%10);
while (!(IFG2 & UTXIFG1));              // USART1 TX buffer ready?
TXBUF1 = dv2;                           //1000
while (!(IFG2 & UTXIFG1));              // USART1 TX buffer ready?
TXBUF1 = cv2;                           //100
while (!(IFG2 & UTXIFG1));              // USART1 TX buffer ready?
TXBUF1 = bv2;                           //10
while (!(IFG2 & UTXIFG1));              // USART1 TX buffer ready?
TXBUF1 = av2;                           //1
while (!(IFG2 & UTXIFG1));              // USART1 TX buffer ready?
TXBUF1 = 0x20;                          // Sent Space

int av1=0x30,bv1=0x30,cv1=0x30,dv1=0x30;
```

```
av1=av1+(SEN_V[1] % 10);
bv1=bv1+((SEN_V[1]/10)%10);
cv1=cv1+((SEN_V[1]/100)%10);
dv1=dv1+((SEN_V[1]/1000)%10);
while (!(IFG2 & UTXIFG1));              // USART1 TX buffer ready?
TXBUF1 = dv1;                           //1000
while (!(IFG2 & UTXIFG1));              // USART1 TX buffer ready?
TXBUF1 = cv1;                           //100
while (!(IFG2 & UTXIFG1));              // USART1 TX buffer ready?
TXBUF1 = bv1;                           //10
while (!(IFG2 & UTXIFG1));              // USART1 TX buffer ready?
TXBUF1 = av1;                           //1
while (!(IFG2 & UTXIFG1));              // USART1 TX buffer ready?
TXBUF1 = 0x20;                          // Sent Space

int av0=0x30,bv0=0x30,cv0=0x30,dv0=0x30;
av0=av0+(SEN_V[0] % 10);
bv0=bv0+((SEN_V[0]/10)%10);
cv0=cv0+((SEN_V[0]/100)%10);
dv0=dv0+((SEN_V[0]/1000)%10);
while (!(IFG2 & UTXIFG1));              // USART1 TX buffer ready?
TXBUF1 = dv0;                           //1000
while (!(IFG2 & UTXIFG1));              // USART1 TX buffer ready?
TXBUF1 = cv0;                           //100
while (!(IFG2 & UTXIFG1));              // USART1 TX buffer ready?
TXBUF1 = bv0;                           //10
while (!(IFG2 & UTXIFG1));              // USART1 TX buffer ready?
TXBUF1 = av0;                           //1
while (!(IFG2 & UTXIFG1));              // USART1 TX buffer ready?
TXBUF1 = 0x20;                          // Sent Space

while (!(IFG2 & UTXIFG1));              // USART1 TX buffer ready?
```

```
  TXBUF1 = 0x0A;                    // new line

  InitData();                       // Initialized data
}
```

Appendix D
Source code of microcontroller in low cost flow direction sensor

```
#include "msp430x16x.h"

void InitData();
void InitUART1();
void InitI2C();
void TX_byte();
void RX_byte();
unsigned char check(unsigned char);

int i=0x30,j,k;
unsigned char XH_DATA, XL_DATA, YH_DATA, YL_DATA, ZH_DATA,
        ZL_DATA,temp;

void main()
{
 WDTCTL = WDTPW + WDTHOLD  ; // Stop watchdog timer

 InitData()              ; // Initialized data
 InitUART1()             ; //Initialized USART1
 InitI2C()               ; //Initialized I2C

 _BIS_SR(LPM1_bits+GIE)    ; // LPM3, enable all interrupt
 while(1)
 {
 }
}

//======================== SUB
        ==================================

//----------- InitData ------------
void InitData()
```

```
{
  XH_DATA = 0x00;
  XL_DATA = 0x00;
  YH_DATA = 0x00;
  YL_DATA = 0x00;
  ZH_DATA = 0x00;
  ZL_DATA = 0x00;
}

//---------- InitUART1 Interrupt ----------
void InitUART1()
{
  P3SEL |= 0xC0           ; // P3.6,7 = USART1 TXD/RXD
  ME2 |= UTXE1 + URXE1    ; // Enable USART1 TXD/RXD
  UCTL1 |= CHAR           ; // 8-bit character
  UTCTL1 |= SSEL0         ; // UCLK = ACLK
  UBR01 = 0x03            ; // 32k/9600 - 3.41
  UBR11 = 0x00            ;
  UMCTL1 = 0x4A           ; // Modulation
  UCTL1 &= ~SWRST         ; // Initialize USART state machine
  IE2 |= URXIE1           ; // Enable USART1 RX interrupt
}

//---------- InitI2C ----------
void InitI2C()
{
  P3SEL  |= 0x0A          ; // Set I2C pins P3.1=SDA,P3.3=SCL
  U0CTL  |= I2C + SYNC    ; // Switch USART0 to I2C mode
  U0CTL  &= ~I2CEN        ; // Disable I2C module :I2CEN=0
  I2CTCTL = I2CSSEL_2     ; // Clock source SMCLK
  I2CSCLH = 0x03          ; // Shift CLK. High period of SCL = 5x(1.333us)
  I2CSCLL = 0x03          ; // Shift CLK. Low  period of SCL = 5x(1.333us)
```

```
  I2CSA   = 0x1E            ; // Slave address
  U0CTL |= I2CEN            ; // Enable I2C, 7 bit addr,
}

//------------ TX Data -----------

void TX_byte (void)
{
  j = 0;
  U0CTL |= MST;                    // Master mode
  I2CTCTL |= I2CSTT+I2CSTP+I2CTRX;     // Initiate transfer
  while ((I2CIFG & TXRDYIFG) == 0){j++; if(j>99) break;}; // Wait for transmitter
          to be ready
  I2CDRB = 0x02;
  j = 0;
  while ((I2CIFG & TXRDYIFG) == 0){j++; if(j>99) break;};    // Wait for
          transmitter to be ready
  I2CDRB = 0x00;
  j = 0;
  while((I2CTCTL & I2CSTP) == 0x02){j++; if(j>99) break;};   // Wait for Stop
          Condition
}

//------------ RX Data -----------

void RX_byte (void)
{
  k = 0;
  U0CTL |= MST;                    // Master mode
  I2CTCTL &= ~I2CTRX;
  I2CTCTL |= I2CSTT+I2CSTP;          // Initiate transfer
```

```
  while ((I2CIFG & RXRDYIFG) == 0){k++; if(k>99) break;};    // Wait for Receiver
        to be ready
  XH_DATA += I2CDRB;
  k = 0;
  while ((I2CIFG & RXRDYIFG) == 0){k++; if(k>99) break;};    // Wait for Receiver
        to be ready
  XL_DATA += I2CDRB;
  k = 0;
  while ((I2CIFG & RXRDYIFG) == 0){k++; if(k>99) break;};    // Wait for Receiver
        to be ready
  YH_DATA += I2CDRB;
  k = 0;
  while ((I2CIFG & RXRDYIFG) == 0){k++; if(k>99) break;};    // Wait for Receiver
        to be ready
  YL_DATA += I2CDRB;
  k = 0;
  while ((I2CIFG & RXRDYIFG) == 0){k++; if(k>99) break;};    // Wait for Receiver
        to be ready
  ZH_DATA += I2CDRB;
  k = 0;
  while ((I2CIFG & RXRDYIFG) == 0){k++; if(k>99) break;};    // Wait for Receiver
        to be ready
  ZL_DATA += I2CDRB;
  k = 0;
  while((I2CTCTL & I2CSTP) == 0x02){k++; if(k>99) break;};   // Wait for Stop
        Condition
}

//------------ check -----------
unsigned char check (unsigned char buff)
{
  int count = 0;
```

```
 if((buff & 0x01) == 1) count++;
 buff >>= 1;
 if((buff & 0x01) == 1) count= count + 2;
 buff >>= 1;
 if((buff & 0x01) == 1) count= count + 4;
 buff >>= 1;
 if((buff & 0x01) == 1) count= count + 8;
 if(count == 0) return (0x30);
 if(count == 1) return (0x31);
 if(count == 2) return (0x32);
 if(count == 3) return (0x33);
 if(count == 4) return (0x34);
 if(count == 5) return (0x35);
 if(count == 6) return (0x36);
 if(count == 7) return (0x37);
 if(count == 8) return (0x38);
 if(count == 9) return (0x39);
 if(count == 10) return (0x41);
 if(count == 11) return (0x42);
 if(count == 12) return (0x43);
 if(count == 13) return (0x44);
 if(count == 14) return (0x45);
 if(count == 15) return (0x46);
 return(0x5A);
}

//USART1 interrupt service routine
#pragma vector=USART1RX_VECTOR
__interrupt void usart1_rx (void)
{
//  while(1){
//    for(int dd=1; dd<10000;dd++){}
```

```
  I2CNDAT = 0x02            ; // Read 2 bytes
  TX_byte()                 ; // Go Sub-Program TX data
  I2CNDAT = 0x06            ; // Read 6 bytes
  RX_byte()                 ; // Go Sub-Program TR data
//  }

  temp = XH_DATA;
  temp >>= 4;
  while (!(IFG2 & UTXIFG1))   ; // Check USART1 TX buffer ready?
  TXBUF1 = check(temp)        ; // Sent data
  while (!(IFG2 & UTXIFG1))   ; // Check USART1 TX buffer ready?
  TXBUF1 = check(XH_DATA)     ; // Sent data
  temp = XL_DATA;
  temp >>= 4;
  while (!(IFG2 & UTXIFG1))   ; // Check USART1 TX buffer ready?
  TXBUF1 = check(temp)        ; // Sent data
  while (!(IFG2 & UTXIFG1))   ; // Check USART1 TX buffer ready?
  TXBUF1 = check(XL_DATA)     ; // Sent data

  temp = YH_DATA;
  temp >>= 4;
  while (!(IFG2 & UTXIFG1))   ;// Check USART1 TX buffer ready?
  TXBUF1 = check(temp)        ;// Sent data
  while (!(IFG2 & UTXIFG1))   ;// Check USART1 TX buffer ready?
  TXBUF1 = check(YH_DATA)     ;// Sent data
  temp = YL_DATA;
  temp >>= 4;
  while (!(IFG2 & UTXIFG1))   ;// Check USART1 TX buffer ready?
  TXBUF1 = check(temp)        ;// Sent data
  while (!(IFG2 & UTXIFG1))   ;// Check USART1 TX buffer ready?
  TXBUF1 = check(YL_DATA)     ;// Sent data
```

```
  temp = ZH_DATA;
  temp >>= 4;
  while (!(IFG2 & UTXIFG1))   ;// Check USART1 TX buffer ready?
  TXBUF1 = check(temp)        ;// Sent data
  while (!(IFG2 & UTXIFG1))   ;// Check USART1 TX buffer ready?
  TXBUF1 = check(ZH_DATA)     ;// Sent data
  temp = ZL_DATA;
  temp >>= 4;
  while (!(IFG2 & UTXIFG1))   ;//Check USART1 TX buffer ready?
  TXBUF1 = check(temp)        ;// Sent data
  while (!(IFG2 & UTXIFG1))   ;//Check USART1 TX buffer ready?
  TXBUF1 = check(ZL_DATA)     ;// Sent data

  InitData();                 // Initialized data
}
```

CIRRICULUM VITAE

NAME : Mr. Rattanasak Kasettham

BIRTH DATE : July 28, 1985

BIRTH PLACE : Nakhonsawan, Thailand

EDUCATION :

YEAR	**INSTITUTE**	**DEGREE/DIPLOMA**
2008	Kasetsart Univ.	B.Eng. (Electrical Engineering)
2011	Kasetsart Univ.	M.Eng. (Information and Communication Technology for Embedded Systems)

POSITION/TITLE : -

WORK PLACE : -

SCHOLARSHIP/AWARDS : TAIST ICTES Master Degree Scholarship

PUBLICATIONS ICTICTES2010 and ICTICTES2011