สัญญาณเตือนภัยน้ำท่วมผ่านทางโทรศัพท์มือถือ

THE FLOOD WARNING SYSTEM OVER MOBILEPHONE

นายทนากร เขื่อนสอน รหัส 51383805

นายปิยะพงศ์ ดวงสุภา รหัส 51384888

นายพงศธร โปธิตา รหัส 51384895

ห้องสมุดคณะวิศวกรรมศาสตร์
วันที่รับ 6 ส.ค. 2555
เลขทะเบียน 16069366
เลขเรียกหนังสือ ปร.
มหาวิทยาลัยนเรศวร ท 117ส 2555

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมไฟฟ้า ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

ปีการศึกษา 2555

ใบรับรองปริญญานิพนธ์

ชื่อหัวข้อโครงงาน สัญญาณเตือนภัยผ่านทางโทรศัพท์มือถือ

ผู้ดำเนินโครงงาน นายทนากร เขื่อนสอน รหัส 51383805

นายปิยะพงศ์ ดวงสุภา รหัส 51384888

นายพงศธร โปธิตา รหัส 51384895

ที่ปรึกษาโครงงาน ผศ.ดร. อัครพันธ์ วงศ์กังแห

สาขาวิชา วิศวกรรมไฟฟ้า

ภาควิชา วิศวกรรมไฟฟ้าและคอมพิวเตอร์

ปีการศึกษา 2555

.....

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร อนุมัติให้ปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมไฟฟ้า

คณะกรรมการสอบโครงงานวิศวกรรม

.....ที่ปรึกษาโครงงาน

(ผศ.ดร. อัครพันธ์ วงศ์กังแห)

.....กรรมการ

(ดร. พิสุทธิ์ อภิชยกุล)

.....กรรมการ

(ผศ.ดร. สมพร เรืองสินชัยวานิช)

ชื่อหัวข้อโครงงาน	สัญญาณเตือนภัยผ่านทางโทรศัพท์มือถือ	
ผู้ดำเนินโครงงาน	นายทนากร เชื่อนสอน	รหัส 51383805
	นายปิยะพงศ์ ดวงสุภา	รหัส 51384888
	นายพงศธร โปธิตา	รหัส 51384895
ที่ปรึกษาโครงงาน	ผศ.ดร. อัครพันธ์ วงศ์กังแห	
สาขาวิชา	วิศวกรรมไฟฟ้า	
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์	
ปีการศึกษา	2555	

.....

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้นำเสนอโครงงานเกี่ยวกับการออกแบบเครื่องสัญญาณเตือนภัยน้ำท่วมผ่านโทรศัพท์มือถือ ออกแบบและสร้างขึ้นโดยได้นำเทคโนโลยีที่ให้ความสะดวกต่อการแจ้งเตือนในสภาวะฉุกเฉินเมื่อเกิดอุทกภัยอย่างรวดเร็ว โดยใช้แผงวงจรไมโครคอนโทรลเลอร์เป็นศูนย์กลางในการควบคุมและสั่งการ เริ่มจากแปลงค่าสัญญาณอนาล็อกเป็นสัญญาณดิจิตอลที่รับมาจากอุปกรณ์ตรวจวัดระยะทาง เมื่อค่าที่ได้รับมาถูกส่งเข้าสู่แผงวงจรไมโครคอนโทรลเลอร์เพื่อประมวลผล หากค่าเกินกำหนดที่ตั้งไว้ แผงวงจรโทรศัพท์จะทำหน้าที่ส่งข้อความไปยังหมายเลขโทรศัพท์ที่กำหนดภายในโปรแกรม โดยข้อความที่ถูกส่งออกไปบอกถึงระดับน้ำที่เพิ่มขึ้นหรือลดลงได้ และเครื่องสัญญาณเตือนภัยน้ำท่วมผ่านโทรศัพท์มือถือสามารถนำไปใช้ในชีวิตประจำวันได้จริง เพื่อเตรียมพร้อมกับสถานการณ์ภัยธรรมชาติที่อาจเกิดขึ้นได้อย่างทันถ่วงทีอย่างมีประสิทธิผล

Project title	The flood warning system over mobile phone	
Name	Mr. Tanakorn Khuensorn	ID. 51383805
	Mr. Piyaphong Duangsupha	ID. 51384888
	Mr.Pongsatorn Potita	ID. 51384895
Project advisor	Mr. Akaraphant Vongkunghae, Ph.D.	
Major	Electrical Engineering	
Department	Electrical and Computer Engineering	
Academic year	2012	

Abstract

This project is a designing and building The flood warning system over mobile phone and take Technology to facilitate the notification in the event of flood emergencies quickly. This project by bring a microcontroller for control system. First,Microcontroller take analog to digital converter a signal from distance sensor. When computer receive a signal and a signal over setting in program, A mobile is send warning message. A message can tell level upper or lower. Which a warning system can be used in real life for ready to flood.

กิตติกรรมประกาศ

โครงงานฉบับนี้เป็นการศึกษาเกี่ยวกับเรื่อง สัญญาณเตือนภัยน้ำท่วมผ่านทางโทรศัพท์มือถือ ซึ่งจะไม่มีทางสำเร็จไปได้ ถ้าไม่ได้รับความช่วยเหลือจากบุคคลดังต่อไปนี้

ขอขอบพระคุณ ผศ.ดร.อัครพันธ์ วงศ์กังแห อาจารย์ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์ มหาวิทยาลัยนเรศวร ผู้เป็นอาจารย์ที่ปรึกษาโครงงานที่ได้ให้ความรู้ ให้คำแนะนำและให้ความช่วยเหลือแก่คณะผู้จัดทำเป็นอย่างดีตลอดมา

ขอขอบพระคุณ ผศ.ดร.สมพร เรืองสินชัยพานิช และ ดร.พิสุทธิ์ อภิชยกุล ซึ่งเป็นกรรมการที่ปรึกษาและให้คำแนะนำโครงงานในครั้งนี้

ขอขอบพระคุณ อาจารย์ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์ มหาวิทยาลัยนเรศวรทุกท่าน ที่ได้ให้ความรู้และให้คำสั่งสอนจนคณะผู้จัดทำมีความรู้ความสามารถนำมาประยุกต์ใช้ในการจัดทำโครงงานในครั้งนี้

และที่สำคัญที่สุดขอขอบพระคุณบิดา มารดา ที่ได้เลี้ยงดูและอบรมสั่งสอนแก่คณะผู้จัดทำจนทำให้คณะผู้จัดทำทุกคนมีวันนี้ได้ ซึ่งเป็นพระคุณอันหาเปรียบไม่ได้

ท้ายนี้คณะผู้จัดทำใคร่ขอขอบพระคุณผู้ที่มีส่วนเกี่ยวข้องทุกท่านที่ไม่ได้กล่าวนามมา ณ ที่นี้ ที่มีส่วนร่วมในการให้ข้อมูลเป็นที่ปรึกษาในการทำปริญญานิพนธ์ฉบับนี้จนเสร็จสมบูรณ์คณะผู้จัดทำ จึงขอขอบพระคุณไว้ ณ ที่นี้

นายทนากร เขื่อนสอน
นายปิยะพงศ์ ดวงสุภา
นายพงศธร โปธิตา

สารบัญ

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญตาราง

ตารางที่ หน้า

สารบัญรูปภาพ

รูปที่ หน้า

สารบัญรูป(ต่อ)

รูปที่ **หน้า**

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของโครงงาน

เนื่องจากปัจจุบันภัยธรรมชาติได้เกิดขึ้นมากมายในหลายๆประเทศต่างประสบปัญหาจากภัยธรรมชาติ เช่น เหตุการณ์แผ่นดินไหวที่ประเทศพม่าและในภาคเหนือของประเทศไทย สึนามิที่ประเทศญี่ปุ่น และเหตุการณ์น้ำท่วมในประเทศไทย เป็นต้น ภัยธรรมชาติเหล่านี้ล้วนก่อให้เกิดความสูญเสียทั้งทางด้านทรัพย์สินเงินทองและชีวิต ทำให้ผู้คนล้มตายเป็นจำนวนมาก

ด้วยเหตุนี้ จึงเกิดความสนใจและเห็นถึงความสำคัญในการป้องกันภัยพิบัติที่จะเกิดขึ้น ดังนั้นจึงเกิดแนวคิดที่จะประยุกต์การสร้างอุปกรณ์เตือนภัยให้แก่ประชาชนในเขตพื้นที่ใกล้เคียงได้ทราบถึงภัยพิบัติที่จะเกิดขึ้นอย่างมีประสิทธิภาพ และเพื่อเป็นการพัฒนาที่ดีต่อไปสำหรับการประมวลผลในการตัดสินใจ แล้วส่งข้อมูลไปยังโทรศัพท์มือถือเพื่อแจ้งเตือนภัย

โครงการนี้จะมุ่งเน้นในเรื่องการประมวลผลและแจ้งเตือนภัยน้ำท่วมผ่านทางโทรศัพท์มือถือ การสื่อสารแบบไร้สาย

1.2 วัตถุประสงค์ของโครงงาน

1. เพื่อศึกษาการรับส่งสัญญาณแบบไร้สาย
2. เพื่อศึกษาการพัฒนาวงจรการประมวลผลและส่งสัญญาณเตือนภัยทางโทรศัพท์มือถือ

1.3 ขอบเขตของโครงงาน

1. ศึกษาการรับส่งสัญญาณแบบไร้สาย
2. พัฒนาอุปกรณ์การประมวลผลและส่งสัญญาณเตือนภัยทางโทรศัพท์มือถือ เพื่อสามารถรับรู้ถึงภัยของน้ำท่วมที่จะเกิดขึ้นล่วงหน้า และสามารถป้องกันและอพยพประชาชนจากเหตุการณ์ได้ทันเวลา ทำให้ไม่สูญเสียชีวิตและทรัพย์สินอย่างที่ผ่านมา

1.4 ขั้นตอนการดำเนินงานและแผนการดำเนินงานตลอดโครงงาน

รายละเอียด	ระยะเวลาดำเนินงาน (เดือน)											
	ปี 2554							ปี 2555				
	มิ.ย.	ก.ค.	ส.ค.	ก.ย.	ต.ค.	พ.ย.	ธ.ค.	ม.ค.	ก.พ.	มี.ค.	เม.ย.	พ.ค.
1. รวบรวมข้อมูล	←	—	→									
2. ศึกษาการทำงานและรวบรวมเนื้อหา			←	—	—	→						
3. จัดทำอุปกรณ์และทำการทดลอง					←	—	—	→				
4. จัดทำรายงานรวบรวมข้อมูลสรุปผลเข้ารูปเล่มพร้อมทั้งชิ้นงานเสนอแนะแนวทางพัฒนาต่อและเตรียมนำเสนอ								←	—	—	—	→

1.5 ประโยชน์ที่คาดหวังจะได้รับจากโครงงาน

1. เข้าใจถึงหลักการส่งสัญญาณแบบไร้สาย
2. เพื่อเป็นการแจ้งเตือนให้ประชาชนในบริเวณใกล้เคียงที่เกิดเหตุทราบถึงระดับน้ำจากแหล่งน้ำที่ใกล้เคียง
3. สามารถนำอุปกรณ์ของโครงการนี้ไปใช้ในการให้บริการในพื้นที่ประสบภัยได้จริง
4. เพื่อเป็นประโยชน์สำหรับประชาชนที่อยู่ในเขตพื้นที่ประสบภัย

1.6 งบประมาณในการทำโครงงาน

1. ค่าอุปกรณ์ในการทำโครงงาน	3000	บาท
2. ถ่ายเอกสารและค่าเข้าเล่มโครงงานฉบับสมบูรณ์	500	บาท
3. ค่าพิมพ์เอกสารและอื่นๆ ฯลฯ	500	บาท
รวมเป็นเงิน	4000	บาท

บทที่2
หลักการและทฤษฎี

จากแนวคิดที่จะสร้างระบบเตือนภัย ซึ่งประกอบไปด้วยอุปกรณ์หลัก ๆ คือ แผงวงจรโทรศัพท์ (ET-GSM SIM300CZ V1.0) ซึ่งจะนำไปเชื่อมต่อกับแผงวงจรไมโครคอนโทรลเลอร์ตระกูล (ET-BASE AVR EASY328) เพื่อจะนำมาเขียนโปรแกรมควบคุมระบบทำงานอุปกรณ์ต่าง ๆ โดยหลักการและทฤษฎีพื้นฐานของแผงวงจรไมโครคอนโทรลเลอร์ ที่ควรทราบมีดังนี้

ไมโครคอนโทรลเลอร์ คือ อุปกรณ์ประเภทสารกึ่งตัวนำที่รวบรวมฟังก์ชันของการทำงานต่าง ๆ ไว้ภายในตัวของมันเอง โดยมีโครงสร้างใกล้เคียงกับคอมพิวเตอร์ คือ ภายในประกอบไปด้วยหน่วยรับข้อมูลโปรแกรมหน่วยรับข้อมูลโปรแกรมหน่วยประมวลผล หน่วยความจำ และหน่วยแสดงผล ซึ่งส่วนประกอบเหล่านี้มีความสมบูรณ์ในตัวของมันเอง ทำให้มีขนาดเล็ก และสามารถเขียนโปรแกรมควบคุมการทำงานของอุปกรณ์ต่างๆ ที่เชื่อมต่อกับตัวของมัน ซึ่งง่ายต่อการนำไปประยุกต์ใช้งาน

2.1 การเชื่อมต่อกับแผงวงจรไมโครคอนโทรลเลอร์ตระกูล ET-BASE AVR EASY328

ET-BASE AVR EASY328เป็นแผงวงจรไมโครคอนโทรเลอร์ในตระกูล AVR โดยแผงวงจรที่เลือกใช้เป็นไมโครคอนโทรเลอร์ตระกูล AVR เบอร์ ATMEGA328ของ ATMEL เป็น MCU ประจำแผงวงจร โดย MCU รุ่นนี้จะบรรจุอยู่ภายในตัวถังแบบ 28 Pin DIP โดย MCU ตัวนี้จะมีจุดเด่นคือเป็นไมโครคอนโทรลเลอร์ขนาดเล็กแต่เพียบพร้อมไปด้วยทรัพยากรพื้นฐานต่างๆอย่างครบถ้วนเหมาะแก่การใช้ในดารศึกษาเรียนรู้สำหรับผู้เริ่มต้นและยังสามารถนำไปประยุกต์ใช้งานต่างๆได้โดยง่ายซึ่ง MCU สามารถทำงานได้ด้วยความถี่สูงสุด 20MHz ที่ 1 Clock / Machine Cycle นอกจากนี้แล้วยังมีความเพียบพร้อมด้วยอุปกรณ์พื้นฐานต่างๆที่จำเป็นต่อการใช้งานไม่ว่าจะเป็นหน่วยความจำสำหรับเก็บข้อมูลแบบ EEPROM ขนาด 512 Byteและหน่วยความจำใช้งานแบบ SRAM อีก 1 K Byte ส่วนในด้านของอุปกรณ์ Peripheral นั้นก็นับว่าครบถ้วนเหมาะแก่การนำไปประยุกต์ใช้งานเกี่ยวกับการควบคุมและประมวลผลต่างๆได้เป็นอย่างดีโดยจะมีทั้งระบบฮาร์ดแวร์ของ SPI, UART, I2C, Watchdog, Timer/Counter, PWM และ ADC ฯลฯโดยการออกแบบโครงสร้างของแผงวงจรนั้นจะเน้นเรื่องขนาดของแผงวงจรให้มีขนาดเล็กเพื่อให้ง่ายต่อการนำไปประยุกต์ใช้งานและสะดวกต่อการพัฒนาโปรแกรมโดยลักษณะของแผงวงจรจะเน้นความคุ้มค่าและมีความอ่อนตัวทั้งด้านของการศึกษาทดลองและการนำไปประยุกต์ใช้งานจริงๆโดยในด้านของการศึกษาทดลองนั้นสามารถเลือกซื้อแผงวงจรทดลอง Input / Output ขนาดเล็กของ ET-MINI I/O แบบต่างๆที่ทางอีทีทีออกแบบและผลิตขึ้นมาสนับสนุนเพื่อเป็นทางเลือกให้ผู้ใช้ได้เลือกชุดอุปกรณ์ที่ตนเองสนใจศึกษาทดลองเพื่อนำมาติดตั้งใช้งานร่วมกับแผงวงจรของ ET-BASE AVR EASY328

ได้อย่างง่ายดายสำหรับกรณีที่จะนำแผงวงจรไปเขียนโปรแกรมเพื่อสร้างเป็นชิ้นงานจริงๆนั้นก็สามารถนำไปดัดแปลงหรือเชื่อมต่อกับอุปกรณ์ต่างๆได้โดยง่ายตามความเหมาะสมซึ่งเรียกได้ว่าแผงวงจรเดียวใช้ได้ทั้งเรียนรู้และใช้งาน

คุณสมบัติของบอร์ดET-BASE AVR EASY328

-เลือกใช้ MCU ตระกูล AVR เบอร์ ATMEGA328

-32KBYTE FLASH SRAM 2 KBYTE, EEPROM 1 KBYTE, RUN ความถี่ 19.6608 MHz

-มี PORT I/O ขนาด 20 BIT จำนวน 3 PORT (PB 6 BIT), (PC 6 BIT), (PD 8 BIT) โดยเป็น RS232, SPI, I2C, TIMER

-2 รูปแบบง่ายๆ ในการพัฒนา ET-BASE AVR EASY328

- รูปแบบโปรแกรมการพัฒนาด้วย ภาษา ซี (C++) ของ Arduino Project ในแบบ OPEN SOURCE โดยตัว MCU ของทาง อีทีที นี้ ได้ติดตั้งโปรแกรม BOOTLOADERไว้ในตัว MCU เรียบร้อยแล้ว สามารถ ดาวน์โหลดได้โดยตรงผ่านทาง RS232 PORT (ในกรณีต้องการผ่านทาง PORT USB ก็สามารถเพิ่มเติมการใช้งานได้ด้วยชุด ET-USB/RS232 MINI)
- รูปแบบโปรแกรมการพัฒนาด้วย AVR ปกติ ซึ่งสามารถเลือกใช้งานในรูป แบบโปรแกรมภาษาใดๆ ที่ทำงานรองรับ AVR เช่น ภาษาเบสิก , ภาษา ซี หรือด้วย WIN AVR ฯลฯ โดยใช้การ DOWNLOAD ผ่านทาง BOOTLOADER ทาง PORT RS232 หรือผ่านทางขั้วต่อ AVR ISP แบบ IDE 10PIN ที่มีอยู่แล้วบนแผงวงจร ใช้ร่วมกับแผงวงจรET-AVR PROG MINI, ET-AVR ISP USB V1 ฯลฯ

-ขั้วต่อใช้งาน 10PIN ET 3 ชุด, ขั้วต่อ OUTPUT ด้วย 74HC595 แบบ 10PIN IDE ET 1 ชุด

-SW RESET และ SW BL (PD2) สำหรับใช้ลบข้อมูลแผงวงจรเข้าในการทำงานแบบ BOOTLOADER ผ่านทาง PORT RS232

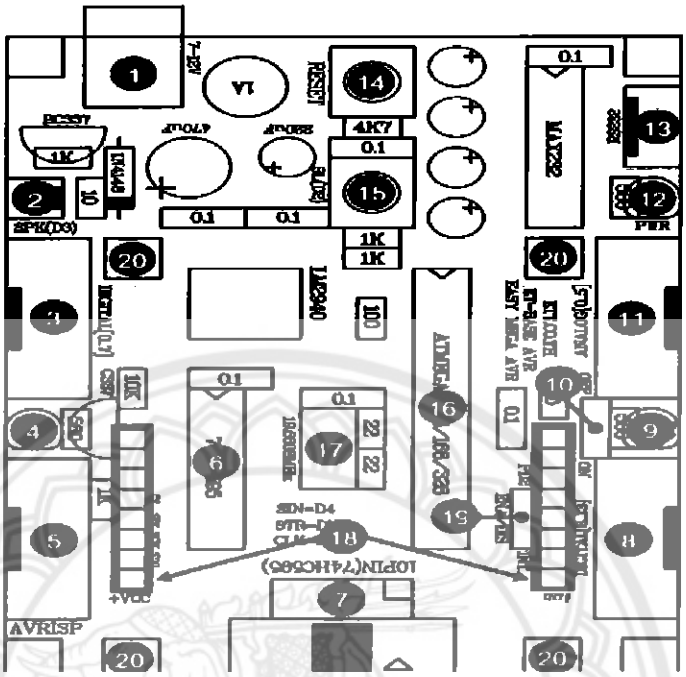
-RS232 PORT แบบ 4 PIN ET ใช้งานและใช้ ดาวน์โหลดโปรแกรม

-10 PIN IDE มาตรฐาน AVR ISP สำหรับโปรแกรมแบบไม่ผ่าน PORT RS232

-มีฐานยึดบนแผงวงจรใช้ติดตั้งแผงวงจรในการทดลองในตระกูล ET-MINI I/O ต่างๆ ได้โดยตรง สะดวกในการทดลอง

-POWER SUPPLY 7 - 10 VDC, ใช้ LM2940 (LOW DROP) ON BOARD

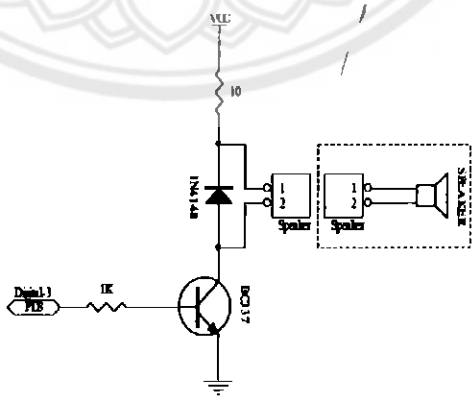
2.2 โครงสร้างแผงวงจร ET-BASE AVR EASY328



รูปที่ 2.1 โครงสร้างของบอร์ด ET-BASE AVR EASY328

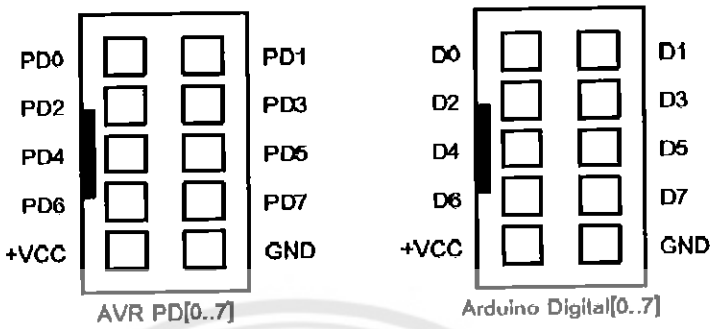
-หมายเลข 1 คือขั้วต่อแหล่งจ่ายไฟเลี้ยงวงจรของแผงวงจรใช้กับแหล่งจ่าย 7-10VAC/DC

-หมายเลข 2 เป็นขั้วต่อสำหรับใช้ต่อกับลำโพงซึ่งถูกวงจรขับผ่านทรานซิสเตอร์ BC337 ดังรูปที่ 2.2



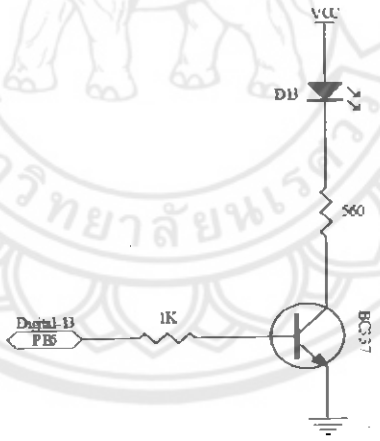
รูปที่ 2.2 วงจรต่อลำโพงผ่านทรานซิสเตอร์ BC337

-หมายเลข 3 เป็นขั้วต่อสัญญาณจาก PD [0..7] ซึ่งในกรณีใช้การพัฒนาโปรแกรมด้วย Arduino จะเป็นขาสัญญาณของ Digital [0..7]



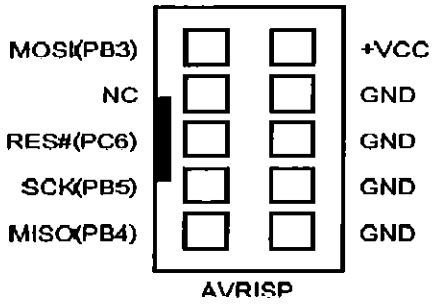
รูปที่ 2.3 ขั้วต่อสัญญาณจาก PD [0.7]

-หมายเลข 4 เป็น LED ใช้แสดงสถานะของขาสัญญาณ PB[5] หรือ Digital[13] ของ Arduino ซึ่งLED นี้จะถูกวงจรขับผ่านทรานซิสเตอร์ BC337 ดังวงจร



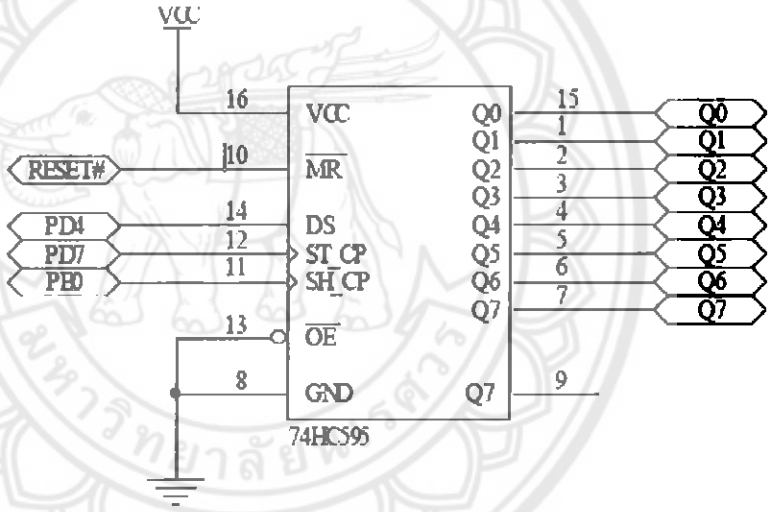
รูปที่ 2.4 วงจรต่อLED ผ่านทรานซิสเตอร์ BC337

-หมายเลข 5 เป็นขั้วต่อ AVRISP ใช้สำหรับดาวน์โหลดคำสั่งให้กับ MCU ในกรณีที่ใช้การพัฒนาโปรแกรมของแผงวงจรเป็นแบบ MCU ของ AVR ตามปรกติโดยไม่ผ่านระบบBoot loaderโดยขั้วต่อAVRISP นี้จะสามารถใช้งานได้กับเครื่องโปรแกรมทุกรุ่นที่รองรับการใช้งานกับ ATMEGA88 และใช้ขั้วต่อตรงตามมาตรฐาน AVRISP ดังรูป



รูปที่ 2.5 ขั้วต่อ AVRISP

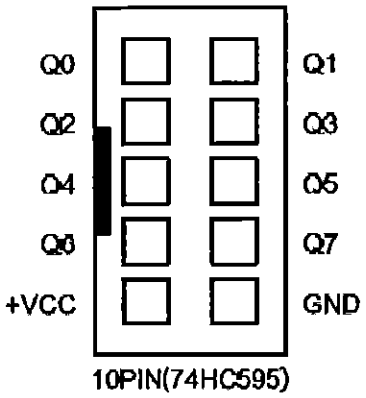
-หมายเลข 6 เป็นไอซีเบอร์ 74HC595 ซึ่งใช้ขยายสัญญาณขาออกขนาด 8 บิตโดยมีการต่อวงจรร่วมกับ MCU ที่ใช้ในบอร์ดคังวงจร



รูปที่ 2.6 การต่อวงจรร่วมกับ MCU

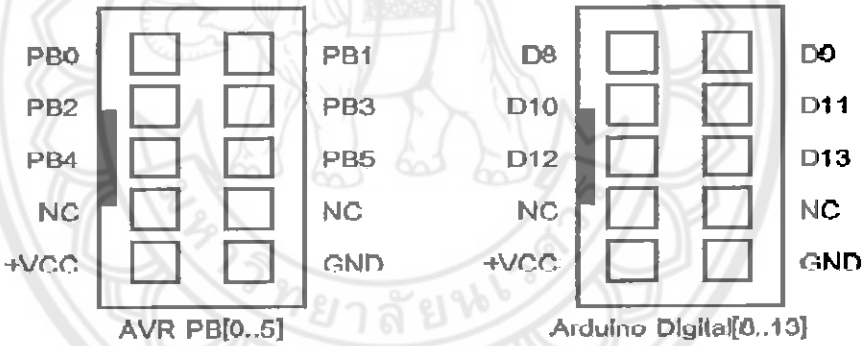
โดยสัญญาณขาออกของ 74HC595 นี้สามารถนำไปประยุกต์ใช้งานเพื่อทำหน้าที่เป็นสัญญาณขาออกทั่วๆไปหรือใช้สำหรับเชื่อมต่อกับ Character LCD ในแบบ 4 Bit Mode ก็ได้เช่นเดียวกัน

-หมายเลข 7 เป็นขั้วต่อสัญญาณขาออกจาก 74HC595 ซึ่งมีขนาด 8บิตก็คือQ [0...7] โดยมีการจัดเรียงขาสัญญาณดังนี้



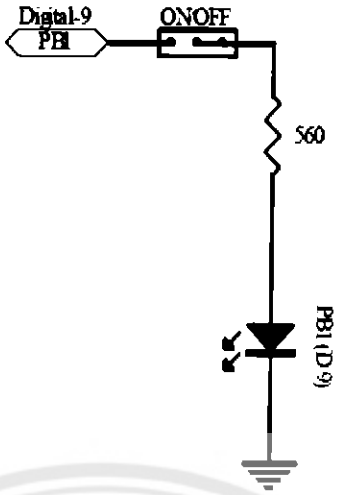
รูปที่ 2.7 ขั้วต่อสัญญาณ สัญญาณขาออกจาก 74HC595

-หมายเลข 8 เป็นขั้วต่อสัญญาณจากPB [0...5] ซึ่งในกรณีใช้การพัฒนาโปรแกรมด้วย Arduino จะเป็นขาสัญญาณของDigital [8...13]



รูปที่ 2.8 ขั้วต่อสัญญาณจาก PB [0...5] ซึ่งในกรณีใช้การพัฒนาโปรแกรมด้วย Arduino จะเป็นขาสัญญาณของ Digital [8...13]

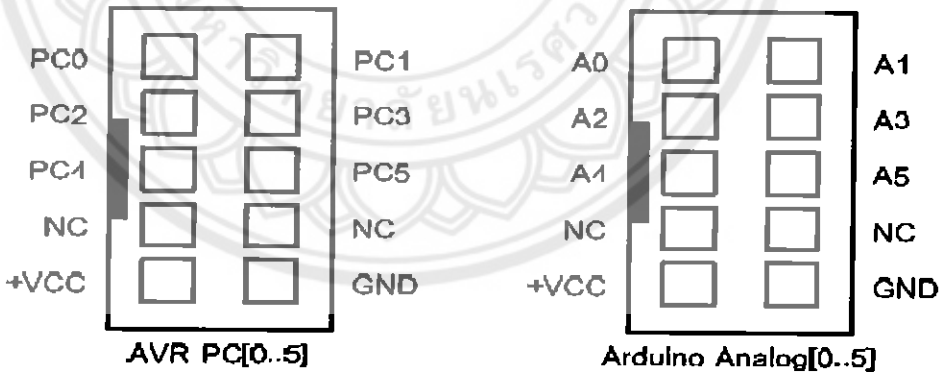
-หมายเลข 9 เป็น LED ใช้แสดงสถานะของขาสัญญาณPB [1] หรือDigital [9] ของ Arduino ซึ่ง LED นี้จะถูกต่อวงจรแบบ Source Current จากขาสัญญาณของ MCU โดยมี Jumper เป็นตัวตัดต่อสัญญาณระหว่าง PB [1] กับ LED ซึ่ง LED นี้สามารถแสดงผลได้ 2 แบบคือใช้ทดสอบการแสดงผลแบบ ON/OFF เมื่อกำหนดขาสัญญาณ PB [1] เป็นแบบสัญญาณดิจิตอลขาออกและใช้ทดสอบการแสดงผลแบบ Dimmer เมื่อกำหนดขาสัญญาณ PB [1] เป็นแบบ Output PWM ดังวงจร



รูปที่ 2.9 การต่อวงจรLED ใช้แสดงสถานะของขาสัญญาณ PB [1] หรือ Digital [9]

-หมายเลข 10 เป็น Jumper สำหรับใช้ในการตัดต่อสัญญาณPB [1] กับ LED โดยเมื่อเลือกไว้ด้าน ON จะเป็นการต่อสัญญาณ PB [1] เข้ากับ LED แต่เมื่อเลือก OFF จะเป็นการตัดการเชื่อมต่อของ PB [1] ออกจากวงจรแสดงผลของ LED

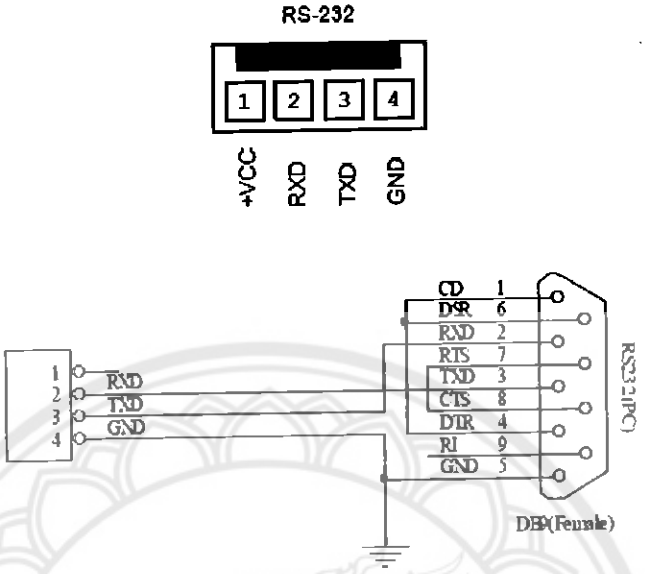
-หมายเลข 11 เป็นขั้วต่อสัญญาณจาก PC [0...5] ซึ่งในกรณีใช้การพัฒนาโปรแกรมด้วย Arduino จะเป็นขาสัญญาณของ Analog [0...5]



รูปที่ 2.10 ขั้วต่อสัญญาณจาก PB [0...5] ซึ่งในกรณีใช้การพัฒนาโปรแกรมด้วย Arduino จะเป็นขาสัญญาณของ Analog [0...5]

-หมายเลข 12 เป็น LED Power ใช้แสดงสถานะของแหล่งจ่ายไฟ +5VDC

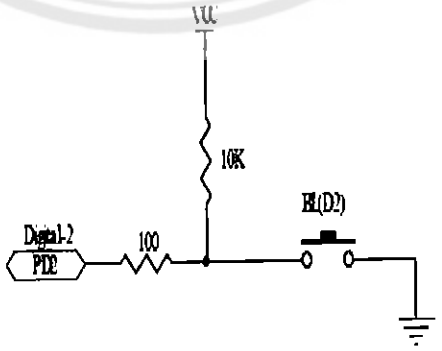
-หมายเลข 13 คือขั้วต่อ RS232 สำหรับใช้งานทั่วไปและ Upload Code ให้กับ MCU ผ่านระบบ Boot loaderโดยมีการจัดเรียงสัญญาณดังนี้



รูปที่ 2.11 ขั้วต่อ RS232 สำหรับใช้งานทั่วไปและ Upload Code ให้กับ MCU

-หมายเลข 14 คือสวิตช์ RESET ใช้สำหรับเริ่มการทำงานใหม่ของ MCU

-หมายเลข 15 คือสวิตช์ BL (Boot loader) โดยต่อผ่านขาสัญญาณPD [2] ใช้สำหรับสร้างสัญญาณโลจิก LOW ให้กับขาสัญญาณ PD [2] เพื่อทดสอบการรับค่า Input รวมทั้งการสร้างสัญญาณ Trigger Interrupt ของ INT0 รวมทั้งการใช้สั่งให้ MCU เข้าทำงานในBoot loaderโดยใช้งานร่วมกับสวิตช์ RESET โดยสวิตช์ BL มีการต่อวงจรดังรูป



รูปที่ 2.12 การต่อวงจรสวิตช์ BL (Boot loader) โดยต่อผ่านขาสัญญาณ PD [2] ใช้สำหรับสร้างสัญญาณโลจิก LOW ให้กับขาสัญญาณ PD [2]

-หมายเลข 16 เป็น MCU ประจำแผงวงจรซึ่งสามารถใช้ได้กับ AVR ขนาด 28ขาได้หลายเบอร์เช่น ATMEGA8, ATMEGA88, ATMEGA168 และ ATMEGA328

-หมายเลข 17 เป็น Crystal Oscillator ค่าความถี่ 19.6608 MHz

-หมายเลข 18 เป็น Header สำหรับรองรับการเชื่อมต่อสัญญาณกับแผงวงจร ET-MINI ENC28J60 ของบริษัทอีทีทีจำกัดสำหรับใช้พัฒนาโปรแกรมใช้งานกับระบบ Ethernet LAN

-หมายเลข 19 เป็น Jumper สำหรับใช้ตัดต่อขาสัญญาณของ PD2 (INT0) ที่เชื่อมต่อระหว่างPD2 (INT0) ของบอร์ด ET-BASE AVR EASY88 กับ INT ของแผงวงจร ET-MINI ENC28J60 ซึ่งถ้าเลือกไว้ด้าน ENA หมายถึง Enable ซึ่งจะเป็นการเชื่อมต่อขา INT จาก ENC28J60 เข้ากับขา PD2 หรือ INT0 ของ ATMEGA88 แต่เมื่อเลือกไว้ทางด้าน DIS จะหมายถึง Disable ซึ่งเป็นการตัดการเชื่อมต่อขา INT ของ ENC28J60 ออกจากขา PD2 (INT0) ของ ATMEGA88 ซึ่งตามปรกติควรเลือกไว้ที่ด้าน DIS เสมอ

-หมายเลข 20 เป็นตำแหน่งฐานรองสำหรับยึดแผงวงจรทดลองขนาดเล็กของบริษัทอีทีทีจำกัดที่มีขนาดมาตรฐานในขนาด MINI I/O Size ซึ่งผู้ใช้สามารถนำแผงวงจรชุด ET-MINI I/O ต่างๆมาต่อทดลองร่วมกับแผงวงจร ET-BASE AVR EASY328 ได้ทันที

2.3 แผงวงจรโทรศัพท์ (ET-GSM SIM300CZ)

แผงวงจรโทรศัพท์ ET-GSM SIM300CZเป็นชุดเรียนรู้และพัฒนาระบบการสื่อสารไร้สาย โดยใช้โมดูล GSM/GPRSรุ่น SIM300CZ ของ "SIMCom Ltd." เป็นอุปกรณ์หลักซึ่ง SIM300CZ เป็นโมดูลสื่อสารระบบ GSM/GPRSขนาดเล็กรองรับระบบสื่อสาร GSM

RS232 ด้วยชุดคำสั่ง AT Command สามารถประยุกต์ใช้งานได้มากมายหลายรูปแบบไม่ว่าจะเป็นการรับส่งสัญญาณแบบ Voice, SMS, Data, FAX และยังรวมถึงการสื่อสารด้วย Protocol TCP/IP ด้วยซึ่งตามปรกติแล้วถึงแม้ว่าโมดูล SIM300CZ จะมีวงจรและ Firmware บรรจุไว้ภายในตัวเป็นที่เรียบร้อยแล้วก็ยังไม่สามารถนำไปใช้งานได้โดยตรงทันทีเนื่องจากในการใช้งานจริงๆนั้นผู้ใช้งานเองจำเป็นต้องออกแบบวงจรรอบนอกที่จำเป็นมาเชื่อมต่อกับขาสัญญาณของตัวโมดูลอีกในบางส่วนไม่ว่าจะเป็นวงจรภาคจ่ายแรงดัน, วงจรเชื่อมต่อกับการ์ดโทรศัพท์รวมไปถึงวงจร Line Driver ของ RS232 เป็นต้นดังนั้นทางทีมงานอีทีทีจึงได้จัดสร้างบอร์ดสำหรับเป็นตัวกลางในการเชื่อมต่อระหว่างโมดูล SIM300CZ กับอุปกรณ์ภายนอกเพื่อให้ผู้ใช้งานสามารถนำโมดูล GSM ของ SIM300CZ ไปทำการทดลองและศึกษาเรียนรู้การสั่งงานต่างๆได้โดยสะดวกก่อนที่จะนำเอาโมดูลตัวนี้ไปออกแบบดัดแปลงและประยุกต์ใช้งานในด้านต่างๆได้ต่อไปในอนาคตซึ่งถึงแม้ว่าวงจรการเชื่อมต่อทั้งหมดที่ทางอีทีทีได้จัดทำขึ้นมานี้จะยังไม่สามารถรองรับการใช้งานทรัพยากรต่างๆที่มีอยู่ภายในโมดูลได้ครบถ้วนทั้งหมดก็ตามทีแต่ในส่วนของการใช้งานโมดูลในส่วนที่เป็นความสามารถหลักๆที่จำเป็นนั้นมีไว้รองรับอย่างครบถ้วนเพียงพอแล้วอย่างไรก็ตามถ้าผู้ใช้งานต้องการพัฒนา Application ที่สูงขึ้นไปก็สามารถประยุกต์ดัดแปลงหรือทำการเชื่อมต่ออุปกรณ์เพิ่มเติมให้กับบอร์ดได้โดยง่ายทั้งนี้ก็เพราะว่าขาสัญญาณต่างๆจากโมดูลในส่วนที่ยังไม่ได้ทำการออกแบบวงจรเตรียมไว้ให้ภายในแผงวงจร เช่นขาสัญญาณสำหรับเชื่อมต่อกับ Keyboard, LCD Display และ GPIO ต่างๆนั้นทางอีทีทีเองก็ได้จัดทำเป็นจุดต่อ Connector เตรียมไว้ให้เป็นที่เรียบร้อยแล้วผู้ใช้เพียงแต่ทำการเชื่อมต่อสัญญาณต่างๆจากจุดเชื่อมต่อที่เตรียมไว้ไปยังวงจรส่วนที่ได้ทำการออกแบบไว้ได้โดยสะดวกอยู่แล้ว

2.3.1 คุณสมบัติของแผงวงจรโทรศัพท์ ET-GSM SIM300CZ V1.0

-มีสวิตช์แบบ Push-Button สำหรับใช้สั่งเปิด-ปิดการทำงานของโมดูลภายในบอร์ด

-มี Socket SIM รองรับ SIM Card พร้อมวงจร ESD ป้องกัน SIM เสียหาย

-มีวงจร Regulate แยกอิสระจำนวน 2 ชุดสามารถใช้กับแหล่งจ่ายภาย Adapter ขนาดตั้งแต่ +5Vขึ้นไปสามารถจ่ายกระแสให้กับโมดูล SIM300CZ และอุปกรณ์เชื่อมต่อต่างๆได้อย่างเพียงพอ

-มีวงจร Regulate ขนาด 4.2V / 3A สำหรับจ่ายให้กับโมดูล SIM300CZ ได้อย่างเพียงพอสามารถใช้กับ SIM ของระบบ GSM900MHz แบบ 2-Watt ได้อย่างไม่เกิดปัญหา

-มีวงจร Regulate ขนาด 3.3V / 1A สำหรับจ่ายให้กับวงจรเชื่อมต่อภายนอกโดยไม่ต้องไปดึงไฟจากตัวโมดูลมาใช้ป้องกันปัญหาโมดูลเสียหายจากวงจรภายนอกดึงกระแสเกินพิกัดและสะดวกต่อการออกแบบวงจรเชื่อมต่อเพิ่มเติมโดยไม่ต้องกังวลว่ากระแสจะไม่พอจ่ายให้กับอุปกรณ์

-มีวงจร Line Driver สำหรับแปลงระดับสัญญาณโลจิกจากโมดูล SIM300CZ ให้เป็น RS232 ระดับมาตรฐานครบทุกเส้นสัญญาณทั้งพอร์ตที่ใช้ในการสื่อสารสำหรับสั่งงานโมดูลและพอร์ตสำหรับใช้ในการพัฒนาโปรแกรม (Debug) สามารถเชื่อมต่อกับพอร์ต RS232 มาตรฐานได้ทันที

-มี LED แสดงสถานะพร้อมในแผงวงจรสำหรับแสดงสถานะของแหล่งจ่ายไฟสถานะพร้อมทำงานของโมดูลสถานะในการเชื่อมต่อกับ Network และสถานะ Power-On/Power-OFF ของโมดูล

-มีขั้วสำหรับเชื่อมต่อกับ Handset (ชุดปากพูดและหูฟังของโทรศัพท์บ้าน) โดยใช้ขั้วต่อแบบ RJ11 มาตรฐานพร้อมวงจร Voice Filter สามารถนำชุด Handset ของโทรศัพท์บ้านต่อเข้ากับแผงวงจรทางขั้วต่อแบบ RJ11 สำหรับใช้พูดคุยโทรออกและรับสายได้โดยสะดวก

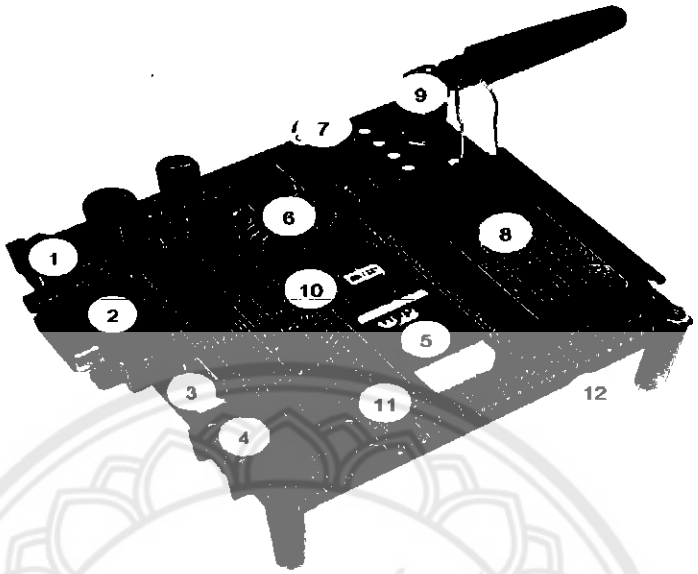
-มี Buzzer พร้อมวงจรขับเพื่อสร้างสัญญาณเสียงในกรณีมีการโทรเรียกเข้ามายังโมดูล

-มีจุดยึดเสาอากาศสำหรับใช้เป็นจุดพักสำหรับเชื่อมต่อกับเสาอากาศแบบต่างๆได้โดยสะดวก

-มีขั้วต่อสำหรับติดตั้งโมดูล SIM300CZ พร้อมเสารองและสกรูยึดโมดูลกับตัวบอร์ด

-มีจุดต่อสัญญาณอื่นๆที่เหลือจากโมดูลเช่นแป้นพิมพ์, หน้าจอ, GPIO, อุปกรณ์ชาร์จแบตเตอรี่ฯลฯสำหรับให้ผู้ใช้ต่อขยายไปยังวงจรที่ออกแบบเพิ่มเติมได้โดยง่ายและสะดวก

2.3.2 โครงสร้างของแผงวงจรโทรศัพท์ ET-GSM SIM300CZ V1.0



รูปที่ 2.13 แผงวงจรโทรศัพท์ ET-GSM SIM300CZ

-หมายเลข 1 เป็น JACK DC-IN แบบมีขั้วโดยมีด้านนอกเป็นขั้วบวกและด้านในเป็น GND ใช้สำหรับรับแหล่งจ่ายไฟจากภายนอกโดยออกแบบให้ใช้กับแหล่งจ่ายไฟขนาด 5 V ขึ้นไปที่จ่ายกระแสได้ 1A ถึง 3A

-หมายเลข 2 เป็นขั้วต่อ RS232 (DCE) แบบ DB9 ตัวเมียสำหรับใช้เชื่อมต่อกับสัญญาณRS232 (DTE) แบบ DB9 ตัวผู้จากคอมพิวเตอร์ PC หรืออุปกรณ์ภายนอกอื่นๆโดยใช้สาย 9 Pinแบบต่อตรง

-หมายเลข 3 เป็นขั้วต่อ DEBUG ใช้สำหรับพัฒนาและ DEBUG โปรแกรมสำหรับต่อกับ RS232ในกรณีที่ต้องการพัฒนาโปรแกรมเพิ่มเติมให้กับโมดูล SIM300CZ เอง

-หมายเลข 4 เป็นขั้วต่อ RJ11 สำหรับใช้เชื่อมต่อกับชุด Handset ในกรณีที่ต้องการใช้งานโมดูล SIM300CZ เพื่อโทรออกและรับสายโดยสามารถเชื่อมต่อกับ Handset มาตรฐานได้ทั่วไป

-หมายเลข 5 เป็น Socket สำหรับติดตั้ง SIM Card ให้กับโมดูล

-หมายเลข 6 เป็น Switch Push-Button สำหรับใช้ Power-On และ Power-OFF ตัวโมดูล

-หมายเลข 7 เป็น Buzzer สำหรับสร้างเสียงเรียกเข้าในกรณีที่มีการโทรเข้ามายังโมดูล SIM300CZ

-หมายเลข 8 เป็นจุดรองรับโมดูล SIM300CZ พร้อมเสาและสกรูสำหรับยึดโมดูลกับแผงวงจร

-หมายเลข 9 เป็นจุดยึด Connector เสาอากาศ GSM/GPRS ย่านความถี่ 900/1800/1900MHz

-หมายเลข 10 เป็น LED แสดงแหล่งจ่าย VBAT โดยจะติดสว่างเมื่อมีการจ่ายไฟให้แผงวงจรแล้ว

-หมายเลข 11 เป็น LED แสดงสถานะของแผงวงจรซึ่งมีด้วยกัน 3 ดวงคือ

-POWER สีแดงจะติดสว่างเมื่อโมดูลอยู่ในสถานะ Power-ON

-NETLIGHT สีเหลืองจะกระพริบเมื่อโมดูลอยู่ในสถานะ Power-ON

-STATUS สีเขียวจะติดสว่างเมื่อโมดูลอยู่ในสถานะ Power-ON

-หมายเลข 12 เป็นจุดต่อสัญญาณเพิ่มเติมในกรณีที่ต้องการประยุกต์ใช้งานโมดูลเพิ่มเติม

2.3.3 คุณสมบัติของโมดูล SIM300CZ

-รองรับความถี่ GSM/GPRS 900/1800/1900MHz

-รองรับ GPRS Multi-Slot Class10 และ GPRS Mobile Station Class B

-รองรับมาตรฐานคำสั่ง AT Command (GSM 07.07 / 07.05 และคำสั่งเพิ่มเติมจาก SIMCOM)

-รองรับ SIM Applications Toolkit

-ทำงานที่ย่านแรงดัน 3.4V ถึง 4.5V

-รองรับการเชื่อมต่อภายนอก

-ใช้ได้กับ SIM 3V และ 1.8V

-มีวงจร Analog Audio (MIC & Speaker) จำนวน 2 ชุด

-รองรับ 5x5 Keypad Interface & SPI LCD Interface

-มีระบบ RTC พร้อมวงจร Backup

-มีขั้วต่อเสาอากาศภายนอกแบบ Connector และจุดเชื่อมต่อแบบ PAD

-มีระบบ Battery Charge ในตัว

2.3.4 อุปกรณ์แสดงการทำงานของโมดูล SIM300CZ

สำหรับแผงวงจรโทรศัพท์ ET-GSM SIM300CZ V1.0 นั้นได้ออกแบบอุปกรณ์แสดงผลการทำงานของแผงวงจรไว้ในแผงวงจรเพื่อใช้แสดงสถานะของการทำงานต่างๆให้ผู้ใช้ทราบด้วย คือ

-Buzzer ใช้แสดงการทำงานของโมดูลเมื่อมีสายเรียกเข้าโดยการทำงานของ Buzzer นี้จะถูกควบคุมด้วยสัญญาณ BUZZER (Pin23) ของโมดูล SIM300CZ และสามารถปรับระดับความดังของเสียงได้จากคำสั่ง "AT+CRSL" ได้อีกด้วย

-LED VBAT ใช้ทำหน้าที่แสดงสถานะของแหล่งจ่ายไฟจากภายนอกที่ต่อมาให้กับแผงวงจรโดย LEDนี้จะติดสว่างก็ต่อเมื่อมีการจ่ายไฟให้กับแผงวงจรเป็นที่เรียบร้อยแล้ว

-**LED POWER** ใช้แสดงสถานะความพร้อมของโมดูล SIM300CZ ว่าอยู่ในสภานะ Power ONหรือPower OFFโดย LED ตัวนี้จะถูกควบคุมการทำงานด้วยสัญญาณ VDD_EXT (Pin15) ของโมดูลเมื่อทำงานจะมีสถานะทางโลจิกเป็นโลจิก "1" โดยถ้า LED Power ติดสว่างแสดงว่าโมดูล SIM300CZ อยู่ในสถานะ Power ON และพร้อมทำงานแต่ถ้า LED นี้ดับแสดงว่าโมดูลอยู่ในสถานะ Power OFF อยู่

-**LED NETLIGHT** ใช้แสดงสถานะของโมดูลในขณะทำการเชื่อมต่อกับเครือข่ายอยู่โดย LED ตัวนี้จะถูกควบคุมด้วยสัญญาณNETLIGHT (Pin16) ของโมดูล SIM300CZ เมื่อทำงานจะมีสถานะทางโลจิกเป็นโลจิก "1" โดยเมื่อโมดูลอยู่ในสถานะพร้อมทำงาน LED นี้จะติดกระพริบด้วยค่าความเร็วต่างๆซึ่งมีความหมายดังนี้

-OFF แสดงว่าโมดูลอยู่ในสถานะของ Power OFF (ไม่ทำงาน)

-64mS ON / 800mS OFF แสดงว่าโมดูล SIM300CZ ทำงานปรกติและไม่ได้อยู่ระหว่างทำการค้นหาเครือข่ายอยู่

-64mS ON / 3000mS OFF แสดงว่าโมดูล SIM300CZ กำลังทำการค้นหาเครือข่ายเพื่อทำการเชื่อมต่อสัญญาณ

-64mS ON / 300mS OFF แสดงว่าโมดูล SIM300CZ อยู่ระหว่างการเชื่อมต่อกับเครือข่ายหรืออุปกรณ์อื่นๆด้วย GPRS อยู่

-**LED STATUS** ใช้แสดงสถานะของโมดูล SIM300CZ ว่าพร้อมทำงานหรือไม่โดย LED ตัวนี้จะถูกควบคุมด้วยสัญญาณ STATUS (Pin19) ของโมดูล SIM300CZ เมื่อทำงานจะมีสถานะทางโลจิกเป็นโลจิก "1" ซึ่งเมื่อ LED นี้ติดสว่างแสดงว่าโมดูลพร้อมรับคำสั่งต่างๆได้แต่ถ้า LED ดับแสดงว่าโมดูลยังไม่พร้อมทำงาน

2.3.5 การสั่งเปิดและปิดการทำงานของโมดูล

ตามปกติแล้วโมดูล SIM300CZ จะมีโหมดการทำงานอยู่หลายโหมดสามารถทำงานสั่งเปิดและปิดการทำงานของโมดูลได้หลายวิธี

-Switch ON/OFF เป็นการสั่งเปิดและปิดการทำงานของโมดูล SIM300CZ ด้วยการกดสวิตช์โดยสวิตช์ตัวนี้จะเป็นแบบ Push-Button Switch (สวิตช์กดติด-ปล่อยดับ) โดยเป็นการกำหนดสถานะทางโลจิกให้กับขาสัญญาณ PWRKEY (Pin17) ของโมดูลโดยเมื่อกดสวิตช์จะเป็นโลจิก "0"เมื่อปล่อยสวิตช์จะเป็นโลจิก "1" โดยการทำงานของสวิตช์จะต้องทำการกดสวิตช์ต่อเนื่องกันเป็นเวลานานอย่างน้อย 2000mS (2 วินาที) จึงจะมีผลต่อการทำงานของโมดูลโดยลักษณะการทำงานของสวิตช์จะเป็นแบบ Toggle กล่าวคือถ้าโมดูลอยู่ในสถานะของ Power OFF อยู่แล้วทำการกดสวิตช์เป็นเวลาอย่างน้อย 2000mS (2 วินาที) จะเป็นการสั่งให้โมดูลกลับเข้าสู่Power On หรือพร้อมทำงานแต่ถ้าหากว่าโมดูลอยู่ในสถานะของ Power ON อยู่แล้วทำการกดสวิตช์เป็นเวลาอย่าง

น้อย 2000mS (2 วินาที) แล้วปล่อยจะเป็นการสั่งให้โมดูลหยุดทำงานและกลับเข้าสู่สถานะของ Power OFF (หยุดทำงาน)

ตารางที่ 2.1 สถานะของ LED ในโหมดต่างๆ

LED สถานะ	Power-ON	Power-OFF
VBAT (แดง)	ติดสว่าง	ติดสว่าง
POWER (แดง)	ติดสว่าง	ดับ
NETLIGHT (เหลือง)	กระพริบ	ดับ
STATUS (เขียว)	ติดสว่าง	ดับ

หลังจากทำการสั่ง Power-ON ในครั้งแรกนั้นก่อนที่จะเริ่มต้นส่งคำสั่งใดๆให้กับโมดูลควรรอให้ตัวโมดูลพร้อมเสียก่อน โดยจะมีข้อความ "Call Ready" ปรากฏให้เห็นในกรณีที่กำหนด Baud rate เป็นแบบAuto Baud rate ไว้ (AT+IPR=0") เมื่อทำการ Power-ON จะได้ผลดังตัวอย่าง

```
Call Ready
```

ในกรณีที่กำหนด Baud rate เป็นแบบ Fix Baud rate ไว้ (AT+IPR=ค่า Baud rate) เมื่อทำการสั่งให้โมดูล Power-ON แต่ละครั้งจะได้ผลดังตัวอย่าง

```
RDY
+CFUN: 1
+CPIN: READY
Call Ready
```

การติดต่อสื่อสารกับโมดูล SIM300CZ

การติดต่อสื่อสารกับโมดูล SIM300CZ ของแผงวงจร ET-GSM SIM300CZ นั้นจะเชื่อมต่อผ่านพอร์ตสื่อสารอนุกรม RS232 โดยใช้ขั้วต่อแบบ DB9 ตัวเมียจัดเรียงสัญญาณตามมาตรฐาน RS232-DCEสามารถนำไปเชื่อมต่อกับสัญญาณ RS232-DTE มาตรฐานโดยใช้สาย DB9 แบบต่อตรงได้ทันทีโดยสัญญาณทั้งหมดที่ DB9 นี้ได้ผ่านวงจร Line Driver เพื่อแปลงสัญญาณระดับโลจิกจากโมดูลให้เป็นสัญญาณระดับมาตรฐาน RS232 เป็นที่เรียบร้อยแล้วซึ่งถ้าต้องการนำไปเชื่อมต่อกับ RS232(Com Port)ของคอมพิวเตอร์ PC ก็สามารถทำการเชื่อมต่อกันโดยตรงได้ทันทีโดยไม่ต้องทำการสลับสายสัญญาณใดๆทั้งสิ้นโดยสัญญาณเชื่อมต่อทางด้านโมดูล SIM300CZ นั้นจะมีทั้งหมด 8 เส้นสัญญาณซึ่งในการเชื่อมต่อใช้งานนั้นจะต่อให้ครบทั้ง 8 เส้นหรือจะเลือกต่อเพียง 3 เส้น (RXD,TXD และ GND) ก็ได้เช่นเดียวกันโดยสามารถกำหนดได้จากการ Setup ค่า Configuration และคำสั่งใช้งานโดยสัญญาณการเชื่อมต่อ RS232ด้านโมดูล SIM300CZ จะมีดังนี้

-Pin 1 เป็นขา DCD (Data Carrier Detect) ของโมดูล SIM300CZ ซึ่งเป็น Output จาก SIM300CZที่ได้ผ่านการแปลงระดับสัญญาณเป็น RS232 แล้วซึ่งตามปรกติจะต่อเข้ากับ DCD Input ของอุปกรณ์ด้าน Host หรือคอมพิวเตอร์

-Pin 2 เป็นขา TXD (Transmit Data) ของโมดูล SIM300CZ ซึ่งเป็น Output จาก SIM300CZ ที่ได้ผ่านการแปลงระดับสัญญาณเป็น RS232 แล้วซึ่งตามปรกติจะต่อเข้ากับ RXD (Receive Data) ของอุปกรณ์ด้าน Host หรือคอมพิวเตอร์

-Pin 3 เป็นขา RXD (Receive Data) ของโมดูล SIM300CZ ซึ่งเป็น Input ของ SIM300CZ สามารถรับสัญญาณระดับ RS232 ได้โดยตรงซึ่งตามปรกติจะต่อเข้ากับ TXD (Transmit Data) จากอุปกรณ์ด้าน Host หรือคอมพิวเตอร์

-Pin 4 เป็นขา DTR (Data Terminal Ready) ของโมดูล SIM300CZ ซึ่งเป็น Input ของ SIM300CZซึ่งตามปรกติจะต่อเข้ากับ DTR จากอุปกรณ์ด้าน Host หรือคอมพิวเตอร์

-Pin 5 เป็นสัญญาณ GND ของโมดูล SIM300CZ ต้องต่อเข้ากับ GND ของอุปกรณ์ด้าน Host หรือคอมพิวเตอร์

-Pin 6 ตามปรกติแล้วเป็นสัญญาณ DSR (Data Set Ready) แต่ในกรณีของ SIM300CZ จะไม่ได้ต่อใช้งานแต่อย่างไรก็ตามในแผงวงจรได้ทำการป้อนสัญญาณย้อนกลับหรือ Loop Back สัญญาณ DTR (Data Terminal Ready) ซึ่งเป็น Output ส่งมาจาก Host หรือคอมพิวเตอร์กลับไปแทนโดยจะถูกต่อไปเข้ากับสัญญาณ DSR Input ของอุปกรณ์ด้าน Host หรือคอมพิวเตอร์

-Pin 7 เป็นขาสัญญาณ RTS (Request to Send) ของโมดูล SIM300CZ ซึ่งเป็น Input ของ SIM300CZ ซึ่งตามปรกติจะต่อเข้ากับ RTS ของอุปกรณ์ด้าน Host หรือคอมพิวเตอร์

-Pin 8 เป็นขาสัญญาณ CTS (Clear to Send) ของโมดูล SIM300CZ ซึ่งเป็น Output จาก SIM300CZ ซึ่งตามปรกติจะต่อเข้ากับ CTS ของอุปกรณ์ด้าน Host หรือคอมพิวเตอร์

-Pin 9 เป็นขาสัญญาณ RI (Ring Indicator) ของโมดูล SIM300CZ ซึ่งเป็น Output จาก SIM300CZซึ่งตามปรกติจะต่อเข้ากับ RI ของอุปกรณ์ด้าน Host หรือคอมพิวเตอร์

ตารางที่ 2.2 แสดงการต่อสายสัญญาณระหว่าง ET-GSM SIM300CZ กับคอมพิวเตอร์

DB9 Female(SIM300CZ)		Signal Direction	DB9 Male(Computer PC)	
Pin	Signal		Signal	Pin
1	DCD	→	DCD	1
2	TXD	→	RXD	2
3	RXD	←	TXD	3
4	DTR	←	DTR	4
5	GND	—	GND	5
6	(DSR)	→	DSR	6
7	RTS	←	RTS	7
8	CTS	→	CTS	8
9	RI	→	RI	9

ตารางที่ 2.3 แสดงการต่อสายสัญญาณระหว่าง ET-GSM SIM300CZ กับไมโครคอนโทรลเลอร์

DB9 Female(SIM300CZ)		Signal Direction	ไมโครคอนโทรลเลอร์
Pin	Signal		Signal
2	TXD	→	RXD
3	RXD	←	TXD
5	GND	—	GND

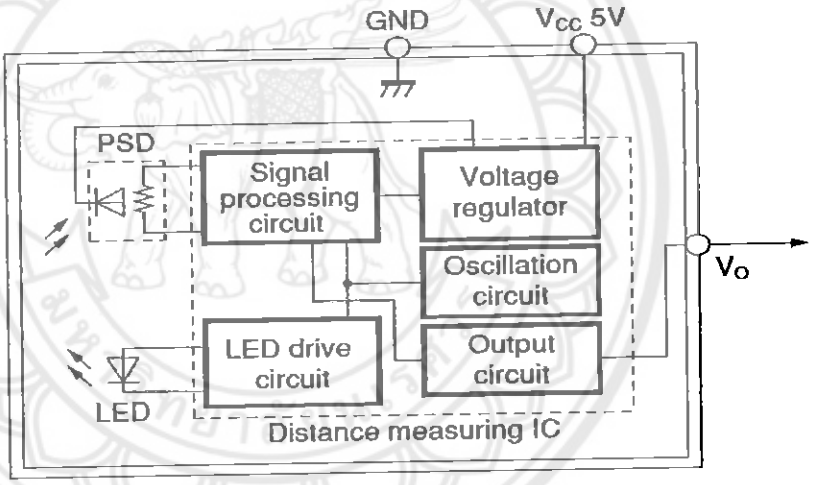
2.4 อุปกรณ์ตรวจวัดระยะทาง GP2Y0A02YK

2.4.1 คุณสมบัติ

-สีผิวของวัตถุต่างๆจะมีผลน้อยมากต่อการสะท้อนสัญญาณของอุปกรณ์ตรวจวัดระยะทาง จะมีผลบ้างในวัตถุที่มีสีดำ ซึ่งเป็นสีที่ทำให้การสะท้อนของสัญญาณอินฟราเรดทำได้ไม่ดีนัก

-ระยะทางในการตรวจจับวัตถุ จะมีหลายขนาดขึ้นอยู่กับตัวอุปกรณ์แต่ละตัวดังนั้นในการเลือกใช้ควรพิจารณาในเรื่องของระยะทางที่เราต้องการใช้งานโดยในเบอร์ GP2Y0A02YK จะให้สัญญาณขาออกออกมาเป็นแรงดัน ซึ่งเป็นอัตราส่วนกับระยะทาง โดยมีระยะทางในการตรวจจับมีความยาว 20 ถึง 150 เซนติเมตร

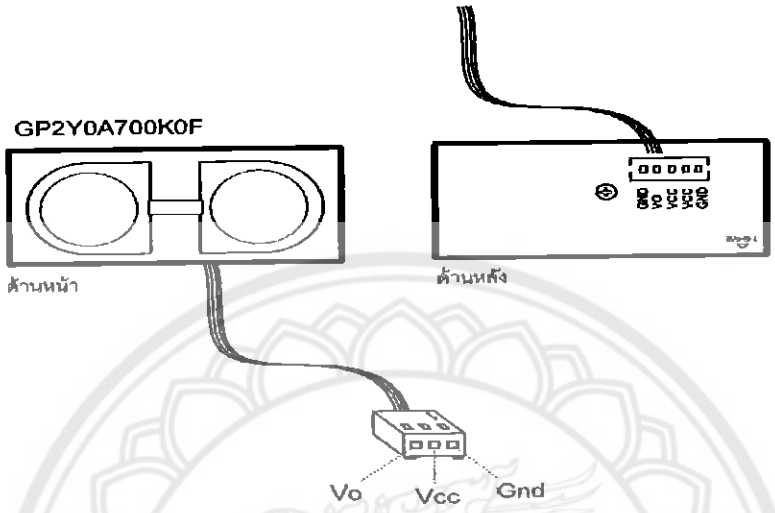
-ไม่จำเป็นต้องมีวงจรควบคุมภายนอกเพียงแค่จ่ายไฟเลี้ยง (VCC,GND) ก็สามารถนำสัญญาณขาออกของอุปกรณ์ตรวจวัดระยะทางไปใช้งานได้เลย ซึ่งแรงดัน Vcc ที่ใช้จะอยู่ในช่วง 4.5 ถึง 5.5V โดยปกติจะใช้ที่ 5 V แต่สามารถทนแรงดันต่ำสุดและสูงได้ -0.3 ถึง +7 V



รูปที่ 2.14 แผนภูมิรูปภาพภายในของ อุปกรณ์ตรวจวัดระยะทาง **GP2Y0A02K**

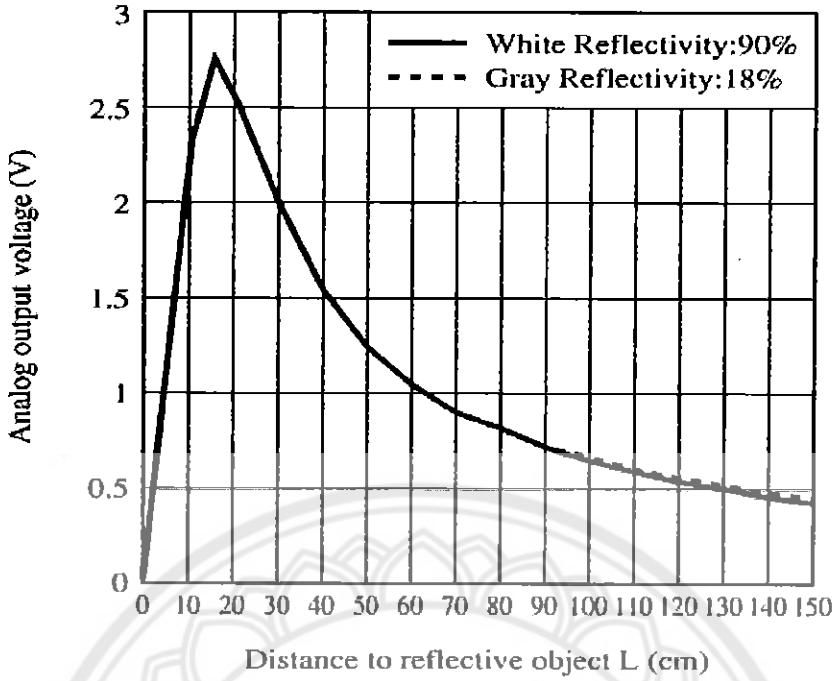
2.4.2 การต่อใช้งาน

จะมีขาสัญญาณสำหรับใช้งาน 3 ขา คือ VCC, GND และ ขาสัญญาณ Vo (Analog Output) มีการจัดเรียงในลักษณะต่อไปนี้



รูปที่ 2.15 รูปการต่อขาสัญญาณ Vo (Analog Output)

เนื่องจากสัญญาณขาออกของอุปกรณ์ตรวจวัดระยะทางจะได้ออกมาเป็นสัญญาณ Analog to Digital Convertor หรืออาจนำเอาสัญญาณขาออกของอุปกรณ์ตรวจวัดระยะทางมาต่อเข้ากับวงจรเปรียบเทียบแรงดัน (Comparator) แล้วให้มีสภาวะสัญญาณขาออกออกมาเป็น "0" (0 V) หรือ "1" (+5V) ก็ได้เช่นกัน



รูปที่ 2.16 Analog Output Voltage vs. Distance Reflective Object

จากกราฟจะเป็นการเปรียบเทียบของแรงดันกับระยะทางของอุปกรณ์ตรวจวัดระยะทาง จะเห็นได้ว่าจากทฤษฎีข้างต้นจะประมาณการวัดระยะทางได้สูงสุดที่ 150 ซมแต่ช่วงสูงสุดจะอยู่ที่ประมาณ 10-20 ซม ซึ่งเท่ากับมีค่าแรงดันเท่ากับ 2.8 โวลต์

บทที่ 3

วิธีการออกแบบ

ในบทที่ผ่านมาเป็นการศึกษาทฤษฎีและหลักการทำงานของอุปกรณ์ รวมไปถึงการเชื่อมต่อกับแผงวงจรไมโครคอนโทรเลอร์และแผงวงจรโทรศัพท์จากการศึกษา บทนี้จะนำมาประยุกต์เป็นโครงงานที่สามารถนำมาใช้ผลิตอุปกรณ์ที่สามารถใช้งานได้จริง

ขั้นตอนการดำเนินงานในบทนี้จะเป็นการออกแบบการวางโครงสร้างของอุปกรณ์และการศึกษาบทบาทหน้าที่ ทดลองการทำงานของอุปกรณ์ต่างๆที่นำมาใช้ในชิ้นงานนี้โดยทำการทดลองอุปกรณ์แบ่งเป็นขั้นตอน เริ่มจากส่วนแรก รับสัญญาณจากปริมาณความสูงของน้ำ โดยใช้อุปกรณ์ตรวจวัดระยะทาง(GP2Y0A700K0F) ส่วนของการควบคุมและการประมวลผลนั้นจะใช้แผงวงจรไมโครคอนโทรลเลอร์ส่วนส่งสัญญาณเตือนภัยน้ำท่วมผ่านทางระบบข้อความผ่านทางโทรศัพท์โดยใช้แผงวงจรโทรศัพท์และการออกแบบโครงสร้างชิ้นงานที่สามารถนำไปติดตั้งและใช้งานได้ในสถานที่จริง

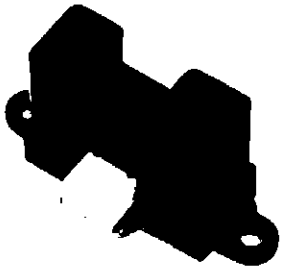
3.1 ส่วนควบคุมและประมวลผล

ในส่วนด้านการทำงานของส่วนควบคุมและประมวลผลนี้จะใช้แผงวงจรไมโครคอนโทรลเลอร์เบอร์ ATMEGA 328 โดยมีหลักการทำงาน คือ รับสัญญาณขาออก เป็นข้อมูลขาเข้า จากอุปกรณ์ตรวจวัดระยะทาง (GP2Y0A700K0F) เข้าแผงวงจรไมโครคอนโทรลเลอร์ที่ขา ADC (Analog-Digital Converter) แล้วเปลี่ยนสัญญาณจากอนาล็อกเป็นดิจิตอลแล้วทำการส่งสัญญาณขาออกไปยังแผงวงจรโทรศัพท์ (ET-GSM SIM300CZ) โดยรับข้อมูลผ่านทางRS 232

รูปที่ 3.1 แผงวงจรไมโครคอนโทรลเลอร์เบอร์ATMEGA 328

ห้องสมุดคณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

3.2 ส่วนรับสัญญาณจากปริมาณความสูงของน้ำ



รูปที่ 3.2 อุปกรณ์ตรวจวัดระยะทาง

โดยอุปกรณ์ที่เลือกใช้เป็นอุปกรณ์ตรวจวัดระยะทาง (GP2Y0A700K0F) สามารถตรวจจับวัตถุได้ในระยะ 20-150 เซนติเมตร ให้ค่าข้อมูลขาออกเป็นแรงดัน (Analog Voltage) และสามารถนำมาใช้ได้โดยการเทียบค่าจากตารางคู่มือ ดังนั้นจึงนำอุปกรณ์ตรวจวัดระยะทางชนิดนี้มาใช้ในการวัดระดับของน้ำและทำการส่งค่าแรงดันที่วัดได้แล้วเปลี่ยนค่าจากแรงดันเป็นระยะทาง ส่งไปยังแผงวงจรไมโครคอนโทรลเลอร์

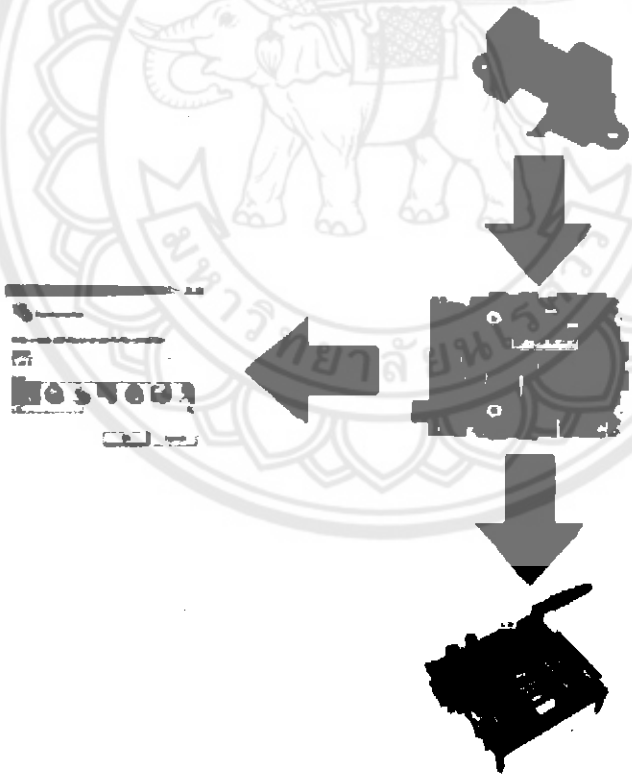
3.3 ส่วนส่งสัญญาณเตือนภัยน้ำท่วมผ่านทางระบบข้อความผ่านทางโทรศัพท์

รูปที่ 3.3 แผงวงจรโทรศัพท์ ET-GSM SIM300CZ V1.0

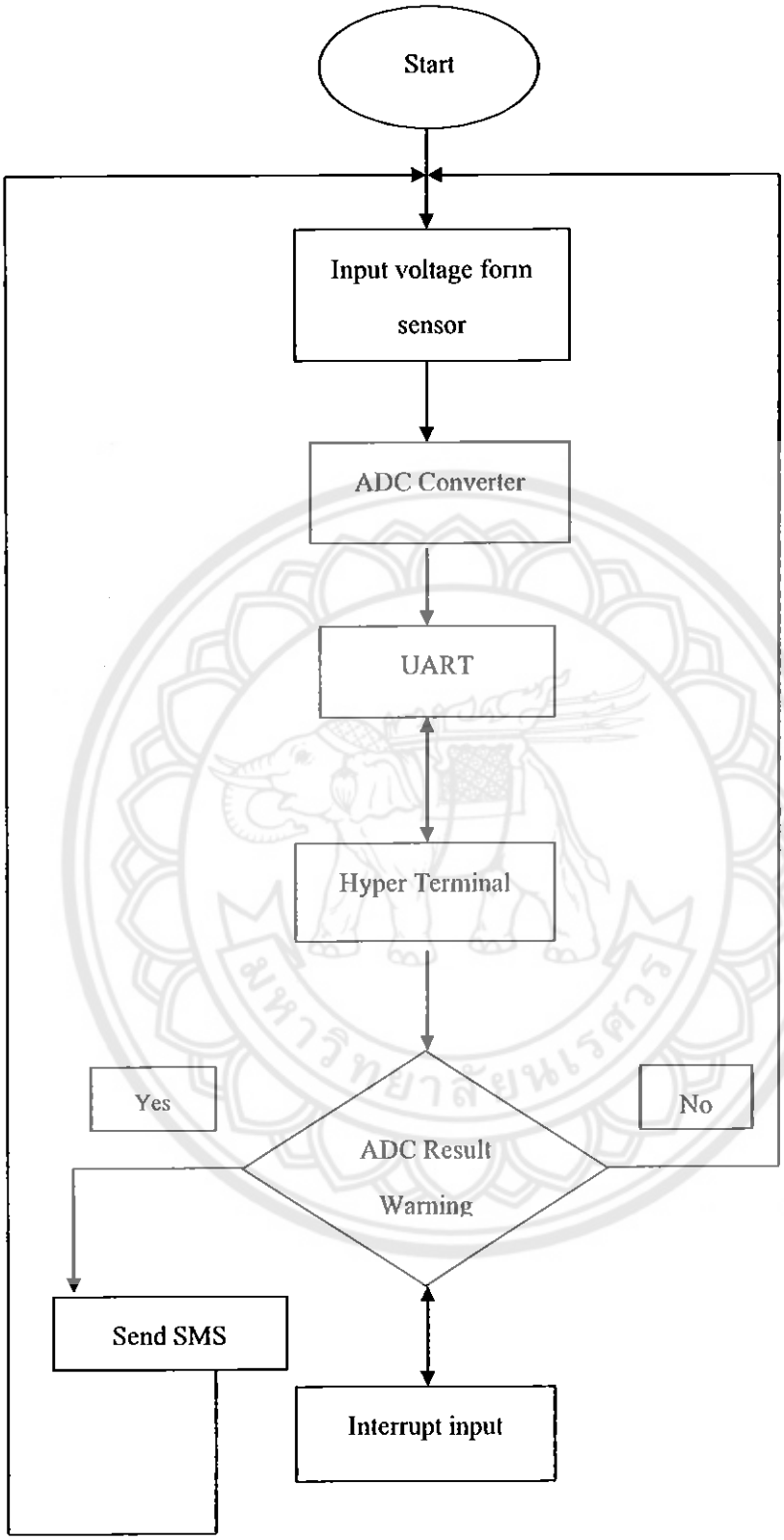
แผงวงจรโทรศัพท์เมื่อได้รับคำสั่งจากแผงวงจรไมโครคอนโทรลเลอร์ (ATMEGA328) โดยผ่านทาง RS232 จากนั้นจะทำการประมวลผลและส่งคำสั่งให้ส่งข้อความตัวอักษร (SMS) ไปยังหมายเลขโทรศัพท์ที่ได้กำหนดไว้ในตัวโปรแกรม

การทำงานของเครื่องสัญญาณเตือนภัยน้ำท่วมผ่านทางโทรศัพท์มือถือ

แผงวงจรไมโครคอนโทรลเลอร์เบอร์ (ATMEGA328) จะทำหน้าที่ประมวลผลรับข้อมูลจากตัวของอุปกรณ์ตรวจวัดระยะทางและส่งข้อมูลผ่าน RS232 เข้าแผงวงจรไมโครคอนโทรลเลอร์เบอร์ (ATMEGA328) เพื่อเปลี่ยนค่าจากอนาล็อกเป็นดิจิตอลแล้วส่งค่าไปแสดงผลยังหน้าจอ LCD และหน้าโปรแกรม HyperTerminal จากนั้นเราสามารถกำหนดค่าเบอร์โทรศัพท์โดยกดปุ่ม Interrupt เพื่อเปลี่ยนเป็นโหมดการกำหนดค่าโทรศัพท์ และเมื่อมีค่าต่ำกว่าเกณฑ์หรือมากกว่าเกณฑ์ที่กำหนดในโปรแกรม จะทำให้ไมโครคอนโทรลเลอร์สั่งงานให้แผงวงจรโทรศัพท์ (ET-GSM SIM300CZ V1.0) ทำการจะส่งข้อมูลเตือนภัยไปยังหมายเลขโทรศัพท์มือถือที่ตั้งไว้ในโปรแกรม โดยแผงวงจรโทรศัพท์ (ET-GSM SIM300CZ V1.0) จะทำงานได้ ต้องทำการเปลี่ยนสาย RS232 ในการเชื่อมต่อแผงวงจรไมโครคอนโทรลเลอร์ ไปแทนที่สาย RS232 ที่ทำหน้าที่ต่อกับโปรแกรม HyperTerminal โดยการทำงานของอุปกรณ์ตรวจวัดระยะทางจะแสดงค่าผ่านทางหน้าจอ LCD เป็นระยะเวลาอย่างต่อเนื่อง



รูปที่ 3.4 การทำงานของเครื่องเตือนภัยน้ำท่วมผ่านโทรศัพท์มือถือ



รูปที่ 3.5 การทำงานของเครื่องเตือนภัยน้ำท่วมผ่านโทรศัพท์มือถือ

บทที่4

ผลการทดลอง

หลังจากที่ได้นำหลักการของทฤษฎีในบทที่ 2 นำมาประยุกต์ใช้ออกแบบสร้างเป็นโครงงานแล้ว ในบทนี้จึงมุ่งเน้นที่จะศึกษาทดสอบการทำงานของแผงวงจรไมโครคอนโทรลเลอร์ที่รับข้อมูลมาจากอุปกรณ์ตรวจวัดระยะทางและส่งข้อมูลผ่านทางโทรศัพท์มือถือว่า สามารถใช้งานได้จริงดังที่ได้ออกแบบไว้ในบทที่ 3 โดยจัดแบ่งการทดลองออกเป็นหัวข้อต่างๆดังต่อไปนี้

4.1 ระบบการทำงาน

สำหรับการทำงานสามารถแบ่งออกเป็น 2 ส่วน ได้แก่ ส่วนแรก รับค่าจากอุปกรณ์ตรวจวัดระยะทางแล้วทำการส่งข้อความเข้าโทรศัพท์มือถือ และส่วนที่สอง การกำหนดหมายเลขเบอร์โทรศัพท์มือถือ

ระบบการทำงานส่วนที่รับค่าจากอุปกรณ์ตรวจวัดระยะทางประกอบไปด้วย ระบบไมโครคอนโทรลเลอร์ ที่รับสัญญาณมาจากอุปกรณ์ตรวจวัดระยะทางแล้วแสดงผลผ่านทางหน้าจอ LCD จากรูปด้านล่างนั้นจะแสดงสัญญาณเตือนภัยที่รับมาจากอุปกรณ์ตรวจวัดระยะทาง

รูปที่ 4.1 รูปแสดงสัญญาณเตือนภัยที่รับมาจากอุปกรณ์ตรวจวัดระยะทาง

รูปที่ 4.2 อุปกรณ์การทำงานของระบบเตือนภัย

ตารางที่ 4.1 ค่าที่วัดจากอุปกรณ์ตรวจวัดระยะทาง

ระยะทาง[ซม.]	โปรแกรม Visual Basic 6.0		
	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3
15	35	32	33
20	45	44	44
25	50	52	51
30	53	50	49
35	81	83	93
40	90	93	93
45	114	116	116
50	118	126	114
55	146	148	150
60	145	149	148

ตารางที่ 4.2 ค่าที่วัดจากโวลต์มิเตอร์

ระยะทาง[ซม.]	Voltmeter		
	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3
15	0.68	0.63	0.64
20	0.88	0.86	0.87
25	0.98	1.02	1.00
30	1.04	0.98	0.96
35	1.58	1.62	1.82
40	1.76	1.82	1.82
45	2.23	2.27	2.27
50	2.31	2.46	2.27
55	2.85	2.89	2.93
60	2.83	2.91	2.89

จากตารางที่ 4.1 และตารางที่ 4.2 จะเปรียบเทียบค่าที่วัดจากอุปกรณ์ตรวจวัดระยะทางกับค่าที่วัดได้จากโวลต์มิเตอร์ซึ่งเป็นไปตามทฤษฎีในบทที่ 2 ทั้งนี้ค่าที่วัดได้จากอุปกรณ์ตรวจวัดระยะทาง จะเป็นค่าที่เป็นแรงดันบิตจะต่างจากค่าที่วัดได้จากโวลต์มิเตอร์ จากทฤษฎีการคำนวณแรงดันขาเข้า จะได้ดังนี้

$$ADC = \frac{Vin.Vbit}{Vref}$$

ADC= แรงดันที่แปลงจากอนาล็อกเป็นดิจิตอล

Vin =แรงดันด้านขาอินพุต

Vref =แรงดันอ้างอิงของเซนเซอร์ ค่าเท่ากับ +5V

Vbit =แรงดันบิตของเซนเซอร์มีความละเอียดขนาด8บิต มีค่าเท่ากับ 0-256

ยกตัวอย่าง วัดระยะทางเซนเซอร์ที่ 15 cmได้เท่ากับ 35 v ในตารางที่ 4.1

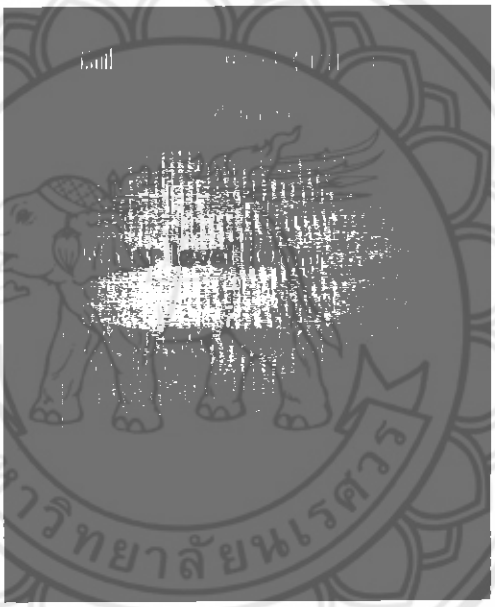
$$Vin = \frac{35.5}{256} = 0.684$$

จากการคำนวณจะได้ค่าที่ใกล้เคียงกับที่โวลต์มิเตอร์วัดได้ในตารางที่ 4.2 = 0.684 v ซึ่งมีระยะทางอยู่ 15 cm

เพื่อเพิ่มประสิทธิภาพในการอ่านค่าของเซนเซอร์ เนื่องจากค่าที่อ่านออกมาได้นั้น เกิดความคลาดเคลื่อน เราจึงนำเอาสูตร Moving everage filter มาใช้ในการเขียนโปรแกรมเพื่อให้เกิดความเสถียรต่อโปรแกรม

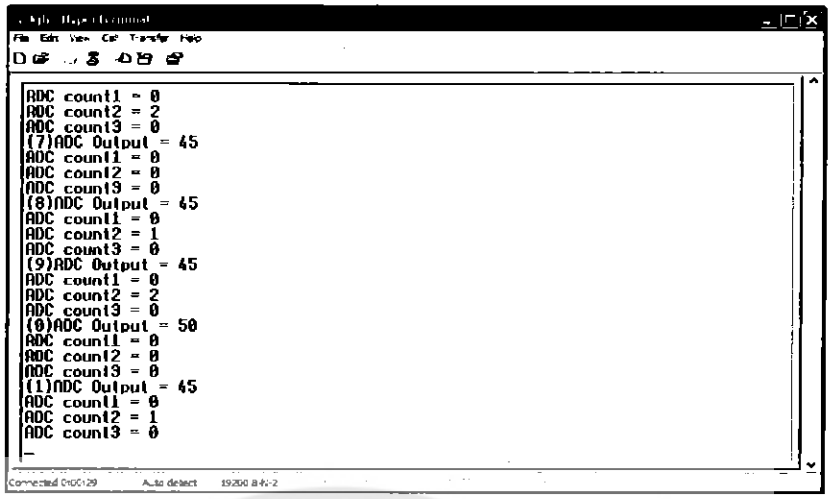
$$X=0.9(Xold) + 0.1(Xnew)$$

แผงวงจรไมโครคอนโทรลเลอร์ ทำหน้าที่ ควบคุมการสั่งงานของระบบ โดยรับค่าอินพุตเข้ามาแปลงเป็นค่าตัวเลขเก็บไว้และทำการตัดสินใจตามเงื่อนไข เมื่อมีค่าตามเกณฑ์ที่กำหนดไว้ภายในตัวโปรแกรม จะทำให้ส่งคำสั่งเตือนภัยผ่านทางผ่านพอร์ต RS232 ที่เชื่อมต่อกับแผงวงจรโทรศัพท์ส่งข้อความออกไปยังหมายเลขโทรศัพท์ปลายทางที่บันทึกไว้



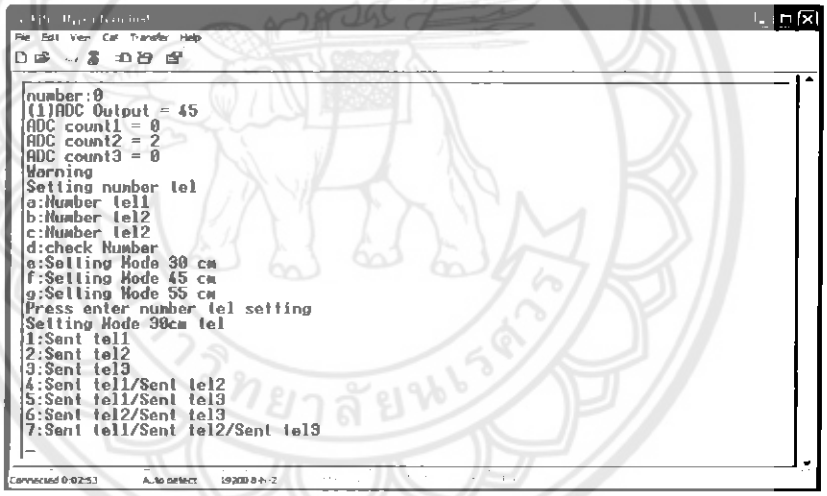
รูปที่ 4.3 แสดงผลการส่งข้อความเตือนภัยระดับน้ำ

โปรแกรม HyperTerminal ทำหน้าที่รับคำสั่ง จากแผงวงจรไมโครคอนโทรลเลอร์ ทำการแสดงผลการรับค่าสัญญาณขาเข้าออกทางหน้าจอคอมพิวเตอร์เหมือนเดียวกันกับที่แสดงค่าในจอ LCD และที่สำคัญคือทำหน้าที่ตั้งค่าเบอร์โทรศัพท์ที่เราจะทำการส่งข้อความ โดยใช้ปุ่ม Interrupt ในการเปลี่ยนโหมด เบอร์โทรศัพท์โดยไม่จำเป็นต้องทำการโปรแกรมเข้าไปใหม่สามารถบันทึกค่าได้ โดยข้อมูลที่บันทึกไว้จะถูกเก็บไว้ใน EEPROM ของแผงวงจรไมโครคอนโทรลเลอร์



รูปที่ 4.4 แสดงสัญญาณเตือนภัยที่รับมาจากอุปกรณ์ตรวจวัดระยะทางใน Hyperterminal

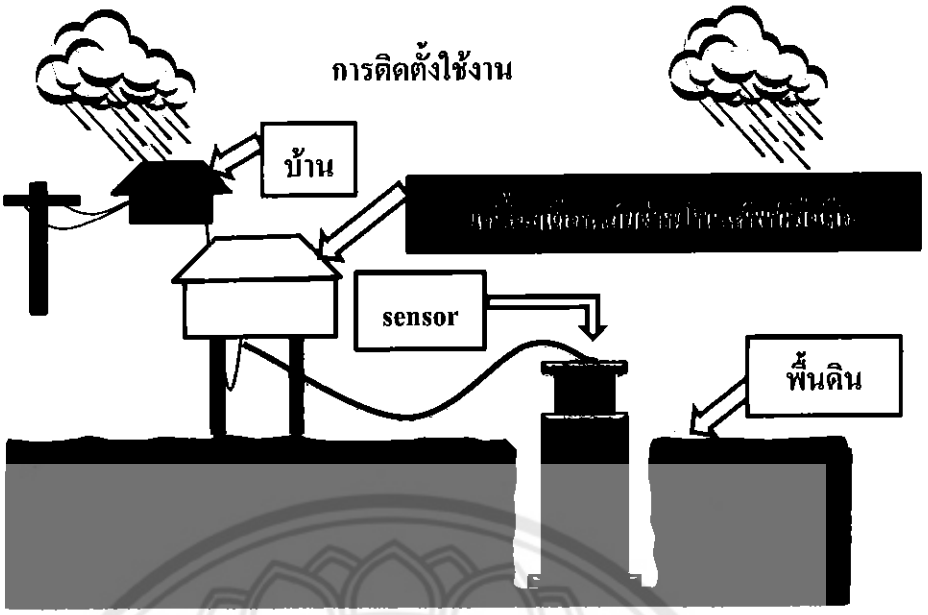
ในโหมดของการตั้งค่าหมายเลขโทรศัพท์เราสามารถตั้งค่าได้ 3 หมายเลข



รูปที่ 4.5 แสดงการตั้งค่าหมายเลขโทรศัพท์ที่เราจะทำการส่งข้อความ

4.2 ผลการทดลอง

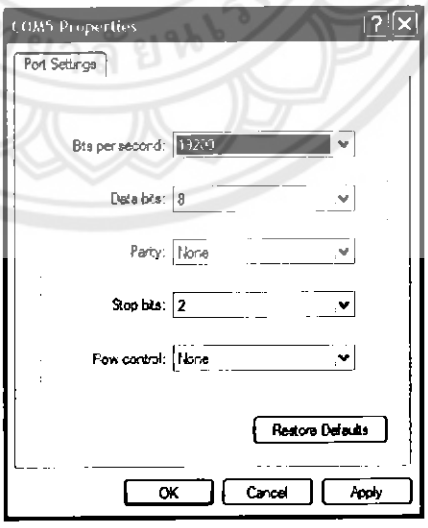
ทำการขุดหลุมดินให้ลึกพอดีกับระดับที่น้ำจะท่วม เพราะว่าการวัดระดับน้ำท่วม เกิดขึ้นเมื่อน้ำฝนไหลเข้าหลุมดินและมีปริมาณเพิ่มขึ้น จากนั้นแผงวงจรไมโครคอนโทรลเลอร์จะทำการเก็บข้อมูลของระดับน้ำที่ส่งเข้ามา เมื่อข้อมูลที่รับเข้ามามีค่าตรงกับระยะที่กำหนดไว้จะมีคำสั่งให้ไมโครคอนโทรลเลอร์ส่งข้อความเตือนภัยไปยังโทรศัพท์มือถือ



รูปที่ 4.6 แผนภาพการติดตั้งการใช้งาน

4.2.1 การกำหนดค่าการทำงานของเครื่องเตือนภัยน้ำท่วม

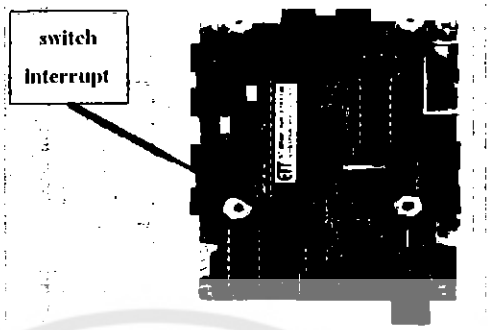
การกำหนดค่าต่างๆ ของเครื่องแจ้งเตือนภัยสามารถแก้ไขผ่าน Hyper Terminal ได้ โดยกำหนดค่าของระบบการสื่อสารดังนี้ 19200 b/sec, 8 bit, None parity bit, Stop bit 2, flow control None



รูปที่ 4.7 กำหนดค่าของระบบการสื่อสารใน Hyper Terminal

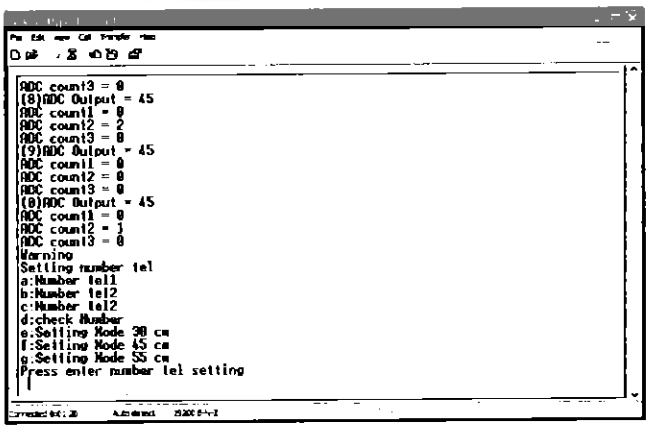
4.2.2 การแก้ไขหมายเลขโทรศัพท์

กด Switch interrupt ที่แผงวงจรไมโครคอนโทรลเลอร์ สามารถเข้าโหมดการเปลี่ยนหมายเลขโทรศัพท์



รูปที่ 4.8 กด Switch Interrupt เพื่อเข้าโหมดเปลี่ยนเบอร์โทรศัพท์

1. กดปุ่ม a บนคีบอร์ดสามารถเลือกเปลี่ยนหรือแก้ไขหมายเลขโทรศัพท์ 1
2. กดปุ่ม b บนคีบอร์ดสามารถเลือกเปลี่ยนหรือแก้ไขหมายเลขโทรศัพท์ 2
3. กดปุ่ม c บนคีบอร์ดสามารถเลือกเปลี่ยนหรือแก้ไขหมายเลขโทรศัพท์ 3
4. กดปุ่ม d บนคีบอร์ดสามารถตรวจสอบหมายเลขที่ตั้งไว้ได้ทั้ง 3 หมายเลข
5. กดปุ่ม e บนคีบอร์ดเข้าโหมดการตั้งค่าการส่งข้อความไปยังกลุ่มเป้าหมายที่ระดับน้ำ 30 เซนติเมตร
6. กดปุ่ม f บนคีบอร์ดเข้าโหมดการตั้งค่าการส่งข้อความไปยังกลุ่มเป้าหมายที่ระดับน้ำ 45 เซนติเมตร
7. กดปุ่ม g บนคีบอร์ดเข้าโหมดการตั้งค่าการส่งข้อความไปยังกลุ่มเป้าหมายที่ระดับน้ำ 55 เซนติเมตร



รูปที่ 4.9 โหมดการตั้งค่าเบอร์โทรศัพท์และเลือกระดับการส่งข้อความ

4.2.3 การกำหนดส่งข้อความแจ้งเตือนผู้ใช้

1.กดปุ่ม 1 ทำการส่งข้อความแจ้งเตือนให้หมายเลขโทรศัพท์ที่ 1

2.กดปุ่ม 2 ทำการส่งข้อความแจ้งเตือนให้หมายเลขโทรศัพท์ที่ 2

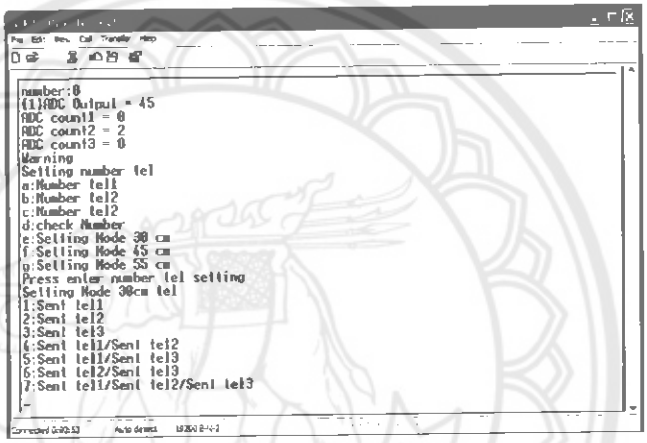
3.กดปุ่ม 3 ทำการส่งข้อความแจ้งเตือนให้หมายเลขโทรศัพท์ที่ 3

4.กดปุ่ม 4 ทำการส่งข้อความแจ้งเตือนให้หมายเลขโทรศัพท์ที่ 1 , 2

5.กดปุ่ม 5 ทำการส่งข้อความแจ้งเตือนให้หมายเลขโทรศัพท์ที่ 1 , 3

6.กดปุ่ม 6 ทำการส่งข้อความแจ้งเตือนให้หมายเลขโทรศัพท์ที่ 2 , 3

7.กดปุ่ม 7 ทำการส่งข้อความแจ้งเตือนให้หมายเลขโทรศัพท์ที่ 1 , 2 , 3



รูปที่ 4.10 การกำหนดเบอร์โทรศัพท์ส่งข้อความแจ้งเตือนผู้ใช้

4.2.4 การทำงานแจ้งเตือนข้อความ

1. ระดับที่กำหนดค่าไว้ที่ 30 cm

ถ้าปริมาณน้ำเพิ่มขึ้นถึงระดับ 30 cm จะแจ้งเตือน "The water level upper to 30 cm"

แต่ปริมาณน้ำลดลงถึงระดับ 30 cm จะแจ้งเตือน "The water level lower to 30 cm"

2. ระดับที่กำหนดค่าไว้ที่ 45 cm

ถ้าปริมาณน้ำเพิ่มขึ้นถึงระดับ 45 cm จะแจ้งเตือน "The water level upper to 45 cm"

แต่ปริมาณน้ำลดลงถึงระดับ 45 cm จะแจ้งเตือน "The water level lower to 45 cm"

3. ระดับที่กำหนดค่าไว้ที่ 55 cm

ถ้าปริมาณน้ำเพิ่มขึ้นถึงระดับ 55 cm จะแจ้งเตือน "The water level upper to 55 cm"

แต่ปริมาณน้ำลดลงถึงระดับ 55 cm จะแจ้งเตือน "The water level lower to 55 cm"

บทที่ 5
บทสรุป

โครงงานนี้ศึกษาและพัฒนาระบบสัญญาณเตือนภัยน้ำท่วมผ่านทางสัญญาณโทรศัพท์มือถือ ซึ่งนำหลักการการส่งสัญญาณข้อมูลที่รวดเร็วและมีมาตรฐานดียิ่งขึ้นกว่าเดิม แล้วสามารถวัดสัญญาณเพื่อนำมาทำการวิเคราะห์ข้อมูลว่าสัญญาณที่รับเข้ามาเป็นค่าที่น่าเชื่อถือหรือไม่ ดังนั้นสามารถนำข้อมูลมาแปลงเป็นค่าที่สามารถนำไปแสดงผล เพื่อใช้ประโยชน์ในส่วนนี้ต่อไป

เมื่อได้ทำการทดลองระบบสัญญาณเตือนภัยน้ำท่วมผ่านทางสัญญาณโทรศัพท์ ขณะที่ได้ทำการทดลองโครงงาน ผลที่ได้ทำให้ทราบข้อมูลจากการพัฒนาระบบในครั้งนี้ ทราบถึงปัญหาขณะการดำเนินงานและการใช้งานของอุปกรณ์ในบางประการ จึงทำให้สามารถสรุปผลของโครงงานนี้แบ่งออกเป็นส่วนๆ ดังนี้

5.1 สรุปผลการทดลอง

โครงงานนี้ได้ศึกษาและพัฒนาระบบสัญญาณเตือนภัยน้ำท่วมผ่านทางสัญญาณโทรศัพท์มือถือ ด้วยการใช้พื้นฐานทางด้านภาษา C ในการเขียนโปรแกรม AVR Studio4 ควบคุมการทำงานระบบสัญญาณเตือนภัยน้ำท่วมผ่านทางสัญญาณโทรศัพท์มือถือ ใช้แผงวงจรไมโครคอนโทรลเลอร์ (ATMEGA 328) ในการรับค่าจากอุปกรณ์วัดระยะทางมาแปลงค่าจากค่าอนาล็อกเป็นค่าดิจิตอลเพื่อส่งค่าที่ได้เป็นแสดงผลในคอมพิวเตอร์ การรับค่าที่มาจากแผงวงจรไมโครคอนโทรลเลอร์ (ATMEGA 328) โดยผ่าน RS232 มาแสดงโดยใช้พื้นฐานภาษา Cในการออกแบบแสดงค่าผ่านทางหน้าจอ LCD และแสดงค่าผ่านทางโปรแกรม Hyper Terminal การรับค่าแต่ละค่าจะนำค่าที่ได้มาเทียบกับค่าที่ได้ตั้งไว้ค่าสัญญาณที่ผ่านเข้ามาแต่ละครั้งใช้ฟังก์ชันรีเลย์การหน่วงเวลาในการรับค่าสัญญาณ เมื่อมีค่าเป็นไปตามเงื่อนไขแล้วจะทำการส่งสัญญาณออกแสดงผลทางหน้าจอ LCD และยังสั่งให้ส่งสัญญาณออกผ่านทาง RS232 เข้าสู่แผงวงจรโทรศัพท์ (ET-GSM SIM300CZ V1.0) เพื่อทำการส่งข้อความเตือนภัยไปยังหมายเลขโทรศัพท์ที่ตั้งไว้

ผลจากการพัฒนาระบบสัญญาณเตือนภัยน้ำท่วมผ่านทางสัญญาณโทรศัพท์มือถือด้วยการใช้ภาษา C ในการออกแบบการเขียนโปรแกรม ทำให้ผลที่ได้ออกมามีความน่าเชื่อถือมากขึ้น และติดต่อกับแผงวงจรโทรศัพท์ (ET-GSM SIM300CZ V1.0) เพื่อส่งข้อความเตือนภัยและทราบถึงระดับน้ำที่เพิ่มขึ้นหรือลดลง เพื่อเป็นประโยชน์ในการรับข้อมูลอย่างสะดวกและรวดเร็ว

5.2 ปัญหาในการทำงานและแนวทางแก้ไข

จากการทำโครงงานระบบสัญญาณเตือนภัยน้ำท่วมผ่านทางสัญญาณโทรศัพท์มือถือพบกับปัญหาและอุปสรรค์ต่างๆดังนี้

1. การดำเนินงานมีความล่าช้า เนื่องจากต้องทำการศึกษาตัวอุปกรณ์แต่ละตัวภายในแผงวงจรไมโครคอนโทรลเลอร์ และแผงวงจรโทรศัพท์หาวิธีแก้ไขทำการศึกษาข้อมูลของ ETT อย่างละเอียดและสอบถามผู้เชี่ยวชาญในด้านนี้

2. การดำเนินงานในการพัฒนาโปรแกรม บางครั้งเกิดปัญหาข้อผิดพลาดในด้านการพัฒนาด้านโปรแกรมทำให้เกิดความล่าช้าในการดำเนินงาน วิธีแก้ไขการศึกษาด้วยตัวเองและสอบถามผู้เชียวชาญในด้านนี้

5.3 ข้อเสนอแนะและแนวทางสำหรับการพัฒนา

1. ในอนาคตระบบการสื่อสารจะมีบทบาทในชีวิตประจำวันมาก การพัฒนาระบบที่เกี่ยวข้องน่าช่วยให้มีประโยชน์อย่างมากเช่นกัน

2. เพิ่มอุปกรณ์รับและส่งข้อมูลเพิ่มเติมเพื่อให้ระบบมีความยืดหยุ่นและสะดวกต่อการใช้งานมากขึ้น

3. เพิ่มการพัฒนาทางด้านโปรแกรมอื่นเข้ามาช่วยในการทำงานเพื่อให้งานมีความสามารถในการทำงานหลากหลายมากขึ้น

เอกสารอ้างอิง

[1] ประภาพร ช่างไม้, คู่มือการเขียนโปรแกรมภาษา C ฉบับผู้เริ่มต้น, กรุงเทพมหานคร : Infopress Developer Book, 2545

[2] การเขียนโปรแกรมควบคุมไมโครคอนโทรลเลอร์ AVR ด้วยภาษา C กับ WinAVR (C Compiler) เบื้องต้น เล่ม 1, กรุงเทพมหานคร : Appsofttech, 2550

[3] การเขียนโปรแกรมควบคุมไมโครคอนโทรลเลอร์ AVR ด้วยภาษา C กับ WinAVR (C Compiler) ประยุกต์ เล่ม 2, กรุงเทพมหานคร : Appsofttech, 2550

[4] พวกเราชาวอีทีที, www.ett.co.th , [Online], Available: http://www.ett.co.th/comparison/ETT-Comparision%20Chart-AVR-v2.pdf

ภาคผนวก

```
#define F_CPU 19660800UL                    // 19.6608 MHz
#include <stdint.h>
#include <avr/io.h>
#include <util/delay.h>
#include <stdio.h>
#include <avr/eeprom.h>
#include <avr/interrupt.h>
#define BAUD 19200                          // 19200 BPS
#define MYUBRR F_CPU/16/BAUD-1
#define  PORT_74HC595_SIN PORTD             // Pin PD
#define  PORT_74HC595_STR PORTD             // Pin PD
#define  PORT_74HC595_CLK PORTB             // Pin PB
#define  DIR_74HC595_SIN  DDRD              // Din PD
#define  DIR_74HC595_STR  DDRD              // Dir PD
#define  DIR_74HC595_CLK  DDRB              // Dir PB
#define  SIN_74HC595_PIN  (1<<4)            // PD:Bit[4]
#define  STR_74HC595_PIN  (1<<7)            // PD:Bit[7]
#define  CLK_74HC595_PIN  (1<<0)            // PB:Bit[0]
#define  SIN_74HC595_HI() PORT_74HC595_SIN |=  SIN_74HC595_PIN
#define  SIN_74HC595_LO() PORT_74HC595_SIN &= ~SIN_74HC595_PIN
#define  STR_74HC595_HI() PORT_74HC595_STR |=  STR_74HC595_PIN
#define  STR_74HC595_LO() PORT_74HC595_STR &= ~STR_74HC595_PIN
#define  CLK_74HC595_HI() PORT_74HC595_CLK |=  CLK_74HC595_PIN
#define  CLK_74HC595_LO() PORT_74HC595_CLK &= ~CLK_74HC595_PIN
#define  SHIFT_BL    0x01                   // Q0 = Backlight(0=FF,1=ON)
#define  SHIFT_RS    0x02                   // Q1 = RS(0=Command,1=Data)
#define  SHIFT_RW    0x04                   //Q2 = RW(0=Write,1=Read)
#define  SHIFT_EN    0x08

int Backlight_LED;
char lcdbuf[16+1];
char uart_buf[24];                          // "sprint" UART[] Buffer
```

```
static char Index_Count;
char uart_buf_A0;
char uart_buf_A1;
char uart_buf_A2;
char uart_buf_A3;
char uart_buf_A4;
char uart_buf_A5;
char uart_buf_A6;
char uart_buf_A7;
char uart_buf_A8;
char uart_buf_A9;

char uart_buf_B0;
char uart_buf_B1;
char uart_buf_B2;
char uart_buf_B3;
char uart_buf_B4;
char uart_buf_B5;
char uart_buf_B6;
char uart_buf_B7;
char uart_buf_B8;
char uart_buf_B9;

char uart_buf_C0;
char uart_buf_C1;
char uart_buf_C2;
char uart_buf_C3;
char uart_buf_C4;
char uart_buf_C5;
char uart_buf_C6;
char uart_buf_C7;
char uart_buf_C8;
```

```
char uart_buf_C9;

char uart_buf_e0;
char uart_buf_e1;
char uart_buf_e2;

char uart_buf_f0;
char uart_buf_f1;
char uart_buf_f2;

uint16_t send_same1;
uint16_t send_same2;
uint16_t send_same3;
uint16_t send_1 =1;
uint16_t send_2 =1;
uint16_t send_3 =1;
uint16_t level ;
/* pototype  section */
void init_serial(unsigned int ubrr);              // Initil UART
int putchar0(unsigned char data);                 // Put Char To UART
unsigned char getchar0(void);                     // Get Char From UART
void print_uart(void);                            // Print String to UART
void LCD_Initial();                               // Initial LCD Display
void LCD_Command(unsigned char value);            // Write Command
void LCD_Ascii(unsigned char);
void LCD_Print_String(const char c[]);
void LCD_Print_Buffer(void);
void LCD_SetCursor(unsigned char Cursor);         // Set Cursor(Set DDRAM Address)
void LCD_ClearScreen(void);                       // Clear Screen Display
void LCD_Backlight(int LED_Status);
void LCD_Busy_LCD(int lcd_command);
```

```
void LCD_Shift_Byte(unsigned char value, int lcd_command);      // Shift Byte(Data,Command)
                                                                to LCD
void LCD_Shift_Nibble(unsigned char nibble, int lcd_command);  // Shift Nibble(4 Bit Data) to
                                                                LCD
void LCD_Shift_74HC595(unsigned char value);                    // Shift Byte to 74HC595
void delay_ms(unsigned int time);
void delay_us(unsigned int time);
void eeprom_write_byte1(uint16_t addr, uint8_t data)
{
 while(EECR & (1<<EEPE))          /*wait until previous write any*/
 ;
EEAR = addr;
EEDR = data;
EECR |= (1<<EEMPE);
EECR |= (1<<EEPE);
}
uint8_t eeprom_read_byte1(uint16_t addr)
{
 while(EECR & (1<<EEPE))          /*wait until previous write any*/
 ;
EEAR = addr;
EECR |= (1<<EERE);                              //disable eeprom ready interrupt
  EECR &= ~(1<<EERIE);
 return EEDR;
}
void chang_A()
{
uart_buf_f0 = getchar0();
sprintf(uart_buf,"%c",uart_buf_f0);
print_uart();
uart_buf_A0 = getchar0();
eeprom_write_byte1(0xC0,uart_buf_A0);
```

```
sprintf(uart_buf,"%c",uart_buf_A0);
print_uart();
uart_buf_A1 = getchar0();
eeprom_write_byte1(0xC1,uart_buf_A1);
sprintf(uart_buf,"%c",uart_buf_A1);
print_uart();
uart_buf_A2 = getchar0();
eeprom_write_byte1(0xC2,uart_buf_A2);
sprintf(uart_buf,"%c",uart_buf_A2);
print_uart();
uart_buf_A3 = getchar0();
eeprom_write_byte1(0xC3,uart_buf_A3);
sprintf(uart_buf,"%c",uart_buf_A3);
print_uart();
uart_buf_A4 = getchar0();
eeprom_write_byte1(0xC4,uart_buf_A4);
sprintf(uart_buf,"%c",uart_buf_A4);
print_uart();
uart_buf_A5 = getchar0();
eeprom_write_byte1(0xC5,uart_buf_A5);
sprintf(uart_buf,"%c",uart_buf_A5);
print_uart();

uart_buf_A6 = getchar0();
eeprom_write_byte1(0xC6,uart_buf_A6);
sprintf(uart_buf,"%c",uart_buf_A6);
print_uart();
uart_buf_A7 = getchar0();
eeprom_write_byte1(0xC7,uart_buf_A7);
sprintf(uart_buf,"%c",uart_buf_A7);
print_uart();
uart_buf_A8 = getchar0();
```

```
eeprom_write_byte1(0xC8,uart_buf_A8);
sprintf(uart_buf,"%c",uart_buf_A8);
print_uart();
uart_buf_A9 = getchar0();
eeprom_write_byte1(0xC9,uart_buf_A9);
sprintf(uart_buf,"%c",uart_buf_A9);
print_uart();
}
void chang_B()
{
uart_buf_f1 = getchar0();
sprintf(uart_buf,"%c",uart_buf_f1);
print_uart();
uart_buf_B0 = getchar0();
eeprom_write_byte1(0xD0,uart_buf_B0);
sprintf(uart_buf,"%c",uart_buf_B0);
print_uart();

uart_buf_B1 = getchar0();
eeprom_write_byte1(0xD1,uart_buf_B1);
sprintf(uart_buf,"%c",uart_buf_B1);
print_uart();
uart_buf_B2 = getchar0();
eeprom_write_byte1(0xD2,uart_buf_B2);
sprintf(uart_buf,"%c",uart_buf_B2);
print_uart();
uart_buf_B3 = getchar0();
eeprom_write_byte1(0xD3,uart_buf_B3);
sprintf(uart_buf,"%c",uart_buf_B3);
print_uart();
uart_buf_B4 = getchar0();
eeprom_write_byte1(0xD4,uart_buf_B4);
```

```
sprintf(uart_buf,"%c",uart_buf_B4);
print_uart();
uart_buf_B5 = getchar0();
eeprom_write_byte1(0xD5,uart_buf_B5);
sprintf(uart_buf,"%c",uart_buf_B5);
print_uart();
uart_buf_B6 = getchar0();
eeprom_write_byte1(0xD6,uart_buf_B6);
sprintf(uart_buf,"%c",uart_buf_B6);
print_uart();
uart_buf_B7 = getchar0();
eeprom_write_byte1(0xD7,uart_buf_B7);
sprintf(uart_buf,"%c",uart_buf_B7);
print_uart();
uart_buf_B8 = getchar0();
eeprom_write_byte1(0xD8,uart_buf_B8);
sprintf(uart_buf,"%c",uart_buf_B8);
print_uart();
uart_buf_B9 = getchar0();
eeprom_write_byte1(0xD9,uart_buf_B9);
sprintf(uart_buf,"%c",uart_buf_B9);
print_uart();
}
void chang_C()
{
uart_buf_f2 = getchar0();
sprintf(uart_buf,"%c",uart_buf_f2);
print_uart();
uart_buf_C0 = getchar0();
eeprom_write_byte1(0xE0,uart_buf_C0);
sprintf(uart_buf,"%c",uart_buf_C0);
print_uart();
```

```
uart_buf_C1 = getchar0();
eeprom_write_byte1(0xE1,uart_buf_C1);
sprintf(uart_buf,"%c",uart_buf_C1);
print_uart();
uart_buf_C2 = getchar0();
eeprom_write_byte1(0xE2,uart_buf_C2);
sprintf(uart_buf,"%c",uart_buf_C2);
print_uart();

uart_buf_C3 = getchar0();
eeprom_write_byte1(0xE3,uart_buf_C3);
sprintf(uart_buf,"%c",uart_buf_C3);
print_uart();
uart_buf_C4 = getchar0();
eeprom_write_byte1(0xE4,uart_buf_C4);
sprintf(uart_buf,"%c",uart_buf_C4);
print_uart();
uart_buf_C5 = getchar0();
eeprom_write_byte1(0xE5,uart_buf_C5);
sprintf(uart_buf,"%c",uart_buf_C5);
print_uart();
uart_buf_C6 = getchar0();
eeprom_write_byte1(0xE6,uart_buf_C6);
sprintf(uart_buf,"%c",uart_buf_C6);
print_uart();
uart_buf_C7 = getchar0();
eeprom_write_byte1(0xE7,uart_buf_C7);
sprintf(uart_buf,"%c",uart_buf_C7);
print_uart();
uart_buf_C8 = getchar0();
eeprom_write_byte1(0xE8,uart_buf_C8);
sprintf(uart_buf,"%c",uart_buf_C8);
```

```
print_uart();
uart_buf_C9 = getchar0();
eeprom_write_byte1(0xE9,uart_buf_C9);
sprintf(uart_buf,"%c",uart_buf_C9);
print_uart();
}
//read eeprom
void read_A()
{
uart_buf_A0 = eeprom_read_byte1(0xC0);
uart_buf_A1 = eeprom_read_byte1(0xC1);
uart_buf_A2 = eeprom_read_byte1(0xC2);
uart_buf_A3 = eeprom_read_byte1(0xC3);
uart_buf_A4 = eeprom_read_byte1(0xC4);
uart_buf_A5 = eeprom_read_byte1(0xC5);
uart_buf_A6 = eeprom_read_byte1(0xC6);
uart_buf_A7 = eeprom_read_byte1(0xC7);
uart_buf_A8 = eeprom_read_byte1(0xC8);
sprintf(uart_buf,"%c",uart_buf_A0);
print_uart();
sprintf(uart_buf,"%c",uart_buf_A1);
print_uart();
sprintf(uart_buf,"%c",uart_buf_A2);
print_uart();
sprintf(uart_buf,"%c",uart_buf_A3);
print_uart();
sprintf(uart_buf,"%c",uart_buf_A4);
print_uart();
sprintf(uart_buf,"%c",uart_buf_A5);
print_uart();
sprintf(uart_buf,"%c",uart_buf_A6);
print_uart();
```

```
sprintf(uart_buf,"%c",uart_buf_A7);
print_uart();
sprintf(uart_buf,"%c",uart_buf_A8);
print_uart();
}
void read_B()
{
uart_buf_B0 = eeprom_read_byte1(0xD0);
uart_buf_B1 = eeprom_read_byte1(0xD1);
uart_buf_B2 = eeprom_read_byte1(0xD2);
uart_buf_B3 = eeprom_read_byte1(0xD3);
uart_buf_B4 = eeprom_read_byte1(0xD4);
uart_buf_B5 = eeprom_read_byte1(0xD5);
uart_buf_B6 = eeprom_read_byte1(0xD6);
uart_buf_B7 = eeprom_read_byte1(0xD7);
uart_buf_B8 = eeprom_read_byte1(0xD8);

sprintf(uart_buf,"%c",uart_buf_B0);
print_uart();
sprintf(uart_buf,"%c",uart_buf_B1);
print_uart();
sprintf(uart_buf,"%c",uart_buf_B2);
print_uart();
sprintf(uart_buf,"%c",uart_buf_B3);
print_uart();
sprintf(uart_buf,"%c",uart_buf_B4);
print_uart();
sprintf(uart_buf,"%c",uart_buf_B5);
print_uart();
sprintf(uart_buf,"%c",uart_buf_B6);
print_uart();
sprintf(uart_buf,"%c",uart_buf_B7);
```

```
print_uart();
sprintf(uart_buf,"%c",uart_buf_B8);
print_uart();
}
void read_C()
{
uart_buf_C0 = eeprom_read_byte1(0xE0);
uart_buf_C1 = eeprom_read_byte1(0xE1);
uart_buf_C2 = eeprom_read_byte1(0xE2);
uart_buf_C3 = eeprom_read_byte1(0xE3);
uart_buf_C4 = eeprom_read_byte1(0xE4);
uart_buf_C5 = eeprom_read_byte1(0xE5);
uart_buf_C6 = eeprom_read_byte1(0xE6);
uart_buf_C7 = eeprom_read_byte1(0xE7);
uart_buf_C8 = eeprom_read_byte1(0xE8);

sprintf(uart_buf,"%c",uart_buf_C0);
print_uart();
sprintf(uart_buf,"%c",uart_buf_C1);
print_uart();
sprintf(uart_buf,"%c",uart_buf_C2);
print_uart();
sprintf(uart_buf,"%c",uart_buf_C3);
print_uart();
sprintf(uart_buf,"%c",uart_buf_C4);
print_uart();
sprintf(uart_buf,"%c",uart_buf_C5);
print_uart();
sprintf(uart_buf,"%c",uart_buf_C6);
print_uart();
sprintf(uart_buf,"%c",uart_buf_C7);
print_uart();
```

```
sprintf(uart_buf,"%c",uart_buf_C8);
print_uart();
}
/*****************************************/
/********** Sent data to telophone *************/
/*****************************************/
void sent_tel1()
{
sprintf(uart_buf,"AT+IPR=19200\n");          // Print Message String
print_uart();
delay_ms(200);
sprintf(uart_buf,"AT+CMGF=1\n");             // Print Message String
print_uart();
delay_ms(200);
sprintf(uart_buf,"AT+CMGS=\"+66");
print_uart();
read_A();
sprintf(uart_buf,"\"\n");
print_uart();
delay_ms(200);
sprintf(uart_buf,"Water Upper to level 40 cm");          // Print Message String
print_uart();
delay_ms(200);
sprintf(uart_buf,"%c\n",0x1A);                           // Print Message String
print_uart();
delay_ms(200);
}
void sent_tel2()
{
sprintf(uart_buf,"AT+IPR=19200\n");                      // Print Message String
print_uart();
delay_ms(200);
```

```
sprintf(uart_buf,"AT+CMGF=1\n");                    // Print Message String
print_uart();
delay_ms(200);
sprintf(uart_buf,"AT+CMGS=\"+66");
print_uart();
read_B();
sprintf(uart_buf,"\"\n");
print_uart();
delay_ms(200);
sprintf(uart_buf,"Water Upper to level 40 cm");     // Print Message String
print_uart();
delay_ms(200);
sprintf(uart_buf,"%c\n",0x1A);                      // Print Message String
print_uart();
delay_ms(200);

}
void sent_tel3()
{
sprintf(uart_buf,"AT+IPR=19200\n");                 // Print Message String
print_uart();
delay_ms(200);
sprintf(uart_buf,"AT+CMGF=1\n");                    // Print Message String
print_uart();
delay_ms(200);
sprintf(uart_buf,"AT+CMGS=\"+66");
print_uart();
read_C();
sprintf(uart_buf,"\"\n");
print_uart();
delay_ms(200);
sprintf(uart_buf,"Water Upper to level 40 cm");     // Print Message String
```

```
print_uart();
delay_ms(200);
sprintf(uart_buf,"%c\n",0x1A);                    // Print Message String
print_uart();
delay_ms(200);
}
void sent_tel4()
{
sprintf(uart_buf,"AT+IPR=19200\n");               // Print Message String
print_uart();
delay_ms(200);
sprintf(uart_buf,"AT+CMGF=1\n");                  // Print Message String
print_uart();
delay_ms(200);
sprintf(uart_buf,"AT+CMGS=\"+66");
print_uart();
read_A();
sprintf(uart_buf,"\"\n");
print_uart();
delay_ms(200);
sprintf(uart_buf,"Water Upper to level 45 cm");   // Print Message String
print_uart();
delay_ms(200);
sprintf(uart_buf,"%c\n",0x1A);                    // Print Message String
print_uart();
delay_ms(200);
}
void sent_tel5()
{
sprintf(uart_buf,"AT+IPR=19200\n");               // Print Message String
print_uart();
delay_ms(200);
```

```
sprintf(uart_buf,"AT+CMGF=1\n");                    // Print Message String
print_uart();
delay_ms(200);
sprintf(uart_buf,"AT+CMGS=\"+66");
print_uart();
read_B();
sprintf(uart_buf,"\"\n");
print_uart();
delay_ms(200);
sprintf(uart_buf,"Water Upper to level 45 cm");     // Print Message String
print_uart();
delay_ms(200);
sprintf(uart_buf,"%c\n",0x1A);                      // Print Message String
print_uart();
delay_ms(200);
}
void sent_tel6()
{
sprintf(uart_buf,"AT+IPR=19200\n");                 // Print Message String
print_uart();
delay_ms(200);
sprintf(uart_buf,"AT+CMGF=1\n");                    // Print Message String
print_uart();
delay_ms(200);
sprintf(uart_buf,"AT+CMGS=\"+66");
print_uart();
read_C();
sprintf(uart_buf,"\"\n");
print_uart();
delay_ms(200);
sprintf(uart_buf,"Water Upper to level 45 cm");     // Print Message String
print_uart();
```

```
delay_ms(200);
sprintf(uart_buf,"%c\n",0x1A);                    // Print Message String
print_uart();
delay_ms(200);
}
void sent_tel7()
{
sprintf(uart_buf,"AT+IPR=19200\n");               // Print Message String
print_uart();
delay_ms(200);
sprintf(uart_buf,"AT+CMGF=1\n");                  // Print Message String
print_uart();
delay_ms(200);
sprintf(uart_buf,"AT+CMGS=\"+66");
print_uart();
read_A();
sprintf(uart_buf,"\"\n");
print_uart();
delay_ms(200);
sprintf(uart_buf,"Water Upper to level 55 cm");   // Print Message String
print_uart();
delay_ms(200);
sprintf(uart_buf,"%c\n",0x1A);                    // Print Message String
print_uart();
delay_ms(200);
}
void sent_tel8()
{
sprintf(uart_buf,"AT+IPR=19200\n");               // Print Message String
print_uart();
delay_ms(200);
sprintf(uart_buf,"AT+CMGF=1\n");                  // Print Message String
```

```
print_uart();
delay_ms(200);
sprintf(uart_buf,"AT+CMGS=\"+66");
print_uart();
read_B();
sprintf(uart_buf,"\"\n");
print_uart();
delay_ms(200);
sprintf(uart_buf,"Water Upper to level 55 cm");          // Print Message String
print_uart();
delay_ms(200);
sprintf(uart_buf,"%c\n",0x1A);                           // Print Message String
print_uart();
delay_ms(200);
}
void sent_tel9()
{
sprintf(uart_buf,"AT+IPR=19200\n");                      // Print Message String
print_uart();
delay_ms(200);
sprintf(uart_buf,"AT+CMGF=1\n");                         // Print Message String
print_uart();
delay_ms(200);
sprintf(uart_buf,"AT+CMGS=\"+66");
print_uart();
read_C();
sprintf(uart_buf,"\"\n");
print_uart();
delay_ms(200);
sprintf(uart_buf,"Water Upper to level 55 cm");          // Print Message String
print_uart();
delay_ms(200);
```

```
sprintf(uart_buf,"%c\n",0x1A);                    // Print Message String
print_uart();
delay_ms(200);
}
void sent_tel10()
{
sprintf(uart_buf,"AT+IPR=19200\n");               // Print Message String
print_uart();
delay_ms(200);
sprintf(uart_buf,"AT+CMGF=1\n");                  // Print Message String
print_uart();
delay_ms(200);
sprintf(uart_buf,"AT+CMGS=\"+66");
print_uart();
read_A();
sprintf(uart_buf,"\"\n");
print_uart();
delay_ms(200);
sprintf(uart_buf,"Water lower to level 40 cm");   // Print Message String
print_uart();
delay_ms(200);
sprintf(uart_buf,"%c\n",0x1A);                    // Print Message String
print_uart();
delay_ms(200);

}
void sent_tel11()
{
sprintf(uart_buf,"AT+IPR=19200\n");               // Print Message String
print_uart();
delay_ms(200);
sprintf(uart_buf,"AT+CMGF=1\n");                  // Print Message String
```

```
print_uart();
delay_ms(200);
sprintf(uart_buf,"AT+CMGS=\"+66");
print_uart();
read_B();
sprintf(uart_buf,"\"\n");
print_uart();
delay_ms(200);
sprintf(uart_buf,"Water lower to level 40 cm");          // Print Message String
print_uart();
delay_ms(200);
sprintf(uart_buf,"%c\n",0x1A);                           // Print Message String
print_uart();
delay_ms(200);
}
void sent_tel12()
{
sprintf(uart_buf,"AT+IPR=19200\n");                      // Print Message String
print_uart();
delay_ms(200);
sprintf(uart_buf,"AT+CMGF=1\n");                         // Print Message String
print_uart();
delay_ms(200);
sprintf(uart_buf,"AT+CMGS=\"+66");
print_uart();
read_C();
sprintf(uart_buf,"\"\n");
print_uart();
delay_ms(200);
sprintf(uart_buf,"Water lower to level 40 cm");          // Print Message String
print_uart();
delay_ms(200);
```

```
sprintf(uart_buf,"%c\n",0x1A);                    // Print Message String
print_uart();
delay_ms(200);
}
void sent_tel13()
{
sprintf(uart_buf,"AT+IPR=19200\n");               // Print Message String
print_uart();
delay_ms(200);
sprintf(uart_buf,"AT+CMGF=1\n");                  // Print Message String
print_uart();
delay_ms(200);
sprintf(uart_buf,"AT+CMGS=\"+66");
print_uart();
read_A();
sprintf(uart_buf,"\"\n");
print_uart();
delay_ms(200);
sprintf(uart_buf,"Water lower to level 45 cm");   // Print Message String
print_uart();
delay_ms(200);
sprintf(uart_buf,"%c\n",0x1A);                    // Print Message String
print_uart();
delay_ms(200);
}
void sent_tel14()
{
sprintf(uart_buf,"AT+IPR=19200\n");               // Print Message String
print_uart();
delay_ms(200);
sprintf(uart_buf,"AT+CMGF=1\n");                  // Print Message String
print_uart();
```

```
delay_ms(200);
sprintf(uart_buf,"AT+CMGS=\"+66");
print_uart();
read_B();
sprintf(uart_buf,"\"\n");
print_uart();
delay_ms(200);
sprintf(uart_buf,"Water lower to level 45 cm");          // Print Message String
print_uart();
delay_ms(200);
sprintf(uart_buf,"%c\n",0x1A);                           // Print Message String
print_uart();
delay_ms(200);
}
void sent_tel15()
{
sprintf(uart_buf,"AT+IPR=19200\n");                      // Print Message String
print_uart();
delay_ms(200);
sprintf(uart_buf,"AT+CMGF=1\n");                         // Print Message String
print_uart();
delay_ms(200);
sprintf(uart_buf,"AT+CMGS=\"+66");
print_uart();
read_C();
sprintf(uart_buf,"\"\n");
print_uart();
delay_ms(200);
sprintf(uart_buf,"Water lower to level 45 cm");          // Print Message String
print_uart();
delay_ms(200);
sprintf(uart_buf,"%c\n",0x1A);                           // Print Message String
```

```
print_uart();
delay_ms(200);
}
void sent_tel16()
{
sprintf(uart_buf,"AT+IPR=19200\n");                    // Print Message String
print_uart();
delay_ms(200);
sprintf(uart_buf,"AT+CMGF=1\n");                       // Print Message String
print_uart();
delay_ms(200);
sprintf(uart_buf,"AT+CMGS=\"+66");
print_uart();
read_A();
sprintf(uart_buf,"\"\n");
print_uart();
delay_ms(200);
sprintf(uart_buf,"Water lower to level 55 cm");        // Print Message String
print_uart();
delay_ms(200);
sprintf(uart_buf,"%c\n",0x1A);                         // Print Message String
print_uart();
delay_ms(200);
}
void sent_tel17()
{
sprintf(uart_buf,"AT+IPR=19200\n");                    // Print Message String
print_uart();
delay_ms(200);
sprintf(uart_buf,"AT+CMGF=1\n");                       // Print Message String
print_uart();
delay_ms(200);
```

```
sprintf(uart_buf,"AT+CMGS=\"+66");
print_uart();
read_B();
sprintf(uart_buf,"\"\n");
print_uart();
delay_ms(200);
sprintf(uart_buf,"Water lower to level 55 cm");          // Print Message String
print_uart();
delay_ms(200);
sprintf(uart_buf,"%c\n",0x1A);                           // Print Message String
print_uart();
delay_ms(200);
}
void sent_tel18()
{
sprintf(uart_buf,"AT+IPR=19200\n");                      // Print Message String
print_uart();
delay_ms(200);
sprintf(uart_buf,"AT+CMGF=1\n");                         // Print Message String
print_uart();
delay_ms(200);
sprintf(uart_buf,"AT+CMGS=\"+66");
print_uart();
read_C();
sprintf(uart_buf,"\"\n");
print_uart();
delay_ms(200);
sprintf(uart_buf,"Water lower to level 55 cm");          // Print Message String
print_uart();
delay_ms(200);
sprintf(uart_buf,"%c\n",0x1A);                           // Print Message String
print_uart();
```

```
delay_ms(200);
}
/************************/
/********Check mode**********/
/************************/
void Set_mode1()
{
uart_buf_e0 = eeprom_read_byte1(0xe0);
if (uart_buf_e0==0x31)
{
sent_tel1();
}
else if (uart_buf_e0==0x32)
{
sent_tel2();
}
else if (uart_buf_e0==0x33)
{
sent_tel3();
}
else if (uart_buf_e0==0x34)
{
sent_tel1();
sent_tel2();
}
else if (uart_buf_e0==0x35)
{
sent_tel1();
sent_tel3();
}
else if (uart_buf_e0==0x36)
{
```

```
sent_tel2();
sent_tel3();
}
else if (uart_buf_e0==0x37)
{
sent_tel1();
sent_tel2();
sent_tel3();
}
}
void Set_mode2()
{
uart_buf_e1 = eeprom_read_byte1(0xe1);
if (uart_buf_e1==0x31)
{
sent_tel4();
}
else if (uart_buf_e1==0x32)
{
sent_tel5();
}
else if (uart_buf_e1==0x33)
{
sent_tel6();
}
else if (uart_buf_e1==0x34)
{
sent_tel4();
sent_tel5();
}
else if (uart_buf_e1==0x35)
{
```

```
sent_tel4();
sent_tel6();
}
else if (uart_buf_e1==0x36)
{
sent_tel5();
sent_tel6();
}
else if (uart_buf_e1==0x37)
{
sent_tel4();
sent_tel5();
sent_tel6();
}
}
void Set_mode3()
{
uart_buf_e2 = eeprom_read_byte1(0xe2);
if (uart_buf_e2==0x31)
{
sent_tel7();
}
else if (uart_buf_e2==0x32)
{
sent_tel8();
}
else if (uart_buf_e2==0x33)
{
sent_tel9();
}
else if (uart_buf_e2==0x34)
{
```

```
sent_tel7();
sent_tel8();
}
else if (uart_buf_e2==0x35)
{
sent_tel7();
sent_tel9();
}
else if (uart_buf_e2==0x36)
{
sent_tel8();
sent_tel9();
}
else if (uart_buf_e2==0x37)
{
sent_tel7();
sent_tel8();
sent_tel9();
}
}
void Set_mode4()
{
uart_buf_e0 = eeprom_read_byte1(0xe0);
if (uart_buf_e0==0x31)
{
sent_tel10();
}
else if (uart_buf_e0==0x32)
{
sent_tel11();
}
else if (uart_buf_e0==0x33)
```

```
{
sent_tel12();
}
else if (uart_buf_e0==0x34)
{
sent_tel10();
sent_tel11();
}
else if (uart_buf_e0==0x35)
{
sent_tel10();
sent_tel12();
}
else if (uart_buf_e0==0x36)
{
sent_tel11();
sent_tel12();
}
else if (uart_buf_e0==0x37)
{
sent_tel10();
sent_tel11();
sent_tel12();
}
}
void Set_mode5()
{
uart_buf_e1 = eeprom_read_byte1(0xe1);
if (uart_buf_e1==0x31)
{
sent_tel13();
}
```

```
else if (uart_buf_e1==0x32)
{
sent_tel14();
}
else if (uart_buf_e1==0x33)
{
sent_tel15();
}
else if (uart_buf_e1==0x34)
{
sent_tel13();
sent_tel14();
}
else if (uart_buf_e1==0x35)
{
sent_tel13();
sent_tel15();
}
else if (uart_buf_e1==0x36)
{
sent_tel14();
sent_tel15();
}
else if (uart_buf_e1==0x37)
{
sent_tel13();
sent_tel14();
sent_tel15();
}
}

void Set_mode6()
```

```
{
uart_buf_e2 = eeprom_read_byte1(0xe2);
if (uart_buf_e2==0x31)
{
sent_tel16();
}
else if (uart_buf_e2==0x32)
{
sent_tel17();
}
else if (uart_buf_e2==0x33)
{
sent_tel18();
}
else if (uart_buf_e2==0x34)
{
sent_tel16();
sent_tel17();
}
else if (uart_buf_e2==0x35)
{
sent_tel16();
sent_tel18();
}
else if (uart_buf_e2==0x36)
{
sent_tel17();
sent_tel18();
}
else if (uart_buf_e2==0x37)
{
sent_tel16();
```

```
sent_tel17();
sent_tel18();
}
}
/*************************/
/*********Main***********/
/*************************/

int main(void)
{
LCD_Initial();                          // Initial LCD
LCD_Backlight(1);                       // ON Backlight LED
LCD_SetCursor(0x00);                    // Set Cursor Start Line[2]
sprintf(lcdbuf,"The Flood Warning");
LCD_Print_Buffer();
//unsigned int uart_data ;
uint16_t X1,X2,X3,X4,X5,X6,X7,X8,X9,X10;
uint16_t ADCBinary1;
uint16_t ADCBinary2;
uint16_t ADCBinary3;
uint16_t ADCBinary4;
uint16_t ADCBinary5;
uint16_t ADCBinary6;
uint16_t ADCBinary7;
uint16_t ADCBinary8;
uint16_t ADCBinary9;
uint16_t ADCBinary10;
uint16_t ADCResult1;
uint16_t down1 = 0;
uint16_t down2 = 0;
uint16_t down3 = 0;
uint16_t sum_X;
```

```
uint16_t send_same1 = 0;
uint16_t send_same2 = 0;
uint16_t send_same3 = 0;
uint16_t level_same =1;
DIR_LED  |= (1<<LED);
DDRD &= ~(1 << DDD2);                    // Clear the PD2 pin
// PD2 (PCINT0 pin) is now an input
PORTD |= (1 << PORTD2);                  // turn On the Pull-up
// PD2 is now an input with pull-up enabled
EICRA |= (1 << ISC00);                   // set INT0 to trigger on ANY logic change
EIMSK |= (1 << INT0);                    // Turns on INT0
sei();
//unsigned char ctrlz=0x1a;
init_serial(MYUBRR);                     // Initilial UART0 = 9600,N,8,1

/* Start of Initial ADC Function */
ADMUX  = 0x00;                           // Default Value
ADMUX |= (1<<REFS0);                     // Reference = 01 = AVCC
ADMUX |= (1<<ADLAR);                     // ADLAR = 1 = ADC Result 8 Bit on ADCH
ADCSRA |= (1<<ADEN);                     // Enable ADC Function
ADCSRA |= ((1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0));    // ADC Clock Prescale = 128
/* End of Initial ADC Function */
// Selct ADC Channel
ADMUX &= 0xF0;                                   // Reset Select ADC Channel
ADMUX |= 0x00;                                   // Select ADC = Ch[0]
Index_Count = 0x30;
// Loop Receive & Echo Test //
while(1)                                         // Loop Continue
{
Index_Count++;                                   // 0..9
if(Index_Count > 0x39)
Index_Count = 0x30;
```

```
ADCSRA |= (1<<ADSC);                    // Start ADC Conversion
while((ADCSRA & (1<<ADSC)) == 0);       // Wait ADC Complete
sum_X = (int)ADCH;
ADCSRA |= (1<<ADSC);                    // Start ADC Conversion
while((ADCSRA & (1<<ADSC)) == 0);       // Wait ADC Complete
ADCBinary1 = (int)ADCH;
X1 = ADCBinary1;
delay_ms(200);
ADCSRA |= (1<<ADSC);                    // Start ADC Conversion
while((ADCSRA & (1<<ADSC)) == 0);       // Wait ADC Complete
ADCBinary2 = (int)ADCH;
X2 = ADCBinary2;
delay_ms(200);

ADCSRA |= (1<<ADSC);                    // Start ADC Conversion
while((ADCSRA & (1<<ADSC)) == 0);       // Wait ADC Complete
ADCBinary3 = (int)ADCH;
X3 = ADCBinary3;
delay_ms(200);
ADCSRA |= (1<<ADSC);                    // Start ADC Conversion
while((ADCSRA & (1<<ADSC)) == 0);       // Wait ADC Complete
ADCBinary4 = (int)ADCH;
X4 = ADCBinary4;
delay_ms(200);
ADCSRA |= (1<<ADSC);                    // Start ADC Conversion
while((ADCSRA & (1<<ADSC)) == 0);       // Wait ADC Complete
ADCBinary5 = (int)ADCH;
X5 = ADCBinary5;
delay_ms(200);
ADCSRA |= (1<<ADSC);                    // Start ADC Conversion
while((ADCSRA & (1<<ADSC)) == 0);       // Wait ADC Complete
```

```
ADCBinary6 = (int)ADCH;
X6 = ADCBinary6;
delay_ms(200);
ADCSRA |= (1<<ADSC);                    // Start ADC Conversion
while((ADCSRA & (1<<ADSC)) == 0);       // Wait ADC Complete
ADCBinary7 = (int)ADCH;
X7 = ADCBinary7;
delay_ms(200);

ADCSRA |= (1<<ADSC);                    // Start ADC Conversion
while((ADCSRA & (1<<ADSC)) == 0);       // Wait ADC Complete
ADCBinary8 = (int)ADCH;
X8 = ADCBinary8;
delay_ms(200);
ADCSRA |= (1<<ADSC);                    // Start ADC Conversion
while((ADCSRA & (1<<ADSC)) == 0);       // Wait ADC Complete
ADCBinary9 = (int)ADCH;
X9 = ADCBinary9;
delay_ms(200);
ADCSRA |= (1<<ADSC);                    // Start ADC Conversion
while((ADCSRA & (1<<ADSC)) == 0);       // Wait ADC Complete
ADCBinary10 = (int)ADCH;
X10 = ADCBinary10;
delay_ms(200);
sum_X = (X1 + X2 + X3 + X4 + X5 + X6 + X7 + X8 + X9 + X10)/10;

if ((sum_X>=57) && (sum_X<=61))
{
ADCResult1=60;
down1=0;
down2=0;
down3=0;
```

```
delay_ms(1000);
}
else if  ((sum_X>=51) && (sum_X<=56))
{
ADCResult1=55;
down2=0;
down3=0;
down1++;
if(down1==3)
{
send_same1++;
if (send_same1 ==1)
{
if (level_same<6)
{
Set_mode3();
}
else if (level_same>6)
{
Set_mode6();
}
level_same =6;
send_same2 =0;
send_same3 =0;
}
down1=0;
delay_ms(1000);
}
}
else if  ((sum_X>=47) && (sum_X<=50))
{
ADCResult1=50;
```

```
down1=0;
down2=0;
down3=0;
delay_ms(1000);
}
else if  ((sum_X>=42) && (sum_X<=46))
{
ADCResult1=45;
down1=0;
down3=0;
down2++;
if(down2==3)
{
send_same2++;
if (send_same2 ==1)
{
if (level_same<4)
{
Set_mode2();
}
else if (level_same>4)
{
Set_mode5();
}
level_same =4;
send_same1 =0;
send_same3 =0;
}
down2=0;
delay_ms(1000);
}
}
```

```
else if ((sum_X>=37) && (sum_X<=41))
{
ADCResult1=40;
down1=0;
down2=0;
down3++;
if(down3==3)
{
send_same3++;
if (send_same3 ==1)
{
if (level_same<2)
{
Set_mode1();
}
else if (level_same>2)
{
Set_mode4();
}
level_same =2;
send_same2 =0;
send_same1 =0;
}
down3=0;
delay_ms(1000);
}

}
else if  ((sum_X>=35) && (sum_X<=37))
{
ADCResult1=35;
down1=0;
```

```
down2=0;
down3=0;
delay_ms(1000);
}
else if ((sum_X>=33) && (sum_X<=34))
{
ADCResult1=30;
down1=0;
down2=0;
down3=0;
delay_ms(1000);
}

else
{
ADCResult1=20;
down1=0;
down2=0;
down3=0;
delay_ms(1000);
}
sprintf(uart_buf,"(%c)ADC Output = %d \n",Index_Count,ADCResult1);
print_uart();
sprintf(uart_buf,"ADC count1 = %d \n",down1);
print_uart();
sprintf(uart_buf,"ADC count2 = %d \n",down2);
print_uart();
sprintf(uart_buf,"ADC count3 = %d \n",down3);
print_uart();
delay_ms(1000);

LCD_SetCursor(0x00);
```

```
sprintf(lcdbuf,"ADC Output = %d \n   ",ADCResult1);
LCD_Print_Buffer();
LCD_SetCursor(0x40);
sprintf(lcdbuf,"ADC Count=%d=%d=%d \n",down1,down2,down3);
LCD_Print_Buffer();
delay_ms(1000);
}

}
ISR (INT0_vect)
{
init_serial(MYUBRR);
unsigned int uart_data ;
unsigned int setmode1;
unsigned int setmode2;
unsigned int setmode3;

sprintf(uart_buf,"Warning \n\r");
print_uart();
sprintf(uart_buf,"Setting number tel\n\r");
print_uart();
sprintf(uart_buf,"a:Number tel1\n\r");
print_uart();
sprintf(uart_buf,"b:Number tel2\n\r");
print_uart();
sprintf(uart_buf,"c:Number tel2\n\r");
print_uart();
sprintf(uart_buf,"d:check Number\n\r");
print_uart();
sprintf(uart_buf,"e:Setting Mode 30 cm\n\r");
print_uart();
sprintf(uart_buf,"f:Setting Mode 45 cm\n\r");
```

```
print_uart();
sprintf(uart_buf,"g:Setting Mode 55 cm\n\r");
print_uart();
sprintf(uart_buf,"Press enter number tel setting\n\r");
print_uart();
uart_data = getchar0();
if(uart_data==0x61)
{
sprintf(uart_buf,"\n\rSetting Number tel1\n\r");
print_uart();
sprintf(uart_buf,"\n\rPress number:");
print_uart();
chang_A();
sprintf(uart_buf,"\n\rSetting Complete");
print_uart();
uart_data = getchar0();
}
else if(uart_data==0x62)
{
sprintf(uart_buf,"\n\rSetting Number tel2\n\r");
print_uart();
sprintf(uart_buf,"\n\rPress number:");
print_uart();
chang_B();
sprintf(uart_buf,"\n\rSetting Complete");
print_uart();
uart_data = getchar0();
}
else if(uart_data==0x63)
{
sprintf(uart_buf,"\n\rSetting Number tel3\n\r");
print_uart();
```

```
sprintf(uart_buf,"\n\rPress number:");
print_uart();
chang_C();
sprintf(uart_buf,"\n\rSetting Complete");
print_uart();
uart_data = getchar0();
}

else if (uart_data==0x64)
{
sprintf(uart_buf,"\n\rSetting Number tel1\n\r");
print_uart();
sprintf(uart_buf,"\n\rnumber:0");
print_uart();
read_A();
sprintf(uart_buf,"\n\rSetting Number tel2\n\r");
print_uart();
sprintf(uart_buf,"\n\rnumber:0");
print_uart();
read_B();
sprintf(uart_buf,"\n\rSetting Number tel3\n\r");
print_uart();
sprintf(uart_buf,"\n\rnumber:0");
print_uart();
read_C();
sprintf(uart_buf,"\n");
print_uart();
delay_ms(100);
uart_data = getchar0();
}
else if (uart_data==0x65)
{
```

```
sprintf(uart_buf,"Setting Mode 30cm tel\n\r");
print_uart();
sprintf(uart_buf,"1:Sent tel1\n\r");
print_uart();
sprintf(uart_buf,"2:Sent tel2\n\r");
print_uart();
sprintf(uart_buf,"3:Sent tel3\n\r");
print_uart();
sprintf(uart_buf,"4:Sent tel1/Sent tel2\n\r");
print_uart();
sprintf(uart_buf,"5:Sent tel1/Sent tel3\n\r");
print_uart();
sprintf(uart_buf,"6:Sent tel2/Sent tel3\n\r");
print_uart();
sprintf(uart_buf,"7:Sent tel1/Sent tel2/Sent tel3\n\r");
print_uart();
setmode1 = getchar0();
if (setmode1==0x31)
{
eeprom_write_byte1(0xe0,0x31);
sprintf(uart_buf,"Setting Mode 1 Complete\n\r");
print_uart();
uart_data = getchar0();
}
else if (setmode1==0x32)
{
eeprom_write_byte1(0xe0,0x32);
sprintf(uart_buf,"Setting Mode 2 Complete\n\r");
print_uart();
uart_data = getchar0();
}
else if (setmode1==0x33)
```

```
{
eeprom_write_byte1(0xe0,0x33);
sprintf(uart_buf,"Setting Mode 3 Complete\n\r");
print_uart();
uart_data = getchar0();
}
else if (setmode1==0x34)
{
eeprom_write_byte1(0xe0,0x34);
sprintf(uart_buf,"Setting Mode 4 Complete\n\r");
print_uart();
uart_data = getchar0();
}
else if (setmode1==0x35)
{
eeprom_write_byte1(0xe0,0x35);
sprintf(uart_buf,"Setting Mode 5 Complete\n\r");
print_uart();
uart_data = getchar0();
}
else if (setmode1==0x36)
{
eeprom_write_byte1(0xe0,0x36);
sprintf(uart_buf,"Setting Mode 6 Complete\n\r");
print_uart();
uart_data = getchar0();
}
else if (setmode1==0x37)
{
eeprom_write_byte1(0xe0,0x37);
sprintf(uart_buf,"Setting Mode 7 Complete\n\r");
print_uart();
```

```
uart_data = getchar0();
}
}
else if (uart_data==0x66)
{
sprintf(uart_buf,"Setting Mode 45cm tel\n\r");
print_uart();
sprintf(uart_buf,"1:Sent tel1\n\r");
print_uart();
sprintf(uart_buf,"2:Sent tel2\n\r");
print_uart();
sprintf(uart_buf,"3:Sent tel3\n\r");
print_uart();
sprintf(uart_buf,"4:Sent tel1/Sent tel2\n\r");
print_uart();
sprintf(uart_buf,"5:Sent tel1/Sent tel3\n\r");
print_uart();
sprintf(uart_buf,"6:Sent tel2/Sent tel3\n\r");
print_uart();
sprintf(uart_buf,"7:Sent tel1/Sent tel2/Sent tel3\n\r");
print_uart();
setmode2 = getchar0();
if (setmode2==0x31)
{
eeprom_write_byte1(0xe1,0x31);
sprintf(uart_buf,"Setting Mode 1 Complete\n\r");
print_uart();
uart_data = getchar0();
}
else if (setmode2==0x32)
{
eeprom_write_byte1(0xe1,0x32);
```

```
sprintf(uart_buf,"Setting Mode 2 Complete\n\r");
print_uart();
uart_data = getchar0();
}
else if (setmode2==0x33)
{
eeprom_write_byte1(0xe1,0x33);
sprintf(uart_buf,"Setting Mode 3 Complete\n\r");
print_uart();
uart_data = getchar0();
}
else if (setmode2==0x34)
{
eeprom_write_byte1(0xe1,0x34);
sprintf(uart_buf,"Setting Mode 4 Complete\n\r");
print_uart();
uart_data = getchar0();
}
else if (setmode2==0x35)
{
eeprom_write_byte1(0xe1,0x35);
sprintf(uart_buf,"Setting Mode 5 Complete\n\r");
print_uart();
uart_data = getchar0();
}
else if (setmode2==0x36)
{
eeprom_write_byte1(0xe1,0x36);
sprintf(uart_buf,"Setting Mode 6 Complete\n\r");
print_uart();
uart_data = getchar0();
}
```

```
else if (setmode2==0x37)
{
eeprom_write_byte1(0xe1,0x37);
sprintf(uart_buf,"Setting Mode 7 Complete\n\r");
print_uart();
uart_data = getchar0();
}
}
else if (uart_data==0x67)
{
sprintf(uart_buf,"Setting Mode 55 cm tel\n\r");
print_uart();
sprintf(uart_buf,"1:Sent tel1\n\r");
print_uart();
sprintf(uart_buf,"2:Sent tel2\n\r");
print_uart();
sprintf(uart_buf,"3:Sent tel3\n\r");
print_uart();
sprintf(uart_buf,"4:Sent tel1/Sent tel2\n\r");
print_uart();
sprintf(uart_buf,"5:Sent tel1/Sent tel3\n\r");
print_uart();
sprintf(uart_buf,"6:Sent tel2/Sent tel3\n\r");
print_uart();
sprintf(uart_buf,"7:Sent tel1/Sent tel2/Sent tel3\n\r");
print_uart();
setmode3 = getchar0();
if (setmode3==0x31)
{
eeprom_write_byte1(0xe2,0x31);
sprintf(uart_buf,"Setting Mode 1 Complete\n\r");
print_uart();
```

```
uart_data = getchar0();
}
else if (setmode3==0x32)
{
eeprom_write_byte1(0xe2,0x32);
sprintf(uart_buf,"Setting Mode 2 Complete\n\r");
print_uart();
uart_data = getchar0();
}
else if (setmode3==0x33)
{
eeprom_write_byte1(0xe2,0x33);
sprintf(uart_buf,"Setting Mode 3 Complete\n\r");
print_uart();
uart_data = getchar0();
}
else if (setmode3==0x34)
{
eeprom_write_byte1(0xe2,0x34);
sprintf(uart_buf,"Setting Mode 4 Complete\n\r");
print_uart();
uart_data = getchar0();
}
else if (setmode3==0x35)
{
eeprom_write_byte1(0xe2,0x35);
sprintf(uart_buf,"Setting Mode 5 Complete\n\r");
print_uart();
uart_data = getchar0();
}
else if (setmode3==0x36)
{
```

```
eeprom_write_byte1(0xe2,0x36);
sprintf(uart_buf,"Setting Mode 6 Complete\n\r");
print_uart();
uart_data = getchar0();
}
else if (setmode3==0x37)
{
eeprom_write_byte1(0xe2,0x37);
sprintf(uart_buf,"Setting Mode 7 Complete\n\r");
print_uart();
uart_data = getchar0();
}
}
}
/**********************/
/** Shift Byte to 74HC595 **/
/**********************/
void LCD_Shift_74HC595(unsigned char value)   // Send Byte(Software SPI)
{
unsigned int Bit = 0;                         // Bit Counter
CLK_74HC595_LO();                             // Standby SCLK
for (Bit = 0; Bit < 8; Bit++)                 // 8 Bit Write
{
CLK_74HC595_LO();                             // Standby SCLK
if((value&0x80)>>7)
{
SIN_74HC595_HI();
}
else
{
SIN_74HC595_LO();
}
```

```
// Shift Bit Data
CLK_74HC595_HI();                    // Strobe Signal Bit(SDIN)
value <<= 1;                         // Next Bit Data
}
CLK_74HC595_LO();                    // Standby SCLK

//Strobe Latch Output
STR_74HC595_HI();                    // Latch Output
delay_us(50);
STR_74HC595_LO();
}
/**********************/
/* Send 4 Bit Data to LCD */
/**********************/
void LCD_Shift_Nibble(unsigned char nibble, int lcd_command)
{
if(Backlight_LED)
  {
   nibble |=  SHIFT_BL;              // BL = HIGH(On Backlight)
  }
  else
  {
   nibble &= ~SHIFT_BL;              // BL = LOW(OFF Backlight)
  }

  if (lcd_command)
  {
   nibble &= ~SHIFT_RS;              // RS = LOW(Command)
  }
  else
  {
   nibble |=  SHIFT_RS;              // RS = HIGH(Data)
```

```
 }

 nibble &= ~SHIFT_RW;                          // RW = LOW(Write)

 nibble &= ~SHIFT_EN;                          // Enable = LOW(Standby)
 LCD_Shift_74HC595(nibble);
 nibble |= SHIFT_EN;                           // Enable = HIGH(Start Enable)
 LCD_Shift_74HC595(nibble);
 nibble &= ~SHIFT_EN;                          // Enable = LOW(Standby)
 LCD_Shift_74HC595(nibble);
}
/***************/
/* Shift Byte to LCD */
/***************/
void LCD_Shift_Byte(unsigned char value, int lcd_command)
{
 int nibble = 0;
 nibble = value & 0xF0;                        // Send LCD Data MSB 4 Bit
 LCD_Shift_Nibble(nibble, lcd_command);
 nibble = (value << 4) & 0xF0;                 // Send LCD Data LSB 4 Bit
 LCD_Shift_Nibble(nibble, lcd_command);
}
/**********************/
/* Read Busy LCD Status */
/**********************/
void LCD_Busy_LCD(int lcd_command)
{
 if (!lcd_command)
 {
  delay_us(150);                               // Delay Data = 100uS
 }
 else
```

```
 {
  delay_ms(150);                                  // Delay Command = 2mS
 }
}
/*******************************/
/* Backlight LED ON/OFF Control */
/*******************************/
void LCD_Backlight(int LED_Status)                // Backlight ON/OFF
{
 Backlight_LED = LED_Status;
}
/*********************/
/* Clear Screen Display */
/*********************/
void LCD_ClearScreen(void)
{
 LCD_Command(0x01);                               // Clear Screen Display
}


/********************/
/* Set Cursor Position */
/********************/
void LCD_SetCursor(unsigned char Cursor)
{
 LCD_Command(Cursor | 0x80);                      // Set DDRAM Command = D7 Set
}
/*******************************/
/* Print Display Data(ASCII) to LCD */
/*******************************/
void LCD_Print_Buffer(void)
{
```

```
 char *p;
 p = lcdbuf;
 do                                    // Get ASCII & Write to LCD Until null
 {
  LCD_Ascii(*p);                       // Write ASCII to LCD
  p++;                                 // Next ASCII
 }
 while(*p != '\0');                    // End of ASCII (null)
 return;
}
/**********************/
/* Print Value of Constant */
/**********************/
void LCD_Print_String(const char c[])
{
 while (*c)
 {
  LCD_Ascii(*c++);
 }
}
/***************************/
/* Print Value of Unsigned Char */
/***************************/
void LCD_Ascii(unsigned char b)
{
 LCD_Shift_Byte(b, 0);                 // Write Data to LCD
 LCD_Busy_LCD(0);                      // Wait Busy Data
}
/**********************/
/* Write Command to LCD */
/**********************/
void LCD_Command(unsigned char value)
```

```
{
 LCD_Shift_Byte(value, 1);                    // Command
 LCD_Busy_LCD(1);                             // Wait Busy Command
}
/***************************/
/* Initial Character LCD Display */
/***************************/
void LCD_Initial()
{
 DIR_74HC595_SIN  |= SIN_74HC595_PIN;         // Initial Output Pin

 DIR_74HC595_STR  |= STR_74HC595_PIN;
 DIR_74HC595_CLK  |= CLK_74HC595_PIN;
 SIN_74HC595_LO();                            // Standby Signal
 STR_74HC595_LO();
 CLK_74HC595_LO();
 delay_ms(150);                               // Power-On Delay
 // Start of 4 Bit LCD Interface Initial
 LCD_Shift_Nibble(0x30, 1);                   // Step[1] = D7:D6:D4:D5 = 0:0:1:1
 delay_ms(5);                                 // Wait 4.1mS
 LCD_Shift_Nibble(0x30, 1);                   // Step[2] = D7:D6:D4:D5 = 0:0:1:1
 delay_us(100);                               // Wait 100uS
 LCD_Shift_Nibble(0x30, 1);                   // Step[3] = D7:D6:D4:D5 = 0:0:1:1
 LCD_Busy_LCD(1);                             // Wait Busy Command
 LCD_Shift_Nibble(0x20, 1);                   // Step[4] = D7:D6:D4:D5 = 0:0:1:0
 LCD_Busy_LCD(1);                             // Wait Busy Command
 // End of 4 Bit LCD Interface Initial
 LCD_Command(0x28);          // Function Set (DL=0 4-Bit,N=1 2 Line,F=0 5X7)
 LCD_Command(0x0C);  // Display on/off Control (Entry Display,Cursor off,Cursor not Blink)
 LCD_Command(0x06);          // Entry Mode Set (I/D=1 Increment,S=0 Cursor Shift)
 LCD_Command(0x01);          // Clear Display  (Clear Display,Set DD RAM Address=0)
}
```

```
/**************/
/* Initial UART */
/**************/
void init_serial(unsigned int ubrr)
{
 /* Set baud rate */
 UBRR0H = (unsigned char)(ubrr>>8);
 UBRR0L = (unsigned char)ubrr;
 /* Enable receiver and transmitter */
 UCSR0B = (1<<RXEN0)|(1<<TXEN0);
 //UCSR0B |= (1<<RXCIE0)|(1<<TXCIE0);
 /* Set frame format: 8data, 2stop bit */
 UCSR0C = (1<<USBS0)|(3<<UCSZ00);
}
/************************/
/* Write Character To UART */
/************************/
int putchar0(unsigned char data)
{
 if (data == '\n')
 {
  while (!( UCSR0A & (1<<UDRE0)));          // Wait TXD Buffer Empty
  UDR0 = 0x0D;                              // Write CR
 }
 while (!( UCSR0A & (1<<UDRE0)));           // Wait TXD Buffer Empty
 UDR0 = data;
  Data =  UDR0 ;                            // Write Data
 return 0;
}

/************************/
/* Get character From UART */
```

```
/*************************/
unsigned char getchar0()
{
 while(!(UCSR0A & (1<<RXC0)));          // Wait RXD Receive Data Ready
 return (UDR0);                         // Get Receice Data & Return
}
/*********************/
/* Print String to UART */
/*********************/
void print_uart(void)
{
 char *p;                               // Pointer Buffer
 p = uart_buf;                          // UART Buffer

 do                                     // Get char & Print Until null
 {
  putchar0(*p);                         // Write char to UART
  p++;                                  // Next char
 }
 while(*p != '\0');                     // End of ASCII (null)
 return;
}
/*********************/
/* Delay 1.65535 mS */
/*********************/
void delay_ms(unsigned int time)
{
 while(time-->0)
 {
  _delay_ms(1.0);
 }
}
```

```
/*******************/
/* Delay 1.65535 ms */
/*******************/
void delay_us(unsigned aint time)
{
 while(time-->0)
 {
  _delay_us(10);
 }
}
```