บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

โครงงาน “ระบบป้องกันการโจรกรรมและติดตามรถจักรยานยนต์พร้อมการแจ้งเตือนผ่านโทรศัพท์มือถือ” เป็นโครงงานที่เกี่ยวข้องกับไมโครคอนโทรลเลอร์และอิเล็กทรอนิกส์ มีเนื้อหาที่เกี่ยวข้องดังต่อไปนี้

2.1 พื้นฐานไมโครคอนโทรลเลอร์ [1]

ไมโครคอนโทรลเลอร์เป็นอุปกรณ์ทางอิเล็กทรอนิกส์ชนิดหนึ่งซึ่งมีความสามารถในการใช้งานเป็นอุปกรณ์ควบคุมที่สามารถโปรแกรมการทำงานได้ซึ่งในปัจจุบันนี้มีการใช้งานไมโครคอนโทรลเลอร์กันอย่างแพร่หลายเนื่องจากใช้งานง่ายและสามารถประยุกต์ใช้งานได้อย่างหลากหลาย

1. ส่วนประกอบของไมโครคอนโทรลเลอร์

ในปัจจุบันนี้ไมโครคอนโทรลเลอร์มีการผลิตขึ้นมาหลายแบบหลายรุ่นขึ้นอยู่กับบริษัทผู้ผลิตแต่โดยทั่วไปแล้วไมโครคอนโทรลเลอร์จะประกอบด้วยส่วนประกอบที่คล้ายกันดังนี้

1.1 หน่วยประมวลผลกลาง (Central Processor Unit) จะเป็นส่วนที่เป็นตัวตัดสินใจต่างๆ ซึ่งจะทำงานตามโปรแกรมที่เราเขียนและอัดเข้าไปในตัวไมโครคอนโทรลเลอร์

1.2 หน่วยความจำ (Memory) เป็นตัวเก็บข้อมูลต่างๆที่ต้องใช้ในไมโครคอนโทรล- เลอร์ ไม่ว่าจะเป็นหน่วยความจำโปรแกรมหรือหน่วยความจำข้อมูลโดยหน่วยความจำที่ใช้ได้แก่ ROM , EPROM , EEPROM , RAM และ FLASH

1.3 พอร์ตสัญญาณเข้าและออก (Input & Output Port) เป็นส่วนที่จะใช้ในการติดต่อกับอุปกรณ์ภายนอก

1.4 คุณสมบัติอื่นๆไมโครคอนโทรลเลอร์ สมัยใหม่จะมีฟังก์ชันพิเศษเพิ่มเติมเช่น Time/Counter, Analog to Digital Converter , Analog Comparator , UART/USART

2. AVR ไมโครคอนโทรลเลอร์ [2]

สำหรับโครงงานนี้เลือกใช้ไมโครคอนโทรลเลอร์ตระกูล AVR ATmega128 เป็นไมโครคอนโทรลเลอร์ในตระกูล AVR ของบริษัท Atmel ซึ่งจะมีคุณสมบัติและข้อมูลทั่วไปของไมโครคอนโทรเลอร์ดังต่อไปนี้

2.1 คุณสมบัติที่สำคัญของไมโครคอนโทรเลอร์ AVR ATmega128

(ก) ไมโครคอนโทรลเลอร์ขนาด 8 บิต กำลังไฟต่ำ ประสิทธิภาพสูง

(ข) สถาปัตยกรรมแบบ Advanced RISC (Reduce Instruction Set Computer)

- 133 คำสั่ง มีความเร็วในการประมวลผล 1 คำสั่งต่อ 1 สัญญาณนาฬิกา
- รีจิสเตอร์ใช้งาน 32 ตัวขนาด 8 บิต และรีจิสเตอร์ควบคุม
- ความเร็วในการทำงานสูงถึง 16 MIPS ที่ความถี่สัญญาณนาฬิกา 16 MHz

(ค) หน่วยความจำโปรแกรมแบบ Nonvolatile และหน่วยความจำข้อมูล

- หน่วยความจำโปรแกรมแบบ Nonvolatile และหน่วยความจำข้อมูล
- หน่วยความจำข้อมูล EEPROM ขนาด 4 กิโลไบต์ เขียน/ลบได้ 100,000 ครั้ง
- หน่วยความจำข้อมูล SRAM ขนาด 4 กิโลไบต์

(ง) สามารถขยายหน่วยความจำได้มากถึง 64 กิโลไบต์

(จ) มีวงจรดีบักอินเตอร์เฟสแบบ JTAG

(ฉ) โมดูลใช้งานประกอบไปด้วย

- โมดูลไทเมอร์/เคาน์เตอร์ ขนาด 8 บิต 2 ตัวพร้อมปรีสเกลเลอร์และโหมดเปรียบเทียบ
- กำหนดให้ทำงานแบบ 16 บิต ด้วยการใช้ไทเมอร์ 2 ตัว ทำงานร่วมกันพร้อมปรีสเกลเลอร์ โหมดเปรียบเทียบและโหมดตรวจจับสัญญาณอินพุต
- ตัวนับแบบเรียลไทม์กับออสซิเลเตอร์แยกเฉพาะ
- โมดูลสร้างสัญญาณ PWM 8 ช่อง
- PWM 6 ช่องกำหนดความละเอียดได้ตั้งแต่ 2 ถึง 16 บิต
- โมดูลเชื่อมต่อแบบอนุกรม TWI(Two-wire Serial), SPI และ USART
- โมดูลวัอตดอกซ์ไทเมอร์ (Watchdog timer)
- โมดูลแปลงสัญญาณอะนาล็อกเป็นดิจิตอลขนาด 10 บิต

- โมดูลเปรียบเทียบแรงดันอนาล็อก

(ช) คุณสมบัติพื้นฐานสำหรับไมโครคอนโทรลเลอร์

- เพาเวอร์ออนรีเซตและเบราว์เอาต์ดีเทคแบบโปรแกรมได้
- RC ออสซิเลเตอร์ภายในเพื่อสร้างสัญญาณความถี่
- แหล่งอินเตอร์รัปต์ทั้งภายในและภายนอก
- 6 โหมดสลีปประกอบด้วย โหมด Idle, ADC Noise Reduction, Power-save, Power-down, Standbye และ Extended Standby
- กำหนดโหมดสัญญาณความถี่นาฬิกาด้วยซอฟต์แวร์

(ฌ) ขาพอร์ตอินพุต/เอาท์พุต 53 ขาประกอบด้วยพอร์ต A, B, C, D, E, F, G ดังแสดงในเรื่องของรายละเอียดขาพอร์ต AVR ATmega128

2.2 การทำงานกับบิตและไบต์ข้อมูล

การใช้งานไมโครคอนโทรเลอร์ โดยส่วนใหญ่แล้วจะเป็นการควบคุมพอร์ตอินพุตเอาท์พุตแบบดิจิตอล ในระดับบิตและไบต์ การสร้างไลบรารีเพื่อจัดการกับข้อมูลในระดับบิตและไบต์ (Bit/Byte Manipulation) จึงมีความจำเป็น นอกจากจะนำมาใช้งานได้อย่างรวดเร็วแล้วยังสามารถนำไปแจกจ่ายให้กับทีมพัฒนา เพื่อการทำงานที่รวดเร็วและเป็นไปในทิศทางเดียวกันอีกด้วย โดยฟังก์ชันไลบรารีที่จะสร้างขึ้นจะประกอบไปด้วยฟังก์ชั่นที่เกี่ยวข้องกับบิตและไบต์ข้อมูลโดยตรง เช่น การเลื่อนบิตข้อมูล การหมุนบิตข้อมูล การเซตบิต การเคลียร์บิต การสลับบิต การสร้างข้อมูลจาก 8 บิต เป็น 16 บิตและ 32 บิตเป็นต้น

2.2.1 บิตและไบต์ข้อมูล

ไมโครคอนโทรลเลอร์ AVR เป็นไมโครคอนโทรลเลอร์ขนาด 8 บิต ประกอบไปด้วยรีจิสเตอร์ขนาด 8 บิต หากต้องการทำงานในแบบ 16 บิตก็จะนำรีจิสเตอร์ ขนาด 8 บิต 2 ตัวมาทำงานร่วมกัน ลักษณะของข้อมูลขนาด 8 บิต แสดงดังตารางที่ 2.1

ตารางที่ 2.1 แสดงตำแหน่งบิตและชื่อเรียก

ตำแหน่งบิต	บิต7	บิต6	บิต5	บิต4	บิต3	บิต2	บิต1	บิต0	ค่า
ขนาด8บิต	MSB							LSB	
ขนาด4บิต	Nibble High				Nibble low				
ขนาด1ไบท์	BYTE								
ค่าในแต่ละบิต	1	0	1	0	1	0	0	1	A9

จากตารางที่ 2.1

- บิตที่ 7 เป็นตำแหน่งบิตสูงสุดหรือที่เรียกว่า บิตนัยสำคัญสุงสุด (Most significant bit) หรือชื่อย่อ MSB บิต

- บิตที่ 0 เป็นตำแหน่งบิตต่ำสุดหรือที่เรียกว่า บิตนัยสำคัญต่ำสุด (Least significant bit) หรือชื่อย่อ LSB บิต

- ตำแหน่งบิตที่ 0 ภึงบิตที่ 3 เรียกว่า 4 บิตร่าง หรือ Nibble Low บิต

- ตำแหน่งบิตที่ 4 ถึงบิตที่ 7 เรียกว่า 4 บิตบน หรือ Nibble High บิต

- บิตที่ 0 ถึง 7 มีขนาด 8 บิตหรือเท่ากับ 1 ไบต์

การทำงานกับบิตข้อมูลจึงต้องเข้าใจถึงชื่อเรียกในแต่ละตำแหน่งและในแต่ละบิตข้อมูล สำหรับภาษา C จะมีการดำเนินการทางบิตข้อมูลที่เรียกว่า Bitwise Operator แสดงดังตารางที่ 2.2

ตารางที่ 2.2 ตัวดำเนินการทางบิตไบต์ข้อมูล

เครื่องหมายดำเนินการ	การทำงาน	ตัวอย่าง
&	การ AND ข้อมูล ผลลัพธ์จะเป็น 1 เมื่อบิตทั้งสองเป็น 1 นอกนั้นเป็น 0 (1 & 1 เท่ากับ 1)	a & b ผลลัพธ์ คือ 0x00
|	การ OR ข้อมูล ผลลัพธ์จะเป็น 0 เมื่อบิตทั้งสองเป็น 0 นอกนั้นเป็น 1 (0 | 0เท่ากับ 0)	a | b ผลลัพธ์ คือ 0x13
^	การ XOR ข้อมูล ผลลัพธ์จะเป็น 0 เมื่อค่าบิตเหมือนกันหากค่าบิตทั้งสองต่างกันจะให้ค่าเป็น 1 (1 ^ 1 เท่ากับ 0 หรือ 0 ^ 0เท่ากับ 0)	a^b ผลลัพธ์ คือ 0x13
<<	การเลื่อนบิตไปทางขวา (Shift Left) ศูนย์จะถูกเติมเข้าไปทางซ้าย	a<<1 ผลลัพธ์ คือ 0x20
>>	การเลื่อนบิตไปทางขวา (Shift Right) ศูนย์จะถูกเติมเข้าไปทางขวา	a>> 1 ผลลัพธ์ คือ 0x08
~	การ NOT คือการกลับค่าบิต ให้เป็นค่าตรงข้าม	~a ผลลัพธ์ คือ 0xFA

จากตารางที่ 2.2 เมื่อกำหนดให้ a = 0x10 และ b=0x03

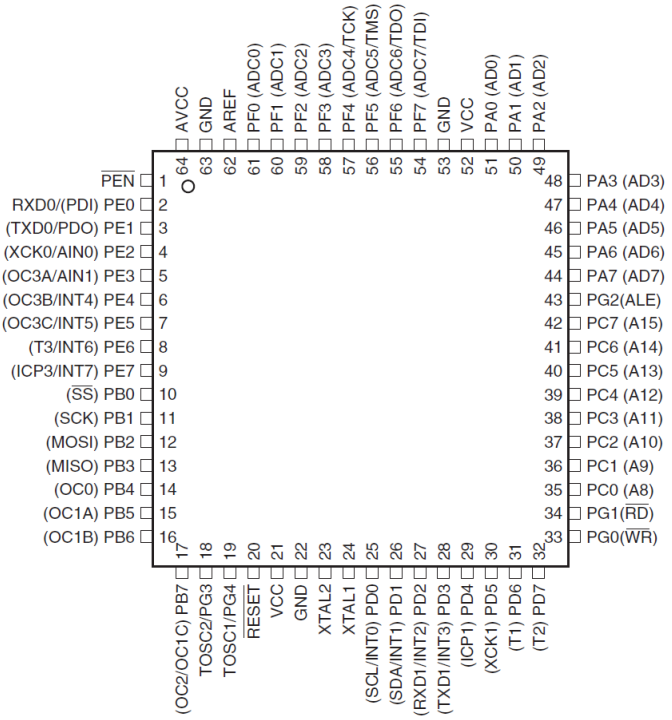
การทำงานกับบิตและไบต์ข้อมูลจึงอาศัยการดำเนินการทางบิตไบต์ข้อมูล ในตารางที่ 2.2 มาใช้ในการจัดการกับข้อมูลทั้งในระดับบิตและในระดับไบต์ การจัดการข้อมูลในระดับบิตและไบต์มีความจำเป็นที่ต้องทำความเข้าใจ เนื่องจากการใช้งานไมโครคอนโทรลเลอร์จะเกี่ยวข้องกับข้อมูลและรีจิสเตอร์ขนาด 8 บิตหรือ 1 ไบต์ โดยผลการกระทำทางบิตสรุปได้ดังตารางที่ 2.3

ตารางที่ 2.3 ตัวดำเนินการทางบิตไบต์ข้อมูลและผลลัพธ์

A	B	A AND B	A OR B	A XOR B	~A	~B
0	0	0	0	0	1	1
0	1	0	1	1	1	0
1	0	0	1	1	0	1
1	1	1	1	0	0	0

2.2.2 อินพุต/เอาต์พุต AVR ATmega128

ATmega128 เป็นไมโครคอนโทรลเลอร์ขนาด 8 บิตตระกูล AVR สถาปัตยกรรมแบบ RISC (Reduced Instruction Set Computer) เช่นเดียวกับ ATmega16 แต่มีคุณสมบัติที่มากกว่า ในตัวถังแบบ Thin Profile Plastic Quad Flat Package (TQFP) และ Micro Lead Frame Package (MLF) การจัดตำแหน่งขาพอร์ตแสดงดังภาพที่ 2.1



ภาพที่ 2.1 รายละเอียดขา AVR ATmega128

2.2.3 รายละเอียดขาพอร์ต ATmega128

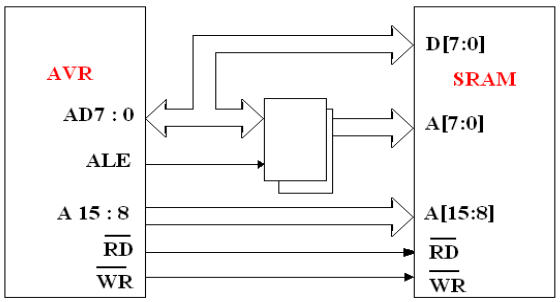
ไมโครคอนโทรลเลอร์ AVR ATmega128 มีจำนวนขาพอร์ตอินพุตดิจิตอลทั้งสิ้น 53 ขา พอร์ตประกอบไปด้วยพอร์ต A ถึง F พอร์ตละ 8 บิต และพอร์ต G จำนวน 5 บิต โดยในแต่ละขาพอร์ตสามารถทำหน้าที่ได้มากกว่าหนึ่งหน้าที่ โดยหน้าที่พิเศษเพิ่มเติมจะอ้างอิงกับคุณสมบัติของโมดูลต่างๆ มีรายละเอียดดังตารางที่ 2.4

ตารางที่ 2.4 แสดงขา PORT A (PA0-PA7)

ขาพอร์ต	ขาพอร์ตฟังก์ชั่นพิเศษ
PA7	AD7 (ขาเชื่อมต่อหน่วยความจำภายนอก แอดเดรสและดาต้า บิตที่ 7)
PA6	AD6 (ขาเชื่อมต่อหน่วยความจำภายนอก แอดเดรสและดาต้า บิตที่ 6)
PA5	AD5 (ขาเชื่อมต่อหน่วยความจำภายนอก แอดเดรสและดาต้า บิตที่ 5)
PA4	AD4 (ขาเชื่อมต่อหน่วยความจำภายนอก แอดเดรสและดาต้า บิตที่ 4)
PA3	AD3 (ขาเชื่อมต่อหน่วยความจำภายนอก แอดเดรสและดาต้า บิตที่ 3)
PA2	AD2 (ขาเชื่อมต่อหน่วยความจำภายนอก แอดเดรสและดาต้า บิตที่ 2)
PA1	AD1 (ขาเชื่อมต่อหน่วยความจำภายนอก แอดเดรสและดาต้า บิตที่ 1)
PA0	AD0 (ขาเชื่อมต่อหน่วยความจำภายนอก แอดเดรสและดาต้า บิตที่ 0)

ขา PORT A นอกจากจะใช้เป็นขาพอร์ตอินพุตเอาท์พุตดิจิตอลขนาด 8 บิตแล้ว ยังมีคุณสมบัติที่สำคัญดังนี้

-ใช้เป็นขาพอร์ตเพื่อต่อขยายหน่วยความจำภายนอกเพิ่มเติม โดยเป็นทั้งขาแอดเดรสและดาต้าขนาด 8 บิต รูปแบบการต่อใช้งานหน่วยความจำภายนอกเพิ่มเติมแสดงดังภาพที่ 2.2 ซึ่งเป็นการต่อขยายหน่วยความจำประเภท SRAM เพิ่มเติม โดยการต่อขยายหน่วยความจำจะใช้ขา PORT A เป็นทั้งขาแอดเดรสและดาต้า 8 บิตล่าง (บิตที่ 0 ถึงบิตที่ 7) และใช้ขา PORTC เป็นขาแอดเดรส 8บิตบน (บิตที่ 8 ถึงบิตที่ 15)



ภาพที่ 2.2 การต่อหน่วยความจำแรม (SRAM) ภายนอกกับ AVR ATmega128

ตารางที่ 2.5 แสดงขา PORT B (PB0-PB7)

ขาพอร์ต	ขาพอร์ตฟังก์ชันพิเศษ
PB7	OC2/OC1C (โมดูลเปรียบเทียบและสร้งสัญญาณเอาต์พุต PWM สำหรับไทเมอร์/เคาน์เตอร์ 2 หรือโมดูลเปรียบเทียบและสร้างสัญญาณเอาต์พุต PWM ช่อง C สำหรับไทเมอร์/เคาน์เตอร์ 1)
PB6	OC1B (โมดูลเปรียบเทียบและสร้างสัญญาณเอาต์พุต PWM ช่อง B สำหรับไทเมอร์/เคาน์เตอร์ 1)
PB5	OC1A (โมดูลเปรียบเทียบและสร้างสัญญาณเอาต์พุต PWM ช่อง A สำหรับไทเมอร์/เคาน์เตอร์ 1)
PB4	OC0 (โมดูลเปรียบเทียบและสร้างสัญญาณเอาต์พุต PWM สำหรับไทเมอร์/เคาน์เตอร์)
PB3	MISO (ขาสัญญาณ อินพุตมาสเตอร์/เอาต์พุตสเลฟ สำหรับบัส SPI)
PB2	MOSI (ขาสัญญาณ อินพุตมาสเตอร์/เอาต์พุตสเลฟ สำหรับบัส SPI)
PB1	SCK (ขาสัญญาณนาฬิกาสำหรับบัส SPI)
PB0	SS (ขาสัญญาณอินพุตเลือกสเลฟ สำหรับบัส SPI)

นอกเหนือจากตารางที่ 2.5 สามารถกำหนดเพิ่มเติมได้ดังนี้

ขา PORT B นอกจากจะใช้เป็นขาพอร์ตอินพุตเอาต์พุตดิจิตอลขนาด 8 บิตแล้ว ยังมีคุณสมบัติที่สำคัญดังนี้

- ใช้เป็นขาพอร์ตสำหรับสร้างสัญญาณ Pulse Width Modulator (PWM) 3ช่องทางคือ A, B และ C ซึ่งนำไปประยุกต์ใช้ในการควบคุมการทำงานของมอเตอร์ และใช้เป็นขาพอร์ตสำหรับเชื่อมต่อกับอุปกรณ์ภายนอกแบบอนุกรม (Serial Peripheral Interface) หรือที่นิยมเรียกว่าบัสแบบ SPI

ตารางที่ 2.6 แสดงขา PORT C (PC0-PC7)

ขาพอร์ต	ขาพอร์ตฟังก์ชันพิเศษ
PC7	A15 (ขาสัญญาณแอดเดรส บิตที่ 15)
PC6	A14 (ขาสัญญาณแอดเดรส บิตที่ 14)
PC5	A13 (ขาสัญญาณแอดเดรส บิตที่ 13)
PC4	A12 (ขาสัญญาณแอดเดรส บิตที่ 12)
PC3	A11 (ขาสัญญาณแอดเดรส บิตที่ 11)
PC2	A10 (ขาสัญญาณแอดเดรส บิตที่ 10)
PC1	A9 (ขาสัญญาณแอดเดรส บิตที่ 9)
PC0	A8 (ขาสัญญาณแอดเดรส บิตที่ 8)

นอกเหนือจากตารางที่ 2.6 สามารถกำหนดเพิ่มเติมได้ดังนี้

ขา PORT C นอกจากจะเป็นขาพอร์ตอินพุตเอาต์พุตดิจิตอลแล้ว ยังมีคุณสมบัติสำคัญดังนี้

- ใช้งานเป็นขาพอร์ตสำหรับเชื่อมต่อหน่วยความจำภายนอก โดยเป็นขาแอดเดรส 8 บิตบน (บิตที่ 8 ถึง 15) ทำงานร่วมกับขา PORT A ดังที่กล่าวมาแล้วข้างต้น

ตารางที่ 2.7 แสดงขา PORT D (PD0-PD7)

ขาพอร์ต	ขาพอร์ตฟังก์ชันพิเศษ
PD7	T2 (อินพุตสัญญาณ สำหรับไทเมอร์/เคาน์เตอร์ 2)
PD6	T1 (อินพุตสัญญาณ สำหรับไทเมอร์/เคาน์เตอร์ 1)
PD5	XCK1 (ขาอินพุตเอาต์พุตสัญญาณนาฬิกาภายนอกสำหรับโมดูล USART1)
PD4	ICP1 (ขาอินพุตตรวจจับสัญญาณของ ไทเมอร์/เคาน์เตอร์ 1)
PD3	INT3/TXD1 (ขาอินพุตอินเตอร์รัปต์เนื่องจากสัญญาณภายนอกช่องที่ 3 หรือขาส่งข้อมูลโมดูล UART1)
PD2	INT2/RXD1 (ขาอินพุตอินเตอร์รัปต์เนื่องจากสัญญาณภายนอกช่องที่ 2 หรือขารับข้อมูลโมดูล UART1)
PD1	INT1/SDA (ขาอินพุตอินเตอร์รัปต์เนื่องจากสัญญาณภายนอกช่องที่ 1 หรือขาข้อมูลอนุกรมสำหรับโมดูล TWI)
PD0	INT0/SCL (ขาอินพุตอินเตอร์รัปต์เนื่องจากสัญญาณภายนอกช่องที่ 0 หรือขาสัญญาณนาฬิกาสำหรับโมดูล TWI)

นอกเหนือจากตารางที่ 2.7 สามารถกำหนดเพิ่มเติมได้ดังนี้

ขา PORT D นอกจากจะใช้เป็นขาพอร์ตอินพุตเอาต์พุตดิจิตอลขนาด 8 บิตแล้ว ยังมีคุณสมบัติที่สำคัญดังนี้

- ขาพอร์ตอินเตอร์รัปต์เนื่องจากสัญญาณภายนอกช่องที่ 0 ถึง 3

- ขาพอร์ตสำหรับอินพุตสัญญาณนาฬิกาจากภายนอกของไทเมอร์เคาน์เตอร์ 1 และ 2 หรือที่เรียกว่าการทำงานในโหมดเคาน์เตอร์

- ขาตรวจจับสัญญาณอินพุต (Input Capture) ของไทเมอร์/เคาน์เตอร์ 1 เมื่อตรวจจับสัญญาณได้จะใช้ได้เวลาในการจับสัญญาณมา สามารถนำมาประยุกต์ใช้ในการจับสัญญาณความถี่หรือหาความกว้างของสัญญาณ PWM ได้ เป็นต้น

- ขาพอร์ตสำหรับโมดูลอนุกรม USART1 ช่องที่ 1 เพื่อใช้ในการสื่อสารข้อมูลอนุกรมกับอุปกรณ์ภายนอก แบบ RS-232 ช่องที่ 1

- ขาพอร์ตสำหรับการเชื่อมต่อด้วยสายสัญญาณ 2 เส้น (Two-wire Serial Interface (TWI)) หรือที่รู้จักกันในชื่อ I^2C บัส

ตารางที่ 2.8 แสดงขา PORT E (PE0-PE7)

ขาพอร์ต	ขาพอร์ตฟังก์ชันพิเศษ
PE7	INT7/ICP3 (ขาอินพุตอินเตอร์รัปต์เนื่องจากสัญญาณภายนอกช่องที่ 7 หรือขาอินพุตตรวจจับสัญญาณของไทเมอร์/เคาน์เตอร์ 3)
PE6	INT6/T3 (ขาอินพุตอินเตอร์รัปต์เนื่องจากสัญญาณภายนอกช่องที่ 7 หรืออินพุตรับสัญญาณสำหรับไทเมอร์/เคาน์เตอร์ 3)
PE5	INT5/OC3C (ขาอินพุตอินเตอร์รัปต์เนื่องจากสัญญาณภายนอกช่องที่ 5 หรือโมดูลเปรียบเทียบและสร้างสัญญาณเอาต์พุต PWM ช่อง C สำหรับไทเมอร์/เคาน์เตอร์3)
PE4	INT4/OC3B (ขาอินพุตอินเตอร์รัปต์เนื่องจากสัญญาณภายนอกช่องที่ 4 หรือโมดูลเปรียบเทียบและสร้างสัญญาณเอาต์พุต PWM ช่อง B สำหรับไทเมอร์/เคาน์เตอร์3)
PE3	AIN1/OC3A (อินพุตเปรียบเทียบแรงดันอะนาลอกด้านลบ หรือโมดูลเปรียบเทียบและสร้างสัญญาณเอาต์พุต PWM ช่อง A สำหรับไทเมอร์/เคาน์เตอร์3)
PE2	AIN0/XCK0 (อินพุตเปรียบเทียบแรงดันอะนาลอกด้านบวก หรืออินพุตเอาต์พุตสัญญาณนาฬิกาจากภายนอกสำหรับโมดูล USART0)
PE1	PDO/TXD0 (ขาโปรแกรมข้อมูลเอาต์พุต หรือขาส่งข้อมูลโมดูล UART0)
PE0	PDI/RXD0 (ขาโปรแกรมข้อมูลอินพุต หรือขารับข้อมูลโมดูล UART0)

นอกเหนือจากตารางที่ 2.8 สามารถกำหนดเพิ่มเติมได้ดังนี้

ขา PORT E นอกจากจะใช้เป็นขาพอร์ตอินพุตเอาร์พุตดิจิตอลขนาด 8 บิตแล้ว ยังมีคุณสมบัติที่สำคัญดังนี้

- ขาพอร์ตอินเตอร์รัปต์เนื่องจากสัญญาณภายนอกช่องที่ 4 ถึง 7
- ขาตรวจจับสัญญาณอินพุต (Input Capture) ไทเมอร์/เคาน์เตอร์ 3

- ขาพอร์ตสร้างสัญญาณ PWM สำหรับไทเมอร์/เคาน์เตอร์3
- ขาอินพุตด้านบวกและลบสำหรับโมดูลเปรียบเทียบแรงดันอะนาลอก
- ขาพอร์ตสำหรับโมดูลอนุกรม USART0 ช่องที่ 0

ตารางที่ 2.9 แสดงขา PORT F (PF0-PF7)

ขาพอร์ต	ขาพอร์ตฟังก์ชันพิเศษ
PF7	ADC7/TDI (ขาอินพุตสัญญาณอะนาลอกช่องที่ 7 หรือขาทดสอบ JTAG สำหรับอินพุตข้อมูล)
PF6	ADC6/TDO (ขาอินพุตสัญญาณอะนาลอกช่องที่ 6 หรือขาทดสอบ JTAG สำหรับเอาต์พุตข้อมูล)
PF5	ADC5/TMS (ขาอินพุตสัญญาณอะนาลอกช่องที่ 5 หรือขาทดสอบ JTAG สำหรับเลือกโหมด)
PF4	ADC4/TCK (ขาอินพุตสัญญาณอะนาลอกช่องที่ 4 หรือขาทดสอบ JTAG สำหรับสัญญาณนาฬิกา)
PF3	ADC3 (ขาอินพุตสัญญาณอะนาลอกช่องที่ 3)
PF2	ADC2 (ขาอินพุตสัญญาณอะนาลอกช่องที่ 2)
PF1	ADC1 (ขาอินพุตสัญญาณอะนาลอกช่องที่ 1)
PF0	ADC0 (ขาอินพุตสัญญาณอะนาลอกช่องที่ 0)

นอกเหนือจากตารางที่ 2.9 สามารถกำหนดเพิ่มเติมได้ดังนี้

ขา PORT F นอกจากจะเป็นขาพอร์ตอินพุตเอาต์พุตดิจิตอลขนาด 8 บิตแล้ว ยังมีคุณสมบัติสำคัญดังนี้

- ขาพอร์ตสำหรับวงจร JTAG สำหรับการดีบักโปรแกรม
- ขาพอร์ตสำหรับโมดูลแปลงสัญญาณอะนาลอกเป็นดิจิตอลขนาด 10 บิต 8 ช่อง

ตารางที่ 2.10 แสดงขา PORT G (PG0-PG7)

ขาพอร์ต	ขาพอร์ตฟังก์ชันพิเศษ
PG4	TOSC1 (ขา RTC ออสซิเลเตอร์ 1 สำหรับไทเมอร์ / เคาเตอร์ 0)
PG3	TOSC2 (ขา RTC ออสซิเลเตอร์ 2 สำหรับไทเมอร์ / เคาเตอร์ 0)
PG2	ALE (ขาแลซ์แอดเดรสสำหรับเอ็นเอเบิลหน่วยความจำภายนอก)
PG1	$\overline{RD}$ (ขาสโตป สัญญาณอ่านข้อมูลหน่วยความจำภายนอก)
PG0	$\overline{WD}$ (ขาสโตป สัญญาณเขียนข้อมูลหน่วยความจำภายนอก)

นอกเหนือจากตารางที่ 2.10 สามารถกำหนดเพิ่มเติมได้ดังนี้

ขา PORT G นอกจากจะใช้เป็นขาพอร์ตอินพุตเอาต์พุตดิจิตอลขนาด 5 บิตแล้ว ยังมีคุณสมบัติสำคัญดังนี้

- ขาพอร์ตออสซิเลเตอร์สำหรับโมดูลสัญญาณนาฬิกาเวลาจริง Real-Time Clock (RTC)
- ขาพอร์ตสำหรับอ่าน-เขียนหน่วยความจำภายนอก ทำงานร่วมกับขา PORT A และ PORT C

พอร์ตอินพุตและพอร์ตเอาต์พุตดิจิตอลของ ATmega128 ดังภาพที่ 2.3 และ 2.4 มีคุณสมบัติแบบเดียวกับ ATmega16 โดยกำหนดคุณสมบัติของขาพอร์ตผ่านทางรีจิสเตอร์ DDRx และ PORTx รวมถึงบิต PUD ในรีจิสเตอร์ SFIOR แสดงดังตารางที่ 2. 11

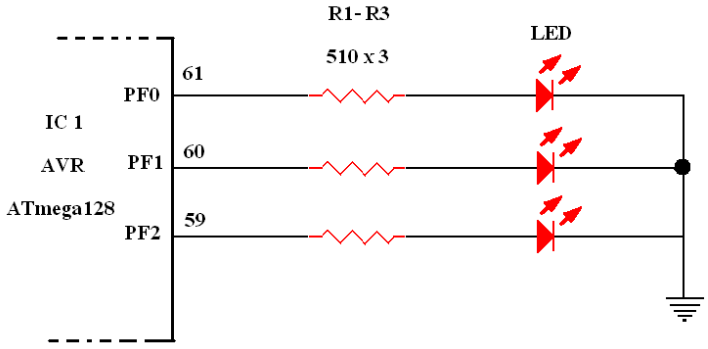
ตารางที่ 2.11 การกำหนดรูปแบบของขาพอร์ต

DDxn	PORTxn	PUD (in SFIOR)	I/O	Pull-Up	สถานะ
0	0	X	Input	No	Tri-state (Hi-Z)
0	1	0	Input	Yes	Pxn will source current if ext. pulled low
0	1	1	Input	No	Tri-state (Hi-Z)
1	0	X	Output	No	Output Low (Sink)
1	1	X	Output	No	Output High (Source)

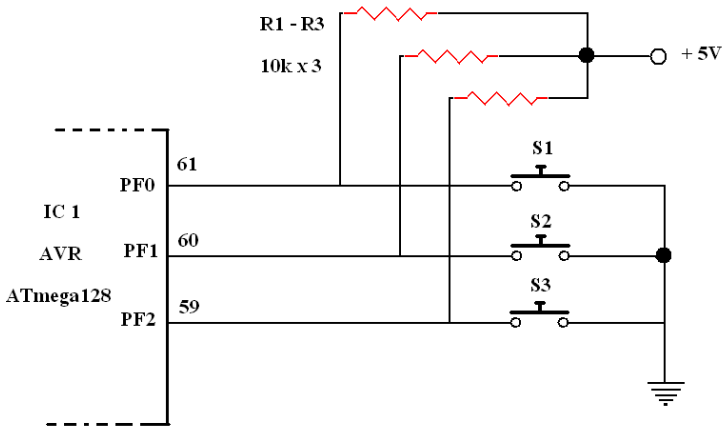
นอกเหนือจากตารางที่ 2.11 สามารถกำหนดเพิ่มเติมได้ดังนี้

(ก) DDxn และ PORTxn เป็นชื่อบิตของรีจิสเตอร์ DDRx และ PORTx โดย x แทนด้วย A, B, C, D … และ n แทนด้วยตัวเลข 0 ถึง 7

(ข) หากใช้งานพอร์ตเป็นเอาต์พุต จะต้องไม่ลืมค่าลอจิก “1” ให้กับรีจิสเตอร์ DDxn เพื่อให้ขาพอร์ตสามารถขับกระแสได้เต็มที่



ภาพที่ 2.3 การใช้งานพอร์ตเอาต์พุตของ AVR ATmega128



ภาพที่ 2.4 การใช้งานพอร์ตอินพุตของ AVR ATmega128

2.2 การเขียนโปรแกรมภาษาซีเบื้องต้น [3]

1. พื้นฐานโปรแกรมภาษาซี

พื้นฐานเกี่ยวกับคอมพิวเตอร์ หน่วยสำคัญที่สุดของคอมพิวเตอร์ก็คือ หน่วยประมวลผล หรือที่เรียกกันว่า CPU โดยปกติ CPU จะมีภาษาของตัวเองที่เรียกว่า ภาษาเครื่อง (Machine Language) ซึ่งจะเป็นภาษาที่ประกอบไปด้วยเลขฐานสองมากมาย ดังนั้นการที่จะเขียนโปรแกรมควบคุมการทำงานของคอมพิวเตอร์ โดยใช้ภาษาเครื่องโดยตรงนั้นจึงทำได้ยาก จึงได้มีการพัฒนาตัวแปรภาษาเครื่องที่เรียกว่า โปรแกรมภาษาระดับสูงขึ้นมา หรือที่เรียกว่า High Level Languages โดยภาษาในระดับสูงเหล่านี้ จะมีลักษณะรูปแบบการเขียน (Syntax) ที่ทำให้เข้าใจได้ง่ายต่อการสื่อสารกับผู้พัฒนา และถูกออกแบบมาให้ง่ายต่อการใช้งาน และจะเปลี่ยนคำสั่งจากผู้ใช้งาน ไปเป็นเป็นภาษาเครื่อง เพื่อที่จะควบคุมการทำงานของคอมพิวเตอร์ต่อไป ตัวอย่างของโปรแกรมภาษาระดับสูง ได้แก่ COBOL ใช้กันมากสำหรับโปรแกรมทางด้านธุรกิจ, Fortran ใช้กันมากสำหรับการพัฒนาโปรแกรมด้านวิทยาศาสตร์และวิศวกรรมศาสตร์ เพราะง่ายต่อการคำนวณ, Pascal มีใช้กันทั่วไป แต่เน้นสำหรับการพัฒนาเครื่องมือสำหรับการเรียนการสอน, C & C++ ใช้ทั่วไป ปัจจุบันมีผู้เลือกที่จะใช้กันอย่างแพร่หลาย, PROLOG เน้นหนักไปทางด้านงานประเภท AI และ JAVA ใช้ได้ทั่วไป ปัจจุบันเริ่มมีผู้หันมาสนใจกันมากและเพิ่มขึ้นอย่างรวดเร็ว

2. ตัวแปร

ตัวแปรจะเป็นชื่อที่ใช้ในการบอกจำนวนหรือปริมาณ ซึ่งสามารถที่จะทำการเปลี่ยนแปลงจำนวนได้ด้วยโปรแกรมคอมพิวเตอร์ การตั้งชื่อตัวแปร จะต้องตั้งชื่อให้แตกต่างไปจากชื่อของตัวแปรอื่นๆ ยกตัวอย่างชื่อของตัวแปร ได้แก่ x, y, peter, num_of_points และ streetnum เป็นต้น โดยปกติการเขียนโปรแกรมที่ดี ควรจะตั้งชื่อตัวแปรให้สอดคล้องกับการทำงานหรือหน้าที่ของตัวแปรนั้นๆ เพราะเมื่อถึงเวลาต้องมาทำการปรับปรุงแก้ไขโปรแกรม จะสามารถทำได้โดยไม่ยากนัก ในภาษา C หรือ C++ ได้มีกฎในการตั้งชื่อตัวแปรที่สามารถใช้งานได้ดังนี้

- ชื่อตัวแปรจะต้องขึ้นต้นด้วยตัวอักษร
- ชื่อตัวแปรจะประกอบไปด้วย ตัวอักษร ตัวเลข และ _ ได้เท่านั้น
- ชื่อตัวแปรจะต้องไม่ใช่ชื่อ reserved word (ชื่อที่มีการจองไว้แล้ว)

ตัวอย่างของชื่อตัวแปรที่สามารถนำมาใช้ตั้งชื่อได้ ได้แก่ length, days_in_year, DataSet1, Profit95, Pressure, first_one และตัวอย่างของชื่อ ที่ไม่สามารถนำมาใช้เป็นชื่อตัวแปรได้ ยกตัวอย่าง เช่นay-in-year, 1data, int, first.val เป็นต้น

reserved word (ชื่อที่มีการจองไว้แล้ว)Reserved words หรือตัวแปรที่ได้จองไว้แล้วนั้น จะประกอบไปด้วยตัวอักษรตัวเล็กทั้งหมด และจะมีความสำคัญสำหรับภาษา C++ และจะไม่นำมาใช้ด้วยวัตถุประสงค์อื่นๆ ตัวอย่างของ Reserved words ได้แก่ and, bool, break, case, catch, char, class, continue, default, delete, do, double, if , else, enum, export, extern เป็นต้น

นอกจากนี้ในภาษา C หรือ C++ ชื่อตัวแปร ที่ประกอบไปด้วยอักษรเล็ก หรือใหญ่ ก็มีความแตกต่างกัน หรือที่เรียกว่า Case sensitive ยกตัวอย่างเช่น‘X’ และ ‘x’ เป็นตัวแปรต่างกัน
‘peter’ และ ‘Peter’ เป็นตัวแปรต่างกัน‘bookno1’ และ ‘bookNo1’ เป็นตัวแปรต่างกัน
‘XTREME’ และ ‘xtreme’ เป็นตัวแปรต่างกัน‘X1’ และ ‘x1’ เป็นตัวแปรต่างกัน
‘int’ และ ‘Int’ เป็นตัวแปรต่างกัน

ในภาษา C หรือ C++ (และโปรแกรมในภาษาอื่นๆ) ตัวแปรทุกตัวที่จะมีการเรียกใช้ในโปรแกรมจำเป็นต้องมีการกำหนดชนิดของตัวแปรนั้นๆ ก่อนที่จะทำการเรียกใช้ตัวแปร การกำหนดชนิดของตัวแปรมีวัตถุประสงค์หลัก 2 ประการได้แก่

- เป็นการบอกชนิด และตั้งชื่อตัวแปรที่จะเรียกใช้ ชนิดของตัวแปรจะทำให้คอมไพเลอร์สามารถแปลคำสั่งได้อย่างถูกต้อง (ยกตัวอย่างเช่น ใน CPU คำสั่งที่ใช้ในการบวกตัวเลขจำนวนเต็ม 2 จำนวน ย่อมแตกต่างจากคำสั่งที่จะบวกจำนวนจริง 2 จำนวนเข้าด้วยกัน)

- ชนิดของตัวแปร ยังเป็นบ่งบอกคอมไพเลอร์ให้ทราบว่าจะต้องจัดเตรียมเนื้อที่ให้กับตัวแปรตัวนั้นมากน้อยเท่าใด และจะจัดวางตัวแปรนั้นไว้แอดเดรส (Address) ไหนที่สามารถเรียกมาใช้ใน code ได้

สำหรับในบทความนี้จะพิจารณาชนิดตัวแปร 4 ชนิดที่ใช้กันมากได้แก่ int, float, bool และ char

int ชนิดตัวแปรที่สามารถแทนค่าจำนวนเต็มได้ทั้งบวกและลบ โดยปกติสำหรับคอมพิวเตอร์ทั่วไป คอมไพเลอร์ จะจองเนื้อที่ 2 ไบต์ สำหรับตัวแปรชนิด int จึงทำให้ค่าของตัวแปรมีค่าตั้งแต่ -32768 ถึง +32768 ตัวอย่างของค่า int ได้แก่ 123 -56 0 5645 เป็นต้น

float ชนิดของตัวแปรที่เป็นตัวแทนของจำนวนจริง หรือตัวเลขที่มีค่าทศนิยม ความละเอียดของตัวเลขหลังจุดทศนิยมขึ้นอยู่กับระบบคอมพิวเตอร์ โดยปกติแล้ว ตัวแปรชนิด float จะใช้เนื้อที่ 4 ไบต์ นั่นคือจะให้ความละเอียดของตัวเลขหลังจุดทศนิยม 6 ตำแหน่ง และมีค่าอยู่ระหว่าง -1038 ถึง +1038 ตัวอย่างของค่า float ได้แก่ 16.315 -0.67 31.567

bool ชนิดของตัวแปรที่สามารถเก็บค่าลอจิก จริง (True) หรือ เท็จ (False) ตัวแปรชนิดนี้เป็นที่รู้จักกันอีกชื่อคือ ตัวแปรบูลีน (Boolean) ตัวอย่างของตัวแปรชนิด bool ได้แก่ 1,0 (เมื่อ 1 = true และ 0 = false)

char เป็นชนิดตัวแปรที่เป็นตัวแทนของ ตัวอักษรเพียงตัวเดียว อาจเป็นตัวอักษร ตัวเลข หรือตัวอักขระพิเศษ โดยปกติตัวแปรชนิดนี้จะใช้เนื้อที่เพียง 1 ไบต์ ซึ่งจะให้ตัวอักษรในรูปแบบที่แตกต่างกันได้ถึง 256 ค่า การเขียนรูปแบบของ char หลายๆ ตัว โดยปกติ จะอ้างอิงกับ American Standard Code for Information Interchange (ASCII)

3. การเขียนโปรแกรมแบบมีทางเลือก

การเขียนโปรแกรมแบบมีทางเลือก จะสามารถทำให้โปรแกรมสามารถตัดสินใจหรือเปรียบเทียบ จากนั้นก็จะเลือกดำเนินการไปในทิศทางหนึ่งจากสองทิศทาง ขึ้นอยู่กับผลที่ได้จากการเปรียบเทียบนั้น

3.1 เงื่อนไข (Condition)

- เป็นตัวกำหนดเงื่อนไขที่ผู้พัฒนาโปรแกรมได้สร้างขึ้นมา
- ผลลัพธ์ที่ได้จากเงื่อนไข จะมีค่า จริงหรือ เท็จ

3.2 โครงสร้างของเงื่อนไข (Condition Control Structures)

ประโยคเงื่อนไขสามารถที่จะเขียนให้อยู่ในรูปภาษา C จะเขียนได้ดังนี้

if condition then A else B ซึ่งหมายความว่า ถ้าเงื่อนไข (condition) มีค่าเป็นจริง ก็จะดำเนินการทำคำสั่ง A มิเช่นนั้นก็จะทำคำสั่ง B

ตัวอย่างของการเขียนโครงสร้างทางเลือกในภาษา C สามารถเขียนได้ดังนี้

```
if (x < y)
    a = x * 2;
else
    a = x + y;
```

ความหมายของ code ดังกล่าว หมายความว่า ถ้า ค่า x มีค่าน้อยกว่า y แล้ว a = x*2 แต่ถ้า x มีค่ามากกว่าหรือเท่ากับ y แล้ว a = x+y รูปแบบของเงื่อนไข ส่วนใหญ่จะอยู่ในรูป “ตัวแปร โอเปอเรเตอร์ ตัวแปร” โอเปอเรเตอร์ที่กล่าวถึงนี้จะมีอยู่ 2 แบบ ด้วยกันคือ โอเปอเรเตอร์สัมพันธ์ (Relational Operator) และ โอเปอเรเตอร์ลอจิก (Logical Operator) โอเปอเรเตอร์สัมพันธ์ที่ใช้ในภาษา C มีดังต่อไปนี้

ตารางที่ 2.12 ตัวดำเนินการโอเปอเรเตอร์

โอเปอร์เรเตอร์	ความหมาย	ยกตัวอย่าง
<	น้อยกว่า	5<4 เป็นเท็จ
>	มากกว่า	5>4 เป็นจริง
<=	น้อยกว่าหรือเท่ากับ	3<=3 เป็นเท็จ
>=	มากกว่าหรือเท่ากับ	3>=3 เป็นจริง
==	เท่ากับ	2==5 เป็นเท็จ
!=	ไม่เท่ากับ	2!=5 เป็นจริง

3.3 การเขียนโปรแกรมแบบวนซ้ำ (Repetition & Loop)

กระบวนการหนึ่งที่สำคัญในการออกแบบอัลกอริทึม ก็คือความสามารถในการวนLoop ของการทำงานของกลุ่มคำสั่งตามที่นักพัฒนาต้องการ ดังนั้นสำหรับตอนนี้ ก็จะนำเสนอการพัฒนาโปรแกรมเพื่อให้บางส่วนของคำสั่งสามารถมีการวนซ้ำได้หลายครั้ง สำหรับคำสั่งที่สามารถใช้ในการเขียนโปรแกรมแบบวนซ้ำในภาษา C ได้แก่ While, Do-while และ For

3.3.1 โครงสร้างการเขียนโปรแกรมแบบวนซ้ำโดยใช้คำสั่ง For

Loop for จะเริ่มด้วยการนำ ค่าเริ่มต้นเปรียบเทียบกับเงื่อนไขที่กำหนดไว้ ถ้าเงื่อนไขเป็นจริง จะทำ statement ในLoop ถ้า เป็นเท็จ จะเลิกทำ งานในLoop กรณีเมื่อตรวจสอบเงื่อนไขแล้วเป็นจริงเมื่อทำ งานในLoopแล้ว ก็จะเพิ่มหรือลดค่าตัวแปรในเงื่อนไขอีก 1 หรือมากกว่าตามที่กำหนดไว้โดยอัตโนมัติ แล้วตรวจสอบเงื่อนไขอีกครั้ง ถ้าเงื่อนไข เป็นเท็จ จะเลิกทำ ซ้ำในLoop มีรูปแบบ Statement ดังนี้

for (เริ่มต้น ; เงื่อนไข ; เปลี่ยนแปลง)

statement;

เมื่อเริ่มต้น เป็นการกำหนดค่าตัวแปรเริ่มต้นที่ต้องการ ส่วนเงื่อนไขหากค่าลอจิกมีค่าเป็นจริง ก็จะทำตามในโครงสร้างของการวนซ้ำคือ run คำสั่ง Statement แต่ถ้าเป็นเท็จก็จะออกจากโครงสร้างการวนซ้ำ ส่วนเปลี่ยนแปลง จะทำการปรับค่าของตัวแปรที่ต้องการ ยกตัวอย่างเช่น

for (count=0 ; count < 10 ; count++)

3.3.2 โครงสร้างการเขียนโปรแกรมแบบวนซ้ำโดยใช้คำสั่ง while

Loop while จะเริ่มการทำ งานด้วยการทดสอบเงื่อนไขที่กำหนดไว้ต้นLoopก่อนเสมอ ถ้าเงื่อนไขเป็นจริง จะทำ งานซ้ำในLoop แต่ถ้าเงื่อนไขเป็น เท็จ จะเลิกทำงานในLoop (ทำงานในLoopขณะที่เงื่อนไขเป็นจริงเท่านั้น) รูปแบบของ while มีดังนี้

Loop while ที่มี statement เดียว มีรูปแบบดังนี้

```
while(เงื่อนไขเปรียบเทียบ)
statement;
```

Loop while ที่มีหลาย statement มีรูปแบบดังนี้

```
while(เงื่อนไขเปรียบเทียบ)
{ statement;
statement;
statement;
}
```

Loop while ซ้อนกัน มีรูปแบบ ดังนี้

```
while(เงื่อนไขเปรียบเทียบของLoopนอก)
{ statement;
statement;
    while(เงื่อนไขเปรียบเทียบของLoopใน)
        { statement;
          statement;
          statement;
        }
statement;
statement;
}
```

3.3.3 โครงสร้างการเขียนโปรแกรมแบบวนซ้ำโดยใช้คำสั่ง do - while

Loop do จะเริ่มด้วยการทำงานรอบแรก 1 รอบก่อนเสมอ และมีการทดสอบเงื่อนไขที่ท้าย Loop ถ้าเงื่อนไขเป็น จริง จะมีการทำ ซ้ำในLoop แต่ถ้าเงื่อนไขเป็น เท็จ จะเลิกทำงานในLoop มีรูปแบบดังนี้

Loop do ที่มี statement เดียว มีรูปแบบดังนี้

```
do
statement;
while(เงื่อนไขเปรียบเทียบ);
```

Loop do ที่มีหลาย statement มีรูปแบบดังนี้

```
do{
  statement;
    statement;
  statement;
}while(เงื่อนไขเปรียบเทียบ);
```

Loop do ซ้อนกัน มีรูปแบบดังนี้

```
do{
    statement;
    statement;
    do{
        statement;
        statement;
    }while(เงื่อนไขเปรียบเทียบLoopใน);
    statement;
}while(เงื่อนไขเปรียบเทียบLoopนอก);
```

4. ฟังก์ชัน

ความหมายของส่วนประกอบในฟังก์ชัน มีดังนี้

function_type คือ ประเภทค่าหรือข้อมูลที่ได้จากการทำงานของฟังก์ชัน ซึ่งจะต้องให้ค่าคืนกลับมาเก็บไว้ในชื่อของฟังก์ชัน (function_name) ถ้าเป็นประเภท void จะเป็นฟังก์ชันประเภทที่ต้องมีการ return value

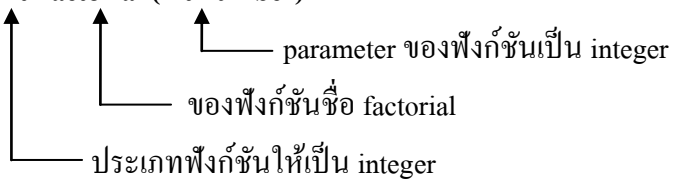
function_name คือ ชื่อฟังก์ชันที่กำ หนดขึ้นตามกฎเกณฑ์การตั้งชื่อของ C++ และจะเป็นชื่อที่ใช้สำ หรับการเรียกใช้ฟังก์ชันต่อไป

(parameter1,parameter2,…) หรือพารามิเตอร์ประกอบไปด้วย ประเภทและชื่อของตัวแปรใช้รับค่าคงที่เพื่อนำ มาใช้ทำ งานในฟังก์ชัน ในขณะที่ฟังก์ชันนั้นถูกเรียกใช้ทำ งาน พารามิเตอร์ของ ฟังก์ชันมีมากกว่า 1 ตัวและหลายประเภทได้

return (value); โดยที่ return เป็นคีย์เวิร์ด และ value คือค่าคงที่ที่ส่งคืนไปให้แก่ชื่อฟังก์ชัน 1 ค่าหรือไม่มีก็ได้ (กรณีเป็นฟังก์ชันที่ไม่ให้ค่าใด ๆ)

ตัวอย่างการกำหนดชื่อฟังก์ชัน เช่น

int factorial (int number)



4.1 การประกาศชื่อฟังก์ชันที่ผู้ใช้กำหนดไว้ที่ตอนต้นโปรแกรมก่อนฟังก์ชัน main() เรียกว่าฟังก์ชันต้นแบบ (prototype) หรือการกำ หนดฟังก์ชัน (function declaration) แสดงว่าในโปรแกรมมีการสร้างและเรียกใช้ฟังก์ชันเหล่านี้ ซึ่งสามารถเรียกใช้ชื่อฟังก์ชันนั้น ณ ตำ แหน่งใด ๆ ของโปรแกรมก็ได้ ยกเว้นห้าม เรียกใช้ชื่อฟังก์ชันนั้นในตัวฟังก์ชันเอง การประกาศฟังก์ชันต้นแบบ (prototype) ที่มีพารามิเตอร์(parameters) ทำ ได้ 2 ลักษณะ คือ

4.1.1 ในส่วนของพารามิเตอร์ของฟังก์ชัน เขียนทั้งชนิดข้อมูลและตัวพารามิเตอร์ เช่น int factorial(int number);

4.1.2 ในส่วนของพารามิเตอร์ของฟังก์ชัน เขียนเฉพาะชนิดข้อมูล เช่น

int factorial(int);

4.2 การเรียกใช้ฟังก์ชันมีรูปแบบการเรียกใช้เสมือนเป็น statement เช่น line()factorial(x) โดยที่ฟังก์ชันใดที่มีพารามิเตอร์กำ หนดไว้ ต้องใส่ค่า argument ให้ถูกต้องในขณะเรียกใช้ด้วย และเรียก xของฟังก์ชัน factorial() ว่าอากิวเมนต์ (argument)

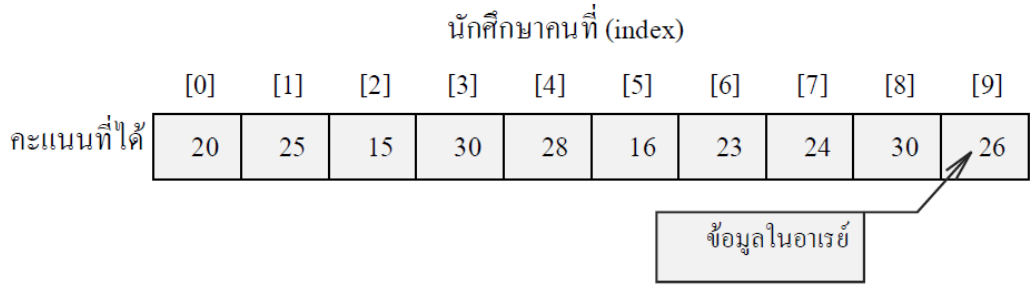
4.3 เมื่อมีการเรียกใช้ฟังก์ชัน ณ ตำ แหน่งใด โปรแกรมจะไปทำ งานในฟังก์ชันนั้นจนเสร็จ แล้วจะกลับมาทำ งานต่อใน statement ถัดไปจากตำ แหน่งที่เรียกใช้ฟังก์ชัน

4.4 สามารถเรียกใช้ฟังก์ชันกี่ครั้งก็ได้ในโปรแกรม

5. ตารางอาเรย์

อาร์เรย์ (array) เป็นข้อมูลชนิดมีโครงสร้าง (structure) หมายถึงเป็นข้อมูลชุดหนึ่งมีจำนวนสมาชิกข้อมูลแน่นอนและต้องเป็นชนิดเดียวกันทั้งชุดแต่ละรายการของข้อมูลในชุดนั้นๆ เรียกว่า สมาชิกของอาร์เรย์ (element of array) สมาชิกแต่ละตัวมีตัวชี้ตำแหน่งที่อยู่เรียกว่า อินเด็กซ์ (index) เพื่อใช้สำหรับการอ้างอิงถึงข้อมูลแต่ละสมาชิก โดยอินเด็กซ์เป็นข้อมูลชนิดที่มีลำดับ เช่น integer ตัวอย่างข้อมูลชนิดอาร์เรย์ 1 มิติ เช่น คะแนนสอบของนักศึกษา 10 คน เป็นตัวเลขประเภท int กำหนดเป็นโครงสร้างของอาร์เรย์ดังภาพที่ 2.5

```
int score[10];
```



ภาพที่ 2.5 โครงสร้างของอาเรย์

ความหมายของข้อมูลในอาร์เรย์โดยใช้ index เป็นตัวชี้ข้อมูลในแต่ละช่อง เป็นดังนี้

อาร์เรย์ช่องที่ [0] มีค่าคะแนนนักศึกษา คนที่ 1 เก็บอยู่ คือ 20

อาร์เรย์ช่องที่ [1] มีค่าคะแนนนักศึกษา คนที่ 2 เก็บอยู่ คือ 25

อาร์เรย์ช่องที่ [2] มีค่าคะแนนนักศึกษา คนที่ 3 เก็บอยู่ คือ 15

อาร์เรย์ช่องที่ [3] มีค่าคะแนนนักศึกษา คนที่ 4 เก็บอยู่ คือ 30

อาร์เรย์ช่องที่ [4] มีค่าคะแนนนักศึกษา คนที่ 4 เก็บอยู่ คือ 28

อาร์เรย์ช่องที่ [5] มีค่าคะแนนนักศึกษา คนที่ 4 เก็บอยู่ คือ 16

อาร์เรย์ช่องที่ [6] มีค่าคะแนนนักศึกษา คนที่ 4 เก็บอยู่ คือ 23

อาร์เรย์ช่องที่ [7] มีค่าคะแนนนักศึกษา คนที่ 4 เก็บอยู่ คือ 24

อาร์เรย์ช่องที่ [8] มีค่าคะแนนนักศึกษา คนที่ 4 เก็บอยู่ คือ 30

อาร์เรย์ช่องที่ [9] มีค่าคะแนนนักศึกษา คนที่ 10 เก็บอยู่ คือ 26

ถ้าคะแนนที่เก็บเป็นข้อมูลชนิด int จะจองพื้นที่หน่วยความจำ ในการเก็บข้อมูล 2 byte * 10 ช่อง เท่ากับ 20 byte

การเรียกใช้จะได้

```
score [0] = 20;
score[1] = 25;
score[2] = 15;
score[9 ]= 26;
```

2.3 คำสั่ง AT-Command [4]

AT-Command คือ ชุดคำสั่งที่ใช้สำหรับเชื่อมต่อระหว่างอุปกรณ์สื่อสารต่างๆ เช่น โมเด็ม หรืออุปกรณ์ DTE (Data Terminal Equipment) เพื่อโต้ตอบสั่งงานอุปกรณ์เหล่านั้น ให้ทำงานตามที่เราต้องการสำหรับการติดต่อโทรศัพท์มือถือ จะใช้ชุดคำสั่งที่เรียกว่า GSM AT COMMAND ยกตัวอย่างการใช้ AT Command เช่น

AT	เป็นชุดคำสั่งเช็คความพร้อมของโทรศัพท์เคลื่อนที่
ATD 0892870254;	เป็นคำสั่งให้โทรออกไปยังหมายเลข 0892870254
ATH	เป็นคำสั่งให้วางสาย
AT+CMGF	เป็นคำสั่งในการเลือกโหมดการส่ง
AT+CMGS	เป็นคำสั่งการส่ง SMS
AT+CMGD	เป็นคำสั่งลบข้อความออกจากหน่วยความจำ

1. หลักการส่ง SMS [3]

SMS ย่อมาจาก Short Message Service เป็นบริการส่งข้อความสั้นๆจากโทรศัพท์มือถือต้นทางผ่านชุมสายไปยังโทรศัพท์มือถือปลายทาง โดยสามารถส่งได้สูงสุด 160 ตัวอักษร โดยแต่ละตัวอักษรใช้รหัส7 bit และมีการใช้ตัวอักษรชนิดอื่นๆเช่นขนาด 8 bit หรือ 16 bit ซึ่งมีวัตถุประสงค์เพื่อการใช้งานที่แตกต่างกันออกไป ตามข้อกำหนดมาตรฐานขององค์การ ETSI (European Telecommunications Standards Institute)

2. รูปแบบการรับส่งข้อมูล SMS ผ่าน AT Command

มี 2 รูปแบบ คือ Text Mode และ PDU Mode

1. Text Mode เป็นการส่งข้อมูลในรูปของตัวอักษรได้โดยตรง ซึ่งตัวเครื่องส่วนใหญ่ไม่รองรับการส่งข้อมูลรูปแบบนี้ผ่านทาง AT Command จึงไม่สามารถใช้งานได้สมบูรณ์

2. PDU Mode PDU ย่อมาจาก (Protocol Description Unit Mode) จะมีการทำงานคล้ายกับแบบ Text Mode แต่แบบ PDU Mode จะสามารถเลือกหรือสร้างการเข้ารหัสและถอดรหัสได้ทุกรูปแบบตามต้องการโดยไม่มีข้อจำกัด

2.4 การสื่อสารแบบอนุกรม [6]

การสื่อสารแบบอนุกรม คือ การรับ-ส่งข้อมูลระหว่างอุปกรณ์ผ่านสายสัญญาณโดยส่งผ่านข้อมูลไปทีละบิตต่อเนื่องกัน แบ่งได้เป็น 2 แบบ คือ

- **Synchronous** มีการส่งสัญญาณ clock ไปพร้อมกับข้อมูล
- **Asynchronous** ไม่มีการส่งสัญญาณ clock ไปพร้อมกับข้อมูล

ในการสื่อสารรับ-ส่งในระยะไกลนั้น จำเป็นต้องใช้ Data Communication Equipment (DCE) อย่างเช่น Modem เพื่อให้ส่งข้อมูลได้ไกลขึ้น แต่สำหรับการสื่อสารไกลในระยะไม่เกิน 50 ฟุต ไม่จำเป็นต้องใช้ DCE เรียกว่า การสื่อสารแบบ Null-modem และในการติดต่อสื่อสารข้อมูลระหว่างอุปกรณ์ จำเป็นจะต้องมีมาตรฐานในการเชื่อมต่อวงจร ซึ่งมาตรฐานที่ใช้กันมากที่สุดก็คือมาตรฐาน RS232-C

การสื่อสารข้อมูลแบบอนุกรมระหว่างคอมพิวเตอร์กับอุปกรณ์อื่นๆ เกือบทั้งหมดเป็นแบบ Asynchronous ดังนั้น เราจะกล่าวถึงรายละเอียดเกี่ยวกับการสื่อสารข้อมูลแบบ Asynchronous เท่านั้น

- **Asynchronous Serial Communication**

  การส่งข้อมูลจะมีลักษณะเป็น FRAME โดยแต่ละ frame ประกอบด้วย

START | DATA | PARITY | STOP

Start bit จำนวน 1 บิต โดยทั่วไปจะเป็น Low (0)

Data จำนวน 5-8 บิต

Parity bit จำนวน 0-1 บิต เป็นการตรวจสอบความถูกต้องของข้อมูล กำหนดเป็น even หรือ odd

Stop bit จำนวน 1-2 บิต โดยทั่วไปจะเป็น high (1)

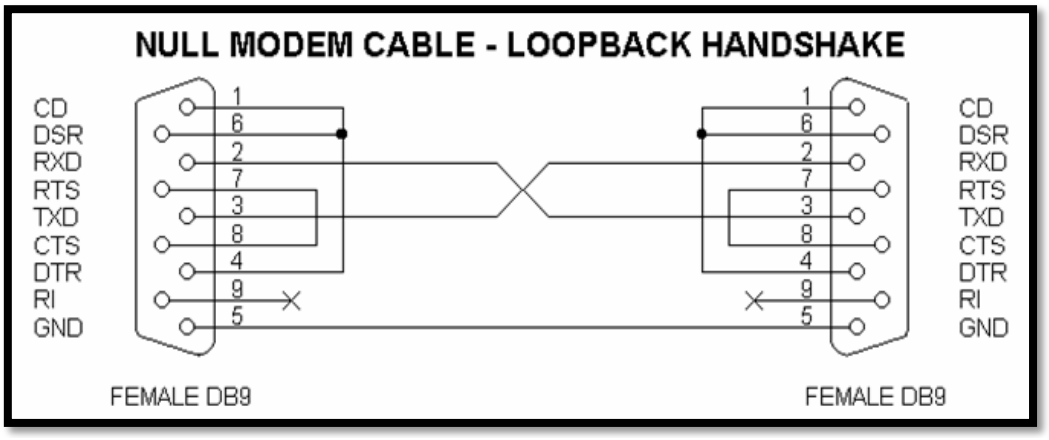
สัญลักษณ์ที่ใช้บอกลักษณะของข้อมูล เช่น 7E1: data 7 bits, even parity, 1 stop bit

8N1: data 8 bits, non-parity, 1 stop bit

เนื่องจากการสื่อสารแบบ Asynchronous ไม่มีการส่งสัญญาณ Clock ไปกับข้อมูล จึงจำเป็นต้องเซตสัญญาณความถี่สัญญาณ Clock ทางด้าน Rx และ Tx ให้เท่ากันเพื่อให้รับ-ส่งข้อมูลระหว่างกันได้ เรียกว่าค่าBit rate หรือ Baud

สำหรับการสื่อสารแบบอนุกรมบน PC รวมถึง Microcontroller จะถูกจัดการโดยส่วนที่เรียกว่า UART(Universal Asynchronous Receiver/Transmitter) ซึ่งจะทำหน้าที่รับ-ส่งข้อมูล Asynchronous โดยแปลงข้อมูลในรูปขนาน เป็นอนุกรม Asynchronous แล้วค่อยส่งออก จากนั้นภาครับก็จะรับอนุกรม Asynchronousมาแปลงเป็นขนานอีกทีหนึ่ง ในการเชื่อมต่อสายสัญญาณเข้ากับอุปกรณ์อนุกรมนิยมใช้หัวต่อชนิด DB9 และDB25 ในที่นี้จะใช้เฉพาะหัวต่อชนิด DB9

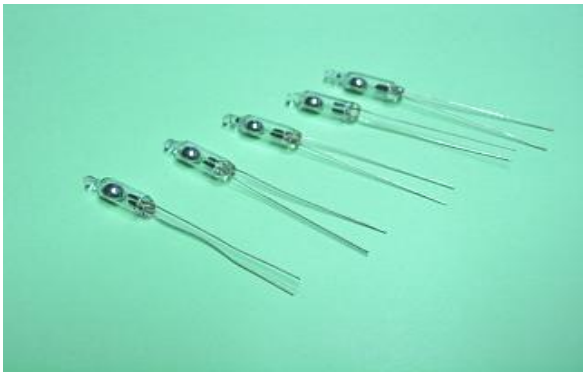
สำหรับสายสัญญาณที่ใช้ในการทดลองในการติดต่อระหว่าง PC กับ Microcontroller คือ Null Modem Cable-Loopback Handshake มี Diagram ดังภาพที่ 2.6



ภาพที่ 2.6 วงจรเชื่อมโยงของ RS232

2.5 สวิทช์ปรอท [7]

สวิทช์ปรอทจะมีลักษณะเป็นกระเปาะแก้วใสดังภาพที่ 2.7 ข้างในจะบรรจุสารปรอทนำไฟฟ้า ซึ่งสามารถไหลไปมาได้ การทำงานจะคล้ายกับเป็นสวิทช์ตัวหนึ่งแต่จะทำงานเมื่อสารปรอทไหลไปด้านที่มีขั้วจะทำให้ขั้วทั้งสองต่อถึงกันทำให้กระแสไหลในวงจร ถ้าสารปรอทไหลไปในทิศทางตรงกันข้ามจะหยุดจ่ายกระแสไฟฟ้าทันที



ภาพที่ 2.7 สวิทช์ปรอท

2.6 จอแสดงผล

จอแสดงผลแบบ LCD (Liquid Crystal Display) จัดเป็นจอแสดงผลอีกรูปแบบหนึ่ง ซึ่งเป็นที่ นิยมนำมาใช้งานกันอย่างแพร่หลายมากขึ้นในปัจจุบัน ซึ่งจอแสดงผลแบบ LCD นี้มีทั้งแบบที่แสดงผลเป็นอักขระเพียงอย่างเดียว (Character LCD) และแบบที่สามารถแสดงผลเป็นรูปภาพหรือสัญลักษณ์ อื่นๆ ตามความต้องการได้ (Graphic LCD) โดย LCD Display ที่เราพบเห็นกันโดยทั่วๆ ไปในชีวิตประจำวันนี้อาจมีอยู่หลายแบบ บางชนิดก็เป็นแบบที่มีการสั่งผลิตขึ้นเฉพาะงานโดยมีรูปแบบและรูปร่างเฉพาะ เช่น LCD Display ที่นำไปใช้ในนาฬิกาข้อมือแบบดิจิตอล เครื่องเล่นเกม เครื่องคิดเลข หรือหน้าปัดวิทยุแบบต่างๆ เป็นต้น แต่ในที่นี้จะกล่าวถึงเฉพาะ Dot-Matrix LCD ที่มีวางจำหน่ายกันทั่วๆไปที่สามารถซื้อหามาใช้งานกันได้ง่าย โดยที่พบเห็นกันทั่วไปได้แก่ขนาด 16 ตัวอักษรไปจนถึง 40 ตัวอักษร และมีจำนวนบรรทัดตั้งแต่ 1 บรรทัดไปจนถึง 4 บรรทัด ดังภาพที่ 2.8โดย LCD เหล่านี้อาจมีหลายผู้ผลิต แต่ส่วนมากแล้วจะมีโครงสร้างการทำงาน

และชุดคำสั่งที่เหมือนกันเกือบทุกประการ อาจมีแตกต่างกันบ้างในเรื่องของความเร็วในการอ่าน/เขียน

1. โครงสร้างโดยทั่วไปของ LCD

โดยปกติโครงสร้างของ LCD จะประกอบขึ้นด้วยแผ่นแก้ว 2 แผ่นประกบกันอยู่ โดยเว้นช่องว่างตรงกลางไว้ 6-10 ไมโครเมตร ผิวด้านในของแผ่นแก้วจะเคลือบด้วยตัวนำไฟฟ้าชนิดใสเพื่อใช้แสดงตัวอักษร ระหว่างตัวนำไฟฟ้าใสกับผลึกเหลวจะมีชั้นสารที่ทำให้โมเลกุลของผลึกรวมตัวกันในทิศทางที่แสงส่องมากระทบซึ่งเรียกว่า Alignment Layer และผลึกเหลวที่ใช้โดยทั่วไปจะเป็นแบบ Magnetic โดย LCD สามารถแสดงผลให้เรามองเห็นได้ทั้งหมด 3 แบบด้วยกันคือ

- แบบใช้การสะท้อนแสง (Reflective Mode) ซึ่งจะใช้สารประเภทโลหะเคลือบอยู่ที่แผ่นหลังของ LCD ซึ่ง LCD ประเภทนี้เหมาะกับการนำมาใช้งานในที่ๆ มีแสงสว่างเพียงพอ
- แบบใช้การส่งผ่าน (Transitive Mode) โดย LCD แบบนี้จะวางหลอดไฟไว้ด้านหลังจอ เพื่อทำให้การอ่านค่าแสดงผลทำได้ชัดเจน
- แบบส่งผ่าน/สะท้อน (Transflective Mode) LCD แบบนี้จะเป็นการนำเอาข้อดีของจอแสดงผล LCD ทั้ง 2 แบบมารวมกัน

2. การควบคุมการแสดงผลของ LCD

ผู้ใช้งานไม่สามารถจ่ายกระแสไฟฟ้าตรงให้กับ LCD ค้างไว้ตลอดเวลาเพื่อให้ LCD แสดงผลตามที่ต้องการได้เนื่องจากจะทำให้เกิดปฏิกิริยาไฟฟ้าเคมีขึ้นและจะทำให้อายุการใช้งาน LCD สั้นลงด้วยเหตุนี้จึงจำเป็นต้องป้อนสัญญาณสลับระหว่างปิดกับเปิด (SCAN) ด้วยความถี่ไม่น้อยกว่า 30 Hz เพื่อไม่ให้หน้าจอกระพริบ LCD โดยทั่วไปจะเป็นแบบที่มีส่วนควบคุม (Controller) รวมไว้ในตัวอยู่แล้วผู้ใช้งานเพียงส่งรหัสคำสั่งสำหรับควบคุมการทำงานของ LCD ให้กับ Controller ว่าต้องการให้ทำงานอย่างไร ซึ่งส่วนใหญ่แล้วจะมีสัญญาณในการเชื่อมต่อระหว่าง LCD กับ Microcontroller ดังนี้

1. GND เป็น Ground ใช้ต่อระหว่าง Ground ของระบบ Microcontroller กับ LCD
2. VCC เป็นไฟเลี้ยงวงจรที่ต้องป้อนให้กับ LCD มีขนาด +5VDC
3. VO เป็นขาสำหรับปรับความสว่างของหน้าจอ LCD

4. RS ใช้สำหรับบอกให้ LCD Controller ทราบว่า Code ที่ส่งให้ทางขา Data เป็นคำสั่งหรือข้อมูล

5. R/W ใช้สำหรับกำหนดว่าจะอ่านหรือเขียนข้อมูลกับ LCD Controller

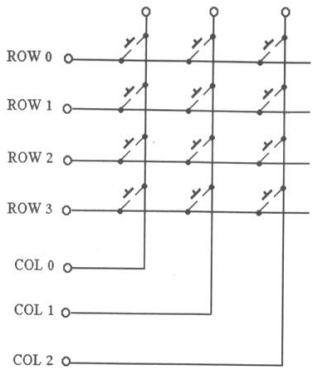
6. E เป็นขา Enable หรือ Chips Select เพื่อกำหนดการทำงานให้กับ LCD Controller

7-14. DB0-DB7 เป็นขาสัญญาณ Data ใช้สำหรับเขียนหรืออ่านข้อมูล/คำสั่ง กับ LCD Controller

ภาพที่ 2.8 จอแสดงผลแบบ LCD ขนาด 16 ตัวอักษร 2 บรรทัด

2.7 สวิทช์แบบเมทริกซ์ [8]

การต่อวงจร สวิทช์แบบเมทริกซ์ โดยวิธีการนี้เหมาะกับระบบที่มีความจำเป็นต้องใช้งานสวิทช์ มากๆ เช่น วงจรแป้นพิมพ์ Keyboard ดังภาพที่ 2.9 ที่ใช้สำหรับป้อนค่าตัวเลข ตัวอักษร และข้อความต่างๆ ซึ่ง จะเกิดความไม่สะดวกสำหรับผู้ใช้งานเป็นอย่างมาก ถ้าออกแบบให้มีจำนวน Switch Keyboard น้อยๆเพราะจะยากแก่การค้นหาตำแหน่งของตัวเลข ตัวอักษรที่อยู่ซ้อนกันอยู่ในคีย์เดียวหลายๆ ชั้น โดยวิธีการนี้จะต้องใช้พอร์ต 2 ส่วน คือ พอร์ตสำหรับอ่านค่าสถานะของ Key Switch จากทางแถว (Row) และพอร์ตสำหรับทำหน้าที่ส่งค่าออกไป Scan key ในแต่ละหลัก (Column) ของวงจร โดยจำนวนของสวิทช์ จะขึ้นอยู่กับขนาดของแถวและหลักที่ใช้ เช่น ถ้าเป็นขนาด 4x3 ก็จะได้ทั้งหมด 12 ตำแหน่ง ดังภาพที่ 2.10



ภาพที่ 2.9 สวิทช์แบบเมทริกช์ขนาด 4x3

ภาพที่ 2.10 การต่อวงจรสวิทช์แบบเมทริกช์

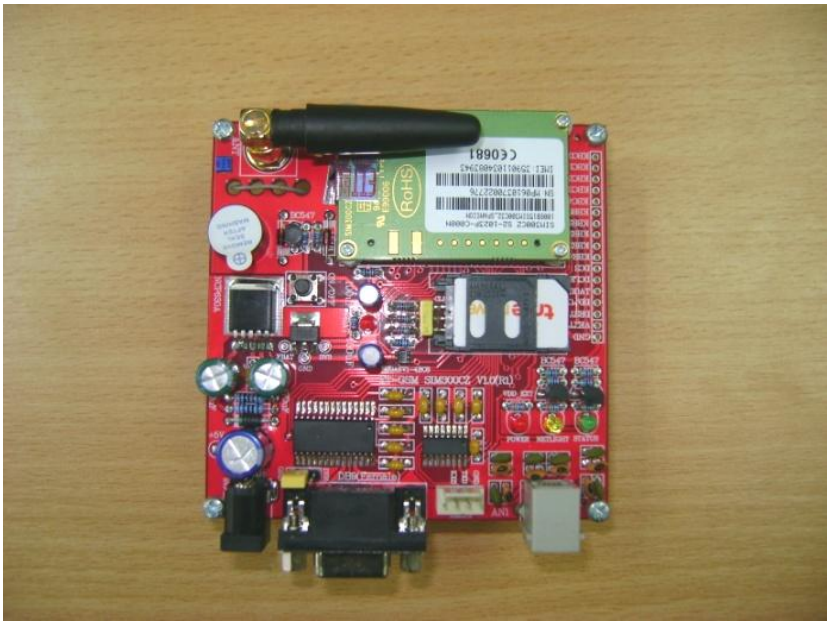
ซึ่งวิธีการ Scan key แบบ Matrix นี้จะทำทีละหลัก (Column) โดยเริ่มจากหลักแรกไปหาหลักสุดท้ายตามลำดับ สำหรับลักษณะของการต่อวงจรโดยทั่วไปของวิธีการนี้จะนิยมคงสถานะของสัญญาณด้านที่เป็น Input ให้มีค่าเป็น “1” รอไว้ก่อนเสมอโดยการต่อตัวต้านทาน Pull-Up เข้ากับ Port- Input รอไว้ก่อน โดยในการ Scan key จะทำทางด้านหลัก (Column) โดยส่งค่าออกไปทางด้าน Port- Input ให้มีค่าเป็น “0” ครั้งละ 1 บิต แล้วก็อ่านค่าจาก Port-Input เข้ามาตรวจสอบว่าทุกบิตยังคงเป็น “1” อยู่หรือไม่ ซึ่งถ้าพบว่ามีบิตใดเป็น “0” (Column Active = “0”) ก็สามารถทราบได้ทันทีว่ามีการกดคีย์ขึ้นที่ตำแหน่ง Row และ Column นั้นๆ แต่ถ้าทุกบิตยังคงมีค่าเป็น “1” ก็ให้เปลี่ยนการ Scan ไปยัง Column ถัดไปอีกโดยทำเหมือนกันกับ Column แรกจนครบทุก Column

2.8 GSM โมดูล [9]

ET-GSM SIM300CZ ดังภาพที่ 2.11 เป็นชุดเรียนรู้และพัฒนาระบบการสื่อสารไร้สายโดยใช้โมดูล GSM/GPRS รุ่น SIM300CZ ของ “ SIMCom Ltd. ” เป็นอุปกรณ์หลักซึ่ง SIM300CZ เป็นโมดูลสื่อสารระบบ GSM/GPRS ขนาดเล็กรองรับระบบสื่อสาร GSM ความถี่900/1800/1900MHz โดยสั่งงานผ่านทางพอร์ตสื่อสารอนุกรม RS232 ด้วยชุดคำสั่ง AT Command สามารถประยุกต์ใช้งานได้มากมายหลายรูปแบบไม่ว่าจะเป็นการ รับส่งสัญญาณแบบ Voice, SMS, Data, FAX และยังรวมถึงการสื่อสารด้วย Protocol TCP/IP ด้วย

1. คุณสมบัติของบอร์ด 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 +5V ขึ้นไปสามารถจ่ายกระแสให้กับโมดูล SIM300CZ และอุปกรณ์เชื่อมต่อต่างๆได้อย่างเพียงพอ
  - o มีวงจร Regulate ขนาด 4.2V / 3A สำหรับจ่ายให้กับโมดูล SIM300CZ ได?อย่างเพียงพอ สามารถใช้กับ SIM ของระบบ GSM900MHz แบบ 2-Watt ได้อย่างไม่เกิดปัญหา
  - o มีวงจร Regulate ขนาด 3.3V / 1A สำหรับจ่ายให้กับวงจรเชื่อมต่อภายนอกโดยไม่ต้องไปดึงไฟจากตัวโมดูลมาใช้ ป้องกันปัญหาโมดูลเสียหายจากวงจรภายนอกดึงกระแสเกินพิกัด และสะดวกต่อการออกแบบวงจรเชื่อมต่อเพิ่มเติมโดยไม่ต้องกังวลว่ากระแสจะไม่พอจ่ายให้กับอุปกรณ์
- มีวงจร Line Driver สำหรับแปลงระดับสัญญาณโลจิกจากโมดูล SIM300CZ ให้เป็น RS232 ระดับ มาตรฐานครบทุกเส้นสัญญาณทั้งพอร์ตที่ใช้ในการสื่อสารสำหรับสั่งงานโมดูลและพอร์ตสำหรับใช้ในการพัฒนาโปรแกรม (Debug) สามารถเชื่อมต่อกับพอร์ต RS232 มาตรฐานได้ทันที
- มี LED แสดงสถานะพร้อมในบอร์ดสำหรับแสดงสถานะของแหล่งจ่ายไฟสถานะพร้อมทำงานของโมดูล สถานะในการเชื่อมต่อกับ Network และสถานะ Power-On/Power-OFF ของโมดูล
- มีขั้วสำหรับเชื่อมต่อกับ Handset (ชุดปากพูดและหูฟังของโทรศัพท์บ้าน) โดยใช้ขั้วต่อแบบ RJ11 มาตรฐานพร้อมวงจร Voice Filter สามารถนำชุด Handset ของโทรศัพท์บ้านต่อเข้ากับบอร์ดทางขั้วต่อแบบ RJ11 สำหรับใช้พูดคุยโทรออกและรับสายได้โดยสะดวก
- มี Buzzer พร้อมวงจรขับเพื่อสร้างสัญญาณเสียงในกรณีมีการโทรเรียกเข้ามายังโมดูล
- มีจุดยึดเสาอากาศสำหรับใช้เป็นจุดพักสำหรับเชื่อมต่อกับเสาอากาศแบบต่างๆได้โดยสะดวก
- มีขั้วต่อสำหรับติดตั้งโมดูล SIM300CZ พร้อมเสารองและสกรูยึดโมดูลกับตัวบอร์ด
- มีจุดต่อสัญญาณอื่นๆที่เหลือจากโมดูลเช่น Keyboard, Display ,GPIO ,Battery Charger ฯลฯ สำหรับให้ผู้ใช้ต่อขยายไปยังวงจรที่ออกแบบเพิ่มเติมได้โดยง่ายและสะดวก



ภาพที่ 2.11 โมดูล GSM Sim300CZ