การเขียนโปรแกรมบน IPST-MicroBOX

นายยอดชาย ขุนสังวาลย์
โรงเรียนสงวนหญิง

เนื้อหา

- แผงวงจรควบคุมหลัก
- เครื่องมือพัฒนาโปรแกรมบน MicroBOX
- การเชื่อมต่อกับพีซี
- การดาวน์โหลดโปรแกรม
- แผงวงจรอินพุท/เอาพุทและการใช้งาน

การเตรียมตัว

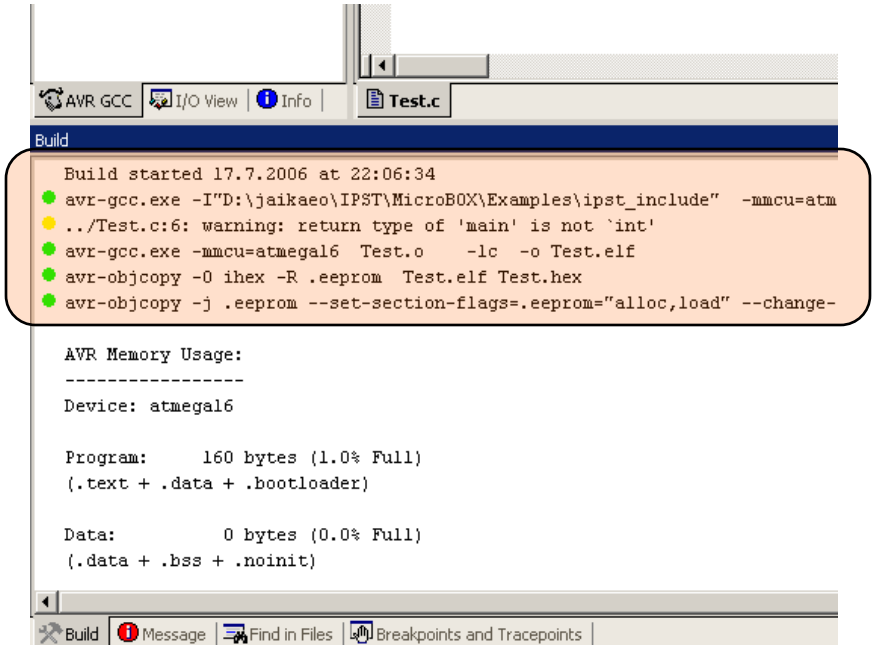
- ติดตั้งโปรแกรม AVR Studio 4
- ติดตั้งโปรแกรม WinAVR
- ติดตั้งโปรแกรม PonyProg2000

โปรแกรม AVR Studio

- เป็นเครื่องมือหลักในการพัฒนาโปรแกรมบนไมโครคอลโทรเลอร์ตระกูล AVR
- คุณสมบัติหลัก ๆ
  - เครื่องมือสำหรับจัดการโครงงาน
  - เอดิเตอร์เพื่อแก้ไขซอร์สไฟล์
  - เครื่องมือจำลองการทำงานของไมโครคอลโทรเลอร์

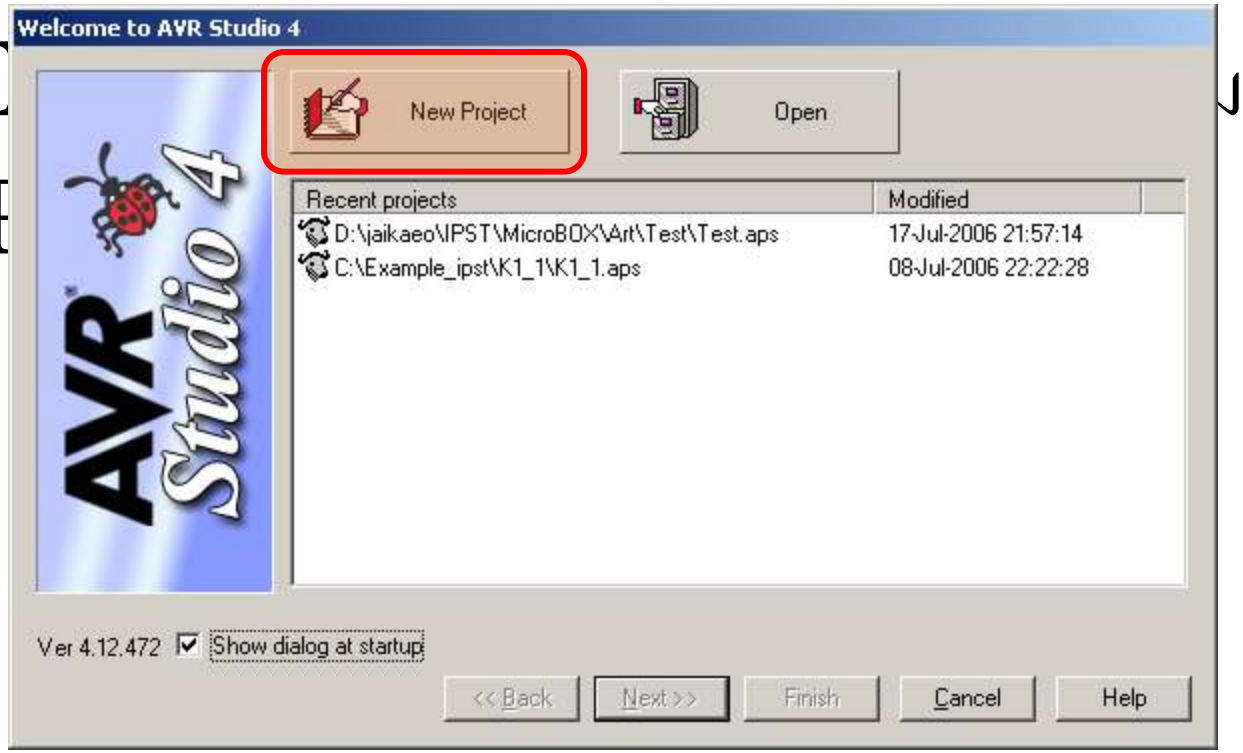
โปรแกรม WinAVR

- คอมไพเลอร์ภาษา C เพื่อแปลโปรแกรมที่เขียนสำหรับสถาปัตยกรรม AVR
  - ถูกเรียกใช้งานผ่านโปรแกรม AVR Studio



ทดลองสร้างโปรเจ็คใหม่

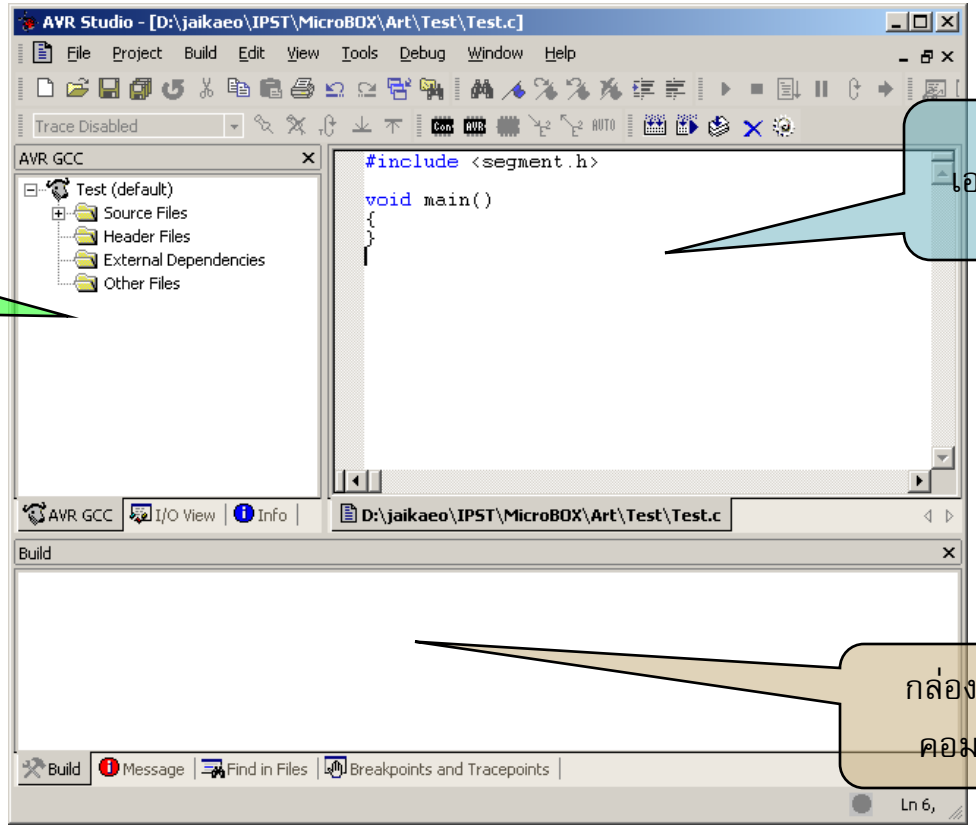
- เลือก J ง AVR



กล่องสร้างโปรเจค

- Project type: → AVR GCC
- Project name: → Counting
- เลือก Location ตามความเหมาะสม

หน้าจอหลักของ AVR Studio


AVR Studio - [D:\jaikaeo\IPST\MicroBOX\Art\Test\Test.c]
File Project Build Edit View Tools Debug Window Help
Trace Disabled
AVR GCC
Test (default)
Source Files
Header Files
External Dependencies
Other Files
#include <segment.h>
void main()
{
}
AVR GCC
I/O View
Info
D:\jaikaeo\IPST\MicroBOX\Art\Test\Test.c
Build
Build
Message
Find in Files
Breakpoints and Tracepoints
Ln 6,

รายชื่อไฟล์ที่เกี่ยวข้อง

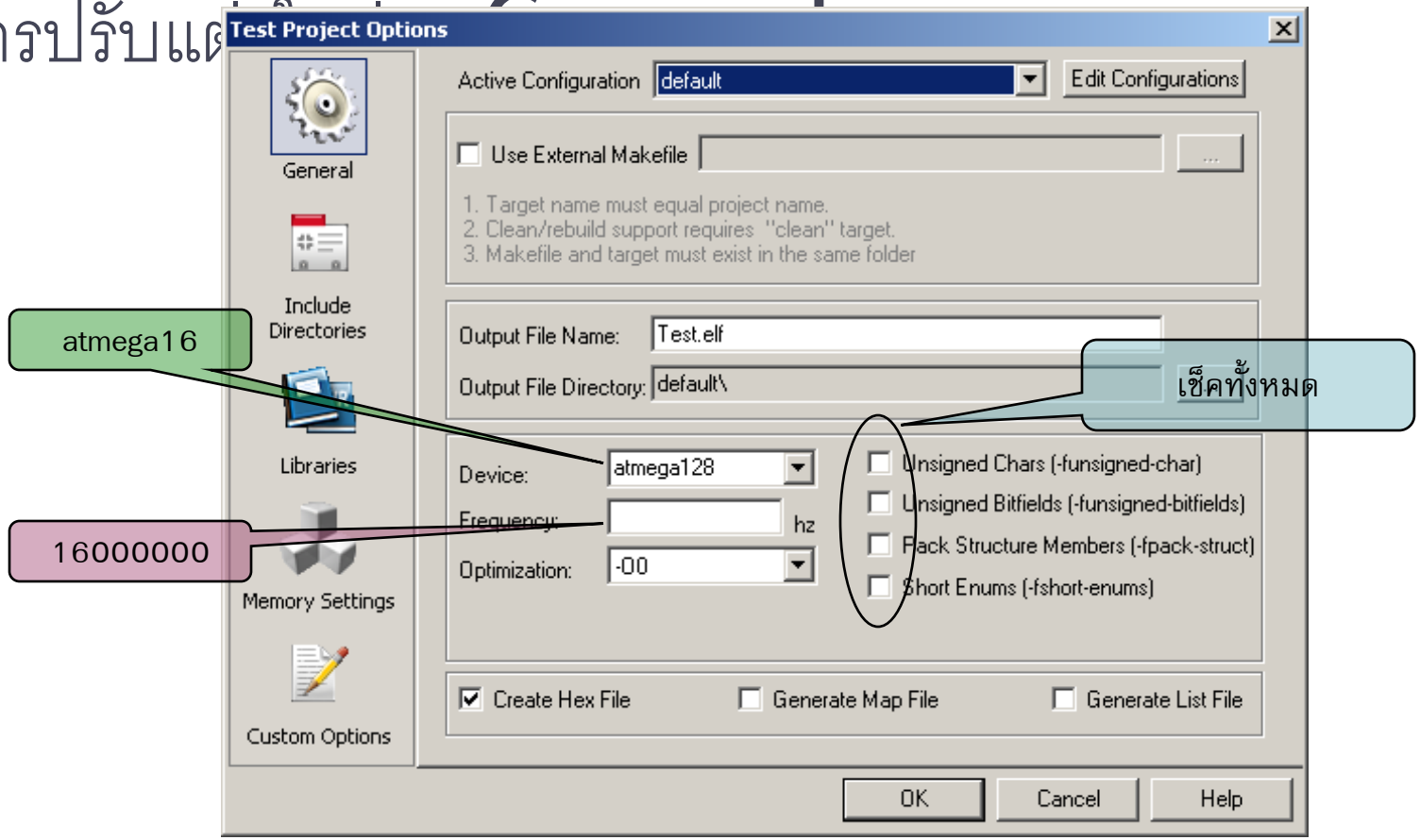
เอดิเตอร์ภาษา C

กล่องรายงานผลการคอมไพล์โปรแกรม

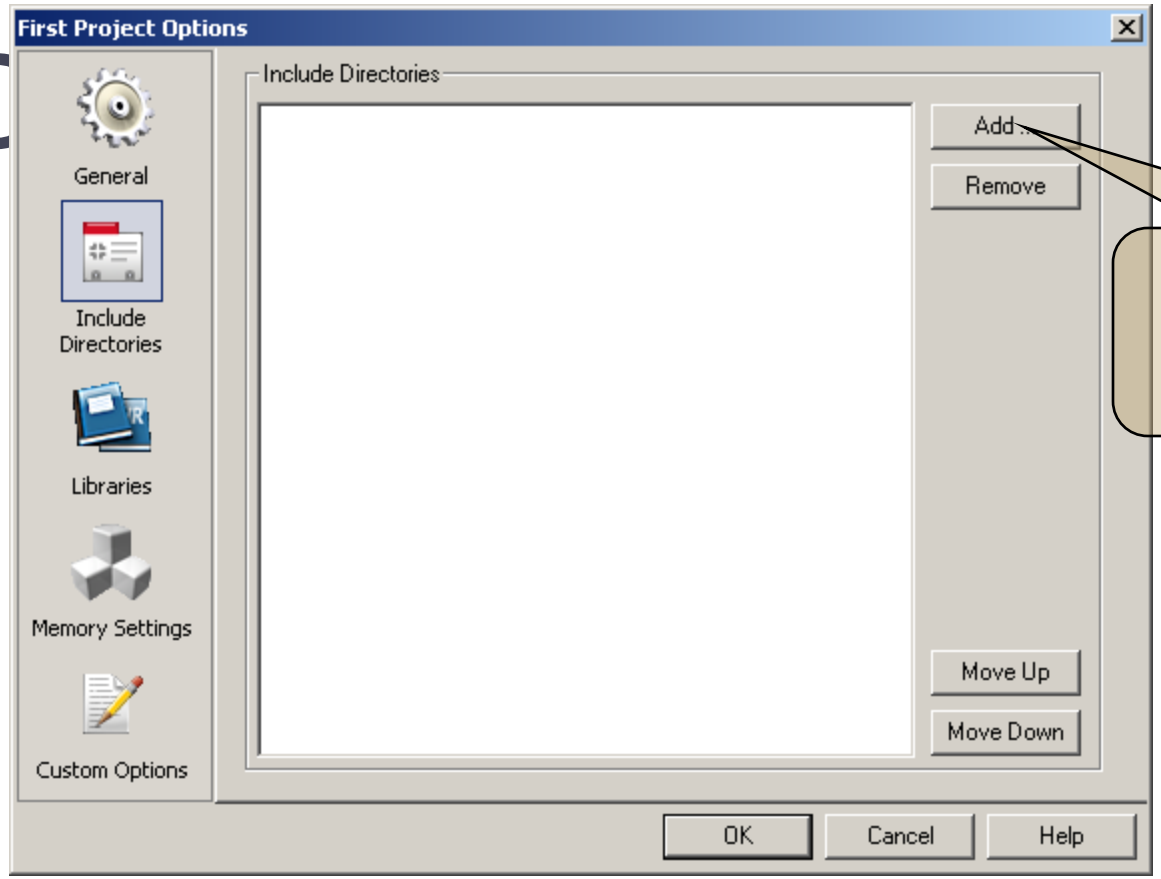
การปรับแต่งค่าในโปรเจ็ค

- โปรเจ็คที่สร้างขึ้นใหม่ต้องผ่านการปรับค่าต่าง ๆ ก่อนเริ่มพัฒนาโปรแกรม
- เลือกเมนู Project → Configuration Options

การปรับแต่


Test Project Options
General
Include Directories
Libraries
Memory Settings
Custom Options
Active Configuration
default
Edit Configurations
Use External Makefile
1. Target name must equal project name.
2. Clean/rebuild support requires "clean" target.
3. Makefile and target must exist in the same folder
Output File Name:
Test.elf
Output File Directory:
default\
Device:
atmega128
Frequency:
hz
Optimization:
-O0
Unsigned Chars (-funsigned-char)
Unsigned Bitfields (-funsigned-bitfields)
Pack Structure Members (-fpack-struct)
Short Enums (-fshort-enums)
Create Hex File
Generate Map File
Generate List File
OK
Cancel
Help
atmega16
16000000
เช็คทั้งหมด

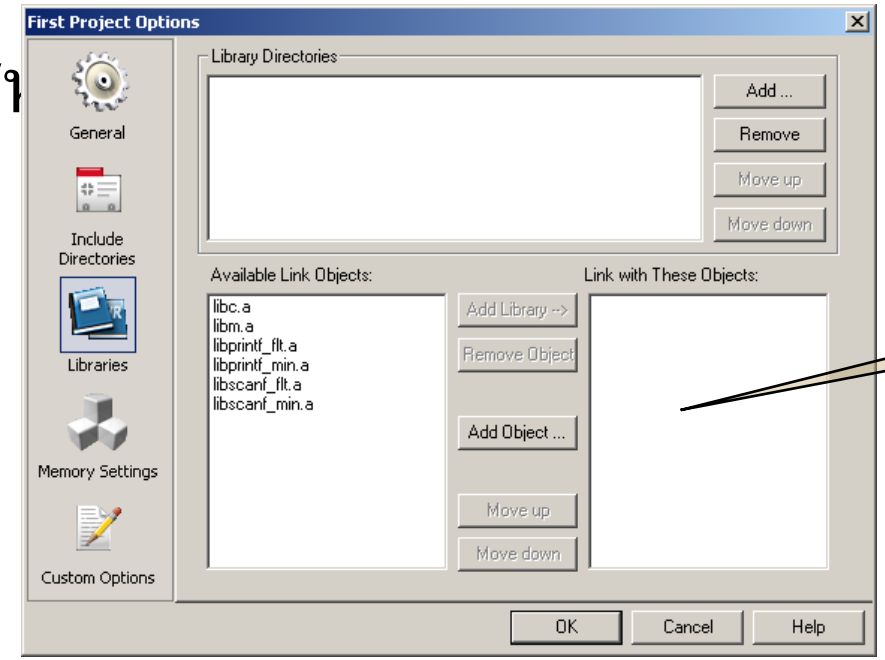
การปรับแต่งในส่วน Include



กด Add เพื่อเพิ่มไดเรคตอรี่ ipst_include ลงไป

การปรับแต่งในส่วน Libraries

- ปรับแต่งเฉพาะเมื่อต้องการใช้งานไฟล์เฮดเดอร์ lcd.h และ/ห



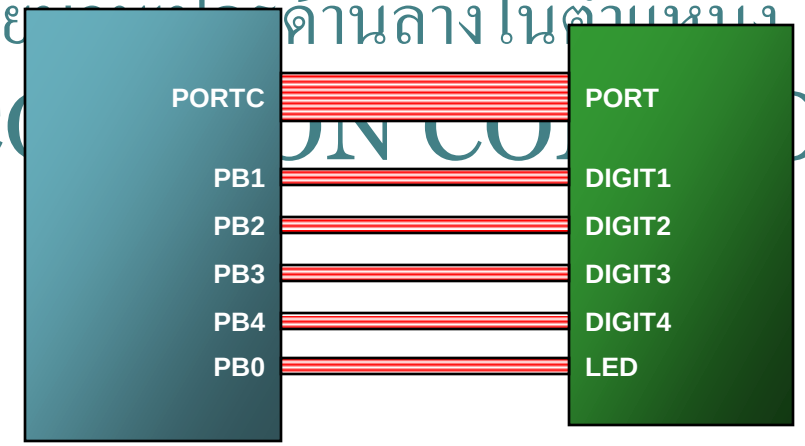
เพิ่มไลบรารี libm.a

ชุดคำสั่งในไลบรารี IPST

- **in_out.h** – รวมคำสั่งพื้นฐานเพื่อจัดการอินพุทเอาท์พุท
- **sleep.h** – รวมคำสั่งหน่วงเวลาเป็นมิลลิวินาที
- **led.h** – รวมคำสั่งแสดงผลข้อมูลบนแถว LED 8 ดวง
- **segment.h** – รวมคำสั่งแสดงผลข้อมูลบน LED แบบ 7 ส่วน
- **timer.h** – รวมคำสั่งจับเวลา
- **analog.h** - รวมคำสั่งอ่านค่าอินพุทแบบแอนะล็อก
- **sound.h** – รวมคำสั่งส่งสัญญาณเสียงออกทางลำโพงเปียโซ
- **motor.h** – รวมคำสั่งควบคุมมอเตอร์ไฟตรง

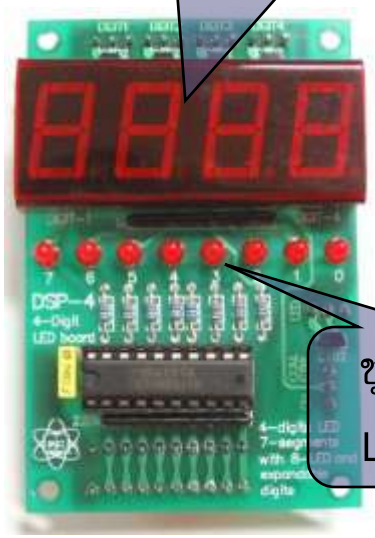
แผงวงจรแสดงผลตัวเลข 4 หลัก

- เป็นอุปกรณ์หลักในการแสดงผล
- การเชื่อมต่อทางฮาร์ดแวร์
  - เสียบ[illegible]ด้านล่างในตำแหน่ง



แผงวงจรหลัก

แผงวงจรแสดงผล

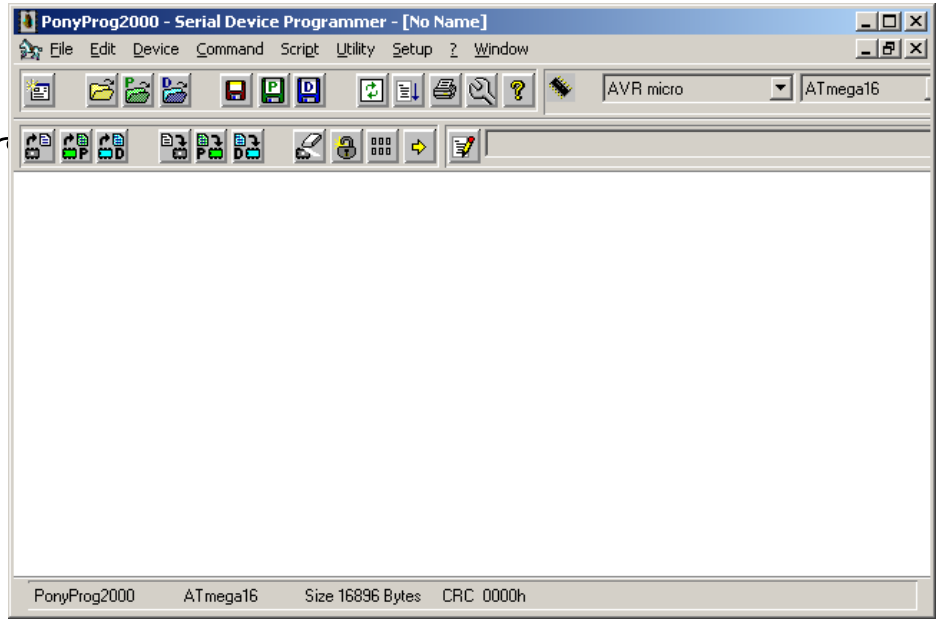


ตัวอย่าง โปรแกรมนับเลข

```
#include <avr/io.h>
#include <in_out.h>
#include <segment.h>
#include <sleep.h>

void main()
{
  uint16_t i = 0;
  while (1)
  {
    sleep(1000);
    segment(i++);
    if (i > 10) i = 0;
  }
}
```

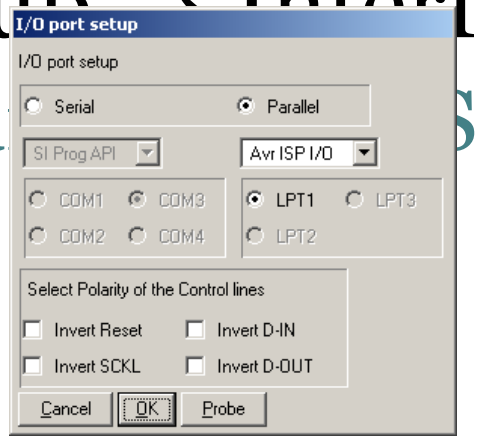
โปรแกรม PonyProg2000

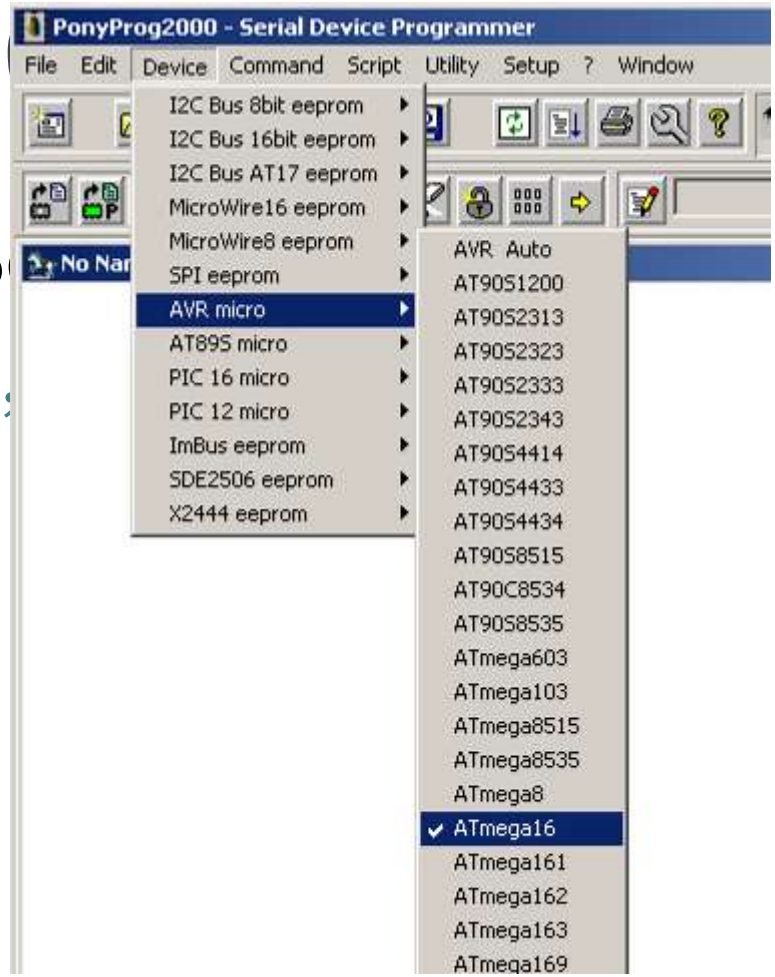
- ใช้สำหรับส่งโปรแกรมที่อยู่ในรหัสภาษาเครื่องไปยังไมโครคอนโทรลเลอร์
- ติดต่อผ่านพอร์ตแบบขนาน
- โปรแกรมจะถูกเขียนลงบนหน่วยความจำแบบแฟลชที่อยู่ภายในไมโครคอนโทรลเลอร์



ปรับแต่งค่าใน PonyProg2

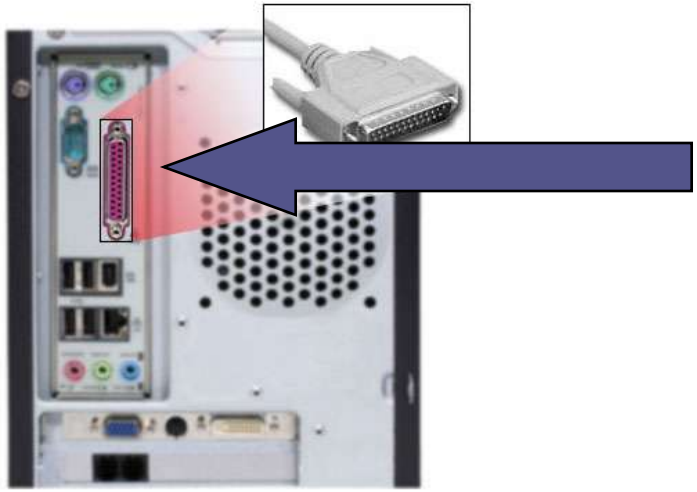
- Setup → Interface S
  - Pa SP I/O,



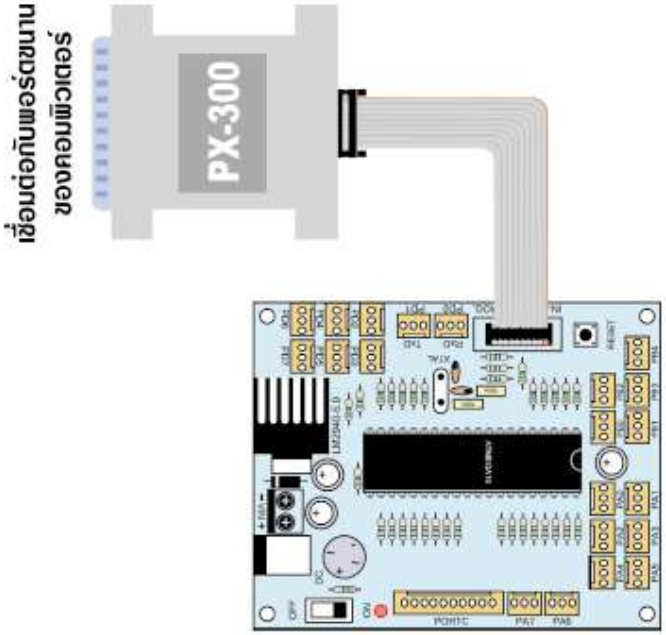


- Device → AVR micro
  → ATmega16

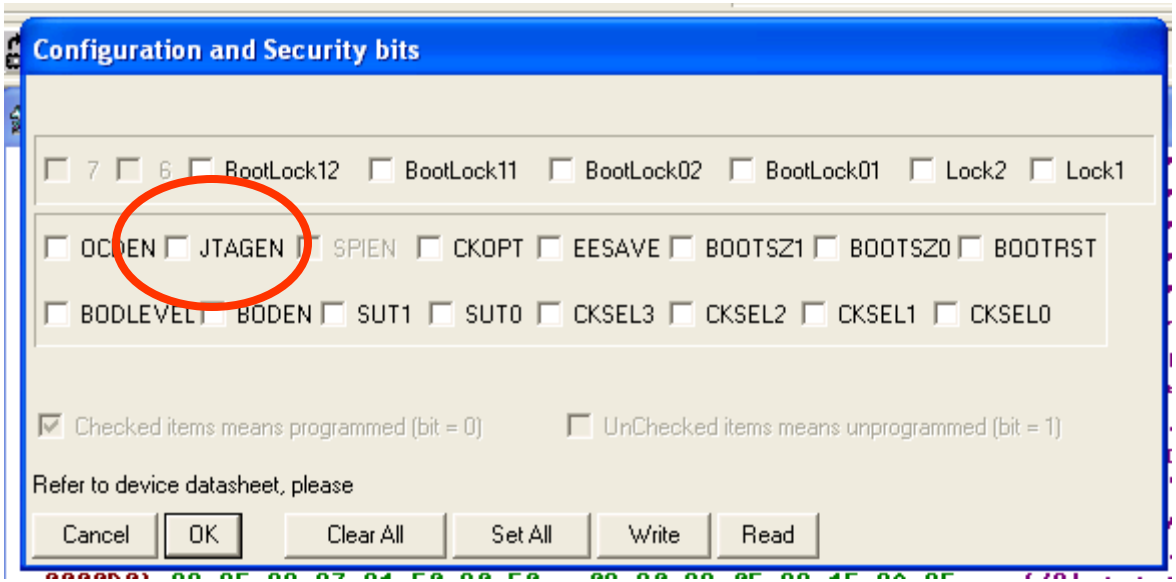
การเชื่อมต่อกับพีซี



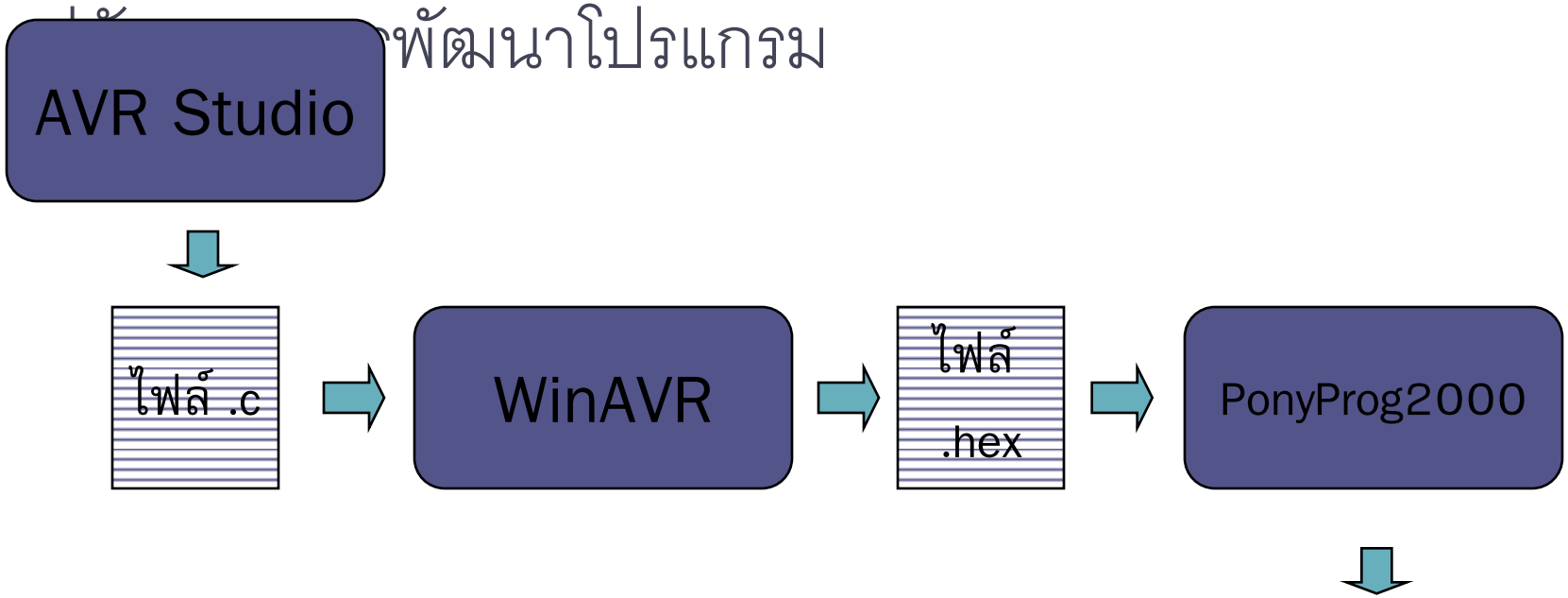
ด้านหลังของเครื่องคอมพิวเตอร์

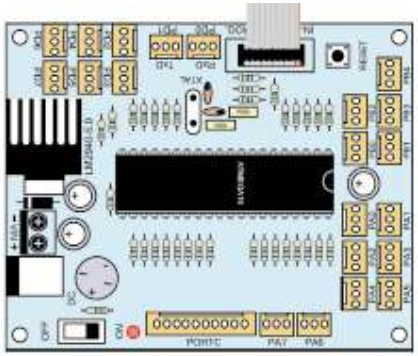


เคลียร์บิต JTAGEN



สพัฒนาโปรแกรม





เฉลย โปรแกรมนับถอยหลัง

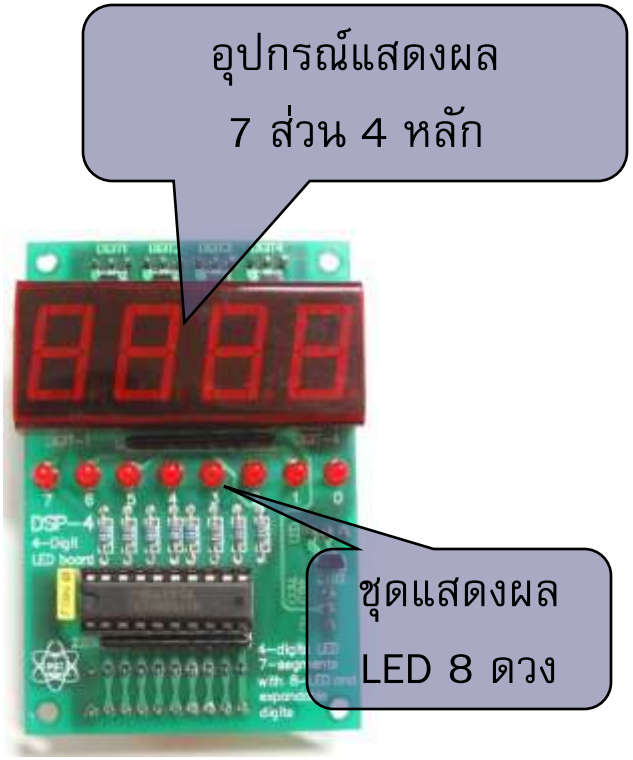
```
#include <avr/io.h>
#include <in_out.h>
#include <segment.h>
#include <sleep.h>

void main()
{
  int16_t i = 20;
  while (1)
  {
    sleep(1000);
    segment(i--);
    if (i < 0) i = 20;
  }
}
```

ให้แน่ใจว่าไม่ใช่ตัวแปรแบบ unsigned

อุปกรณ์แสดงผลพื้นฐาน

- แผงวงจรแสดงผล 4 หลัก

- ไลบรารี่ segment.h



ชุดคำสั่งสำหรับวงจรแสดงผล

- ไฟล์เฮดเดอร์ที่ใช้
  - **avr/io.h, in_out.h, segment.h**
- คำสั่งในไลบรารี
  - **segment(*x*)** – นำค่า ***x*** มาแสดงผลเป็นตัวเลข 4 หลัก
  - **segment_dot(*d*)** – กำหนดจุดในตำแหน่ง ***d*** ได้แก่ 1 (ขวา), 2,3,4 (ซ้าย)
  - **segment_off()** – ปิดการแสดงผลข้อมูลของ LED 7 ส่วน
  - **led8(*d*)** – แสดงค่า ***d*** บนแถว LED ในรูปเลขฐานสอง
  - **led8_off()** – กำหนดให้แถว LED อยู่ในสถานะดับ

sleep - คำสั่งหน่วงเวลา

- อยู่ในไฟล์เฮดเดอร์ **sleep.h**
- การใช้งาน
  - **sleep(m)** – หน่วงเวลา m มิลลิวินาที (โดยประมาณ)

เครื่องตั้งเวลาหนึ่งนาที

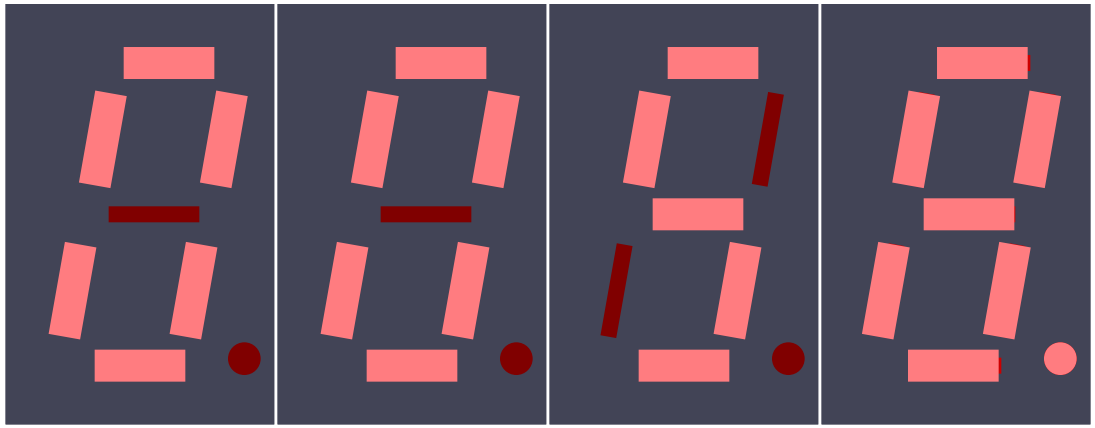
- สร้างโปรเจ็คใหม่: **Timer**
- ให้ MicroBOX ทำงานเป็นนาฬิกาจับเวลา 1 นาที
  - แสดงเลข 60 เมื่อเปิดเครื่อง นับถอยหลังทุกวินาที
  - นับถึงศูนย์ให้หน้าจอดับไป

เฉลย Timer

```
#include <avr/io.h>
#include <in_out.h>
#include <segment.h>
#include <sleep.h>

void main()
{
  int16_t i = 60;
  while (i >= 0)
  {
    segment(i--);
    sleep(1000);
  }
  segment_off();
  while (1) { }
}
```

เพิ่มสีสันให้ไทเมอร์

- ทำให้จุดซ้ายมือกระพริบทุก ๆ ครึ่งวินาทีเมื่อเวลายังเดินอยู่



เฉลย Timer

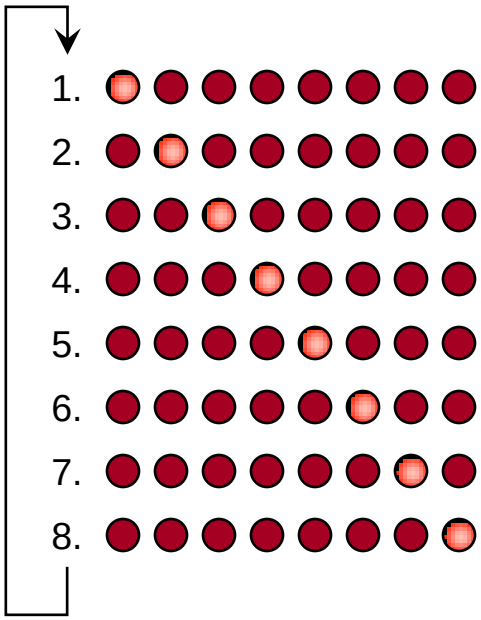
```
#include <avr/io.h>
#include <in_out.h>
#include <segment.h>
#include <sleep.h>

void blink()
{
  while (1)
  {
    segment(0);
    sleep(500);
    segment_off();
    sleep(500);
  }
}
```

```
void main()
{
  int16_t i = 60;
  while (i >= 0)
  {
    segment(i--);
    segment_dot(1);
    sleep(500);
    segment_dot(0);
    sleep(500);
  }

  blink();
}
```

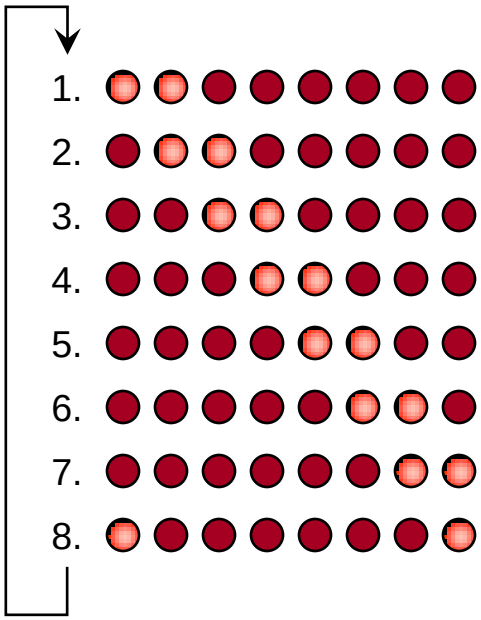
ตัวอย่าง – ใช้งานแอด LED

```
#include <avr/io.h>
#include <in_out.h>
#include <segment.h>
#include <sleep.h>

void main()
{
  uint8_t i = 0;
  while (1)
  {
    led8(i);
    i++;
    sleep(100);
  }
}
```

- เขียนโปรแกรมไฟวิ่งในรูปแบบต่างๆ



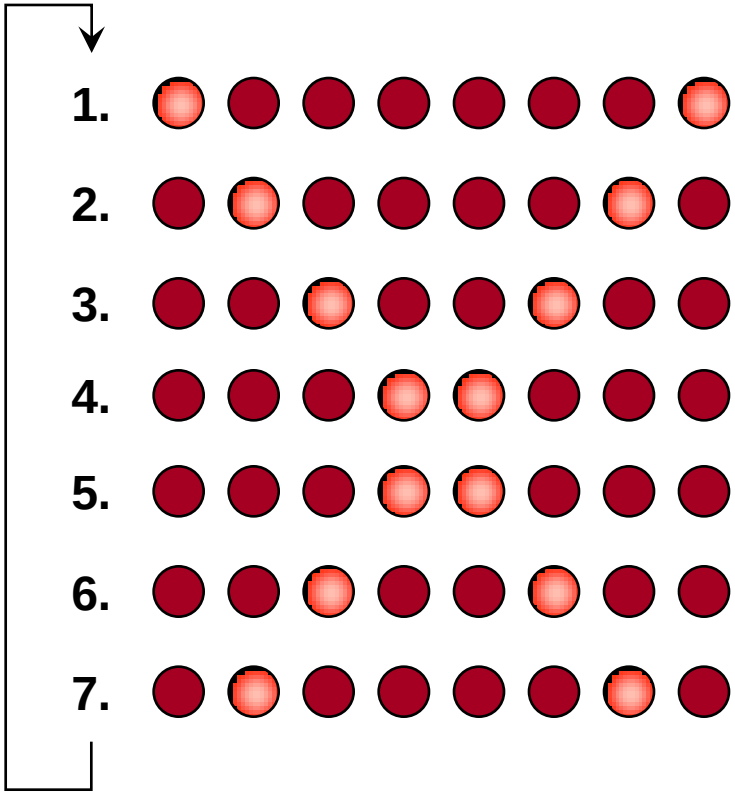
แบบที่ 1



แบบที่ 2

ไฟวิ่งหลายรูปแบบ

- แบบที่สาม



เฉลย

```
#include <avr/io.h>
#include <in_out.h>
#include <segment.h>
#include <sleep.h>

void main()
{
  uint8_t i = 0b10000000;
  while (1)
  {
    led8(i);
    i = i >> 1;
    if (i == 0) i = 0b10000000;
    sleep(100);
  }
}
```

เฉลย

```
#include <avr/io.h>
#include <in_out.h>
#include <segment.h>
#include <sleep.h>

void main()
{
  uint8_t i = 0b11000000;
  while (1)
  {
    led8(i);
    i = i >> 1;
    if (i == 0b00000001) i = 0b10000001;
    if (i == 0b01000000) i = 0b11000000;
    sleep(100);
  }
}
```

เฉลย

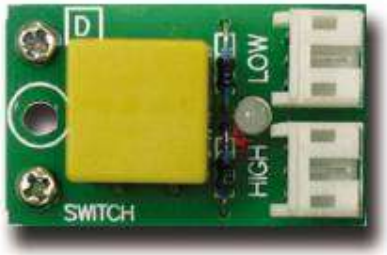
```
#include <avr/io.h>
#include <in_out.h>
#include <segment.h>
#include <sleep.h>

void main()
{
  uint8_t i = 0b10000000, j = 0b01000000;
  while (1)
  {
    led8(i | j);
    i = i >> 1; j = j >> 1;
    if (i == 0) i = 0b10000000;
    if (j == 0) j = 0b10000000;
    sleep(100);
  }
}
```

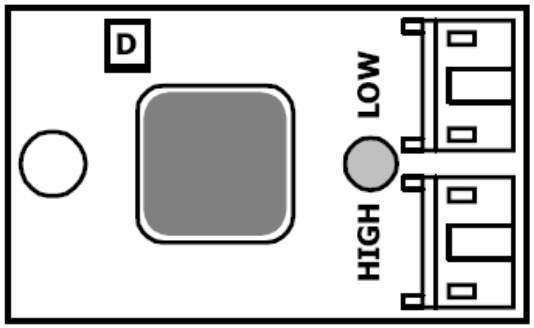
เฉลย

```
#include <avr/io.h>
#include <in_out.h>
#include <segment.h>
#include <sleep.h>

void main()
{
  uint8_t i = 0b10000000, j = 0b00000001;
  while (1)
  {
    led8(i | j);
    i = i >> 1; j = j << 1;
    if (i == 0) i = 0b10000000;
    if (j == 0) j = 0b00000001;
    sleep(100);
  }
}
```

สวิตช์กดติดปล่อยดับ

- เป็นอุปกรณ์อินพุทแบบดิจิตอล
- เชื่อมต่อกับพอร์ตได้ทุกพอร์ต ยกเว้นพอร์ต C





เมื่อกดเป็นลอจิก 0
และแสดงไฟเป็นสีเขียว

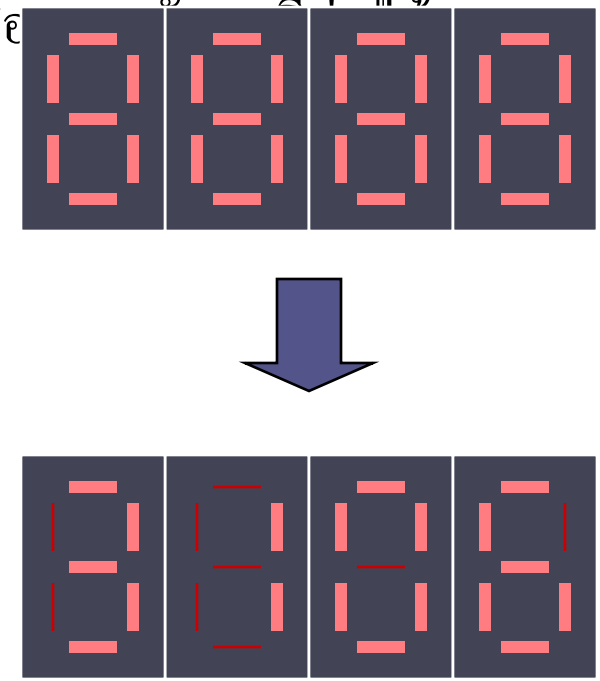
เมื่อกดเป็นลอจิก 1
และแสดงไฟเป็นสีแดง

คำสั่งอินพุทเอาท์พุทแบบดิจิตอล

- ไฟล์เฮดเดอร์ที่ใช้
  - **avr/io.h, in_out.h**
- คำสั่งในไลบรารี
  - **in_a(*p*)** – อ่านค่าดิจิตอลอินพุทจากพอร์ต A ขา *p*
  - **in_b(*p*)** – อ่านค่าดิจิตอลอินพุทจากพอร์ต B ขา *p*
  - **in_c(*p*)** – อ่านค่าดิจิตอลอินพุทจากพอร์ต C ขา *p*
  - **in_d(*p*)** – อ่านค่าดิจิตอลอินพุทจากพอร์ต D ขา *p*
  - **out_a(*p*,*d*)** – ส่งค่า *d* (0 หรือ 1) ไปยังขา *p* ของพอร์ต A
  - **out_b(*p*,*d*)** – ส่งค่า *d* (0 หรือ 1) ไปยังขา *p* ของพอร์ต B

แบบฝึกหัด - *เครื่องออกหวย*

- กดสวิตช์ค้างเพื่อทำการวนสุ่มตัวเลข
- ปล่อยสวิตช์เพื่อ



กดสวิตช์ค้าง

ปล่อย

เฉลย - เครื่องออกหวย

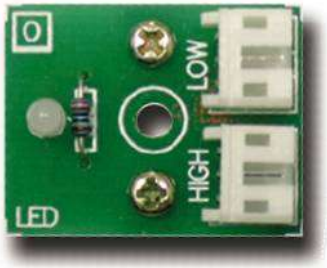
```
#include <avr/io.h>
#include <in_out.h>
#include <segment.h>

void main()
{
  uint16_t i = 0;
  while (1)
  {
    if (in_d(7) == 1)
    {
      i = random() % 10000;
    }
    segment(i);
  }
}
```

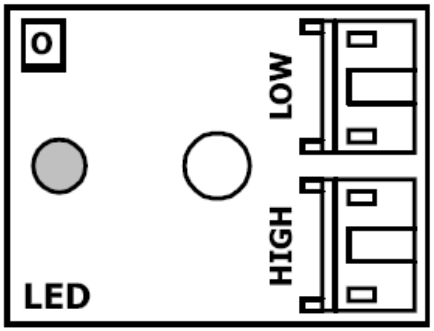
หรือ

```
#include <avr/io.h>
#include <in_out.h>
#include <segment.h>

void main()
{
  uint16_t i = 0;
  while (1)
  {
    if (in_d(7) == 1)
    {
      i++;
      if (i > 9999) i = 0;
    }
    segment(i);
  }
}
```

แผงวงจรแสดงผล LED สองสี

- เป็นอุปกรณ์เอาท์พุทแบบดิจิตอล
- เชื่อมได้กับทุกพอร์ต ยกเว้นพอร์ต C





ถ้าเป็นสัญญาณลอจิก "0"
จะแสดงผลเป็นสีเขียว

ถ้าเป็นสัญญาณลอจิก "1"
จะแสดงผลเป็นสีแดง

เปิดปิด LED ด้วยสวิตช์

- ทำใบงานที่ 4.2
- กดสวิตช์แล้วปล่อยหนึ่งครั้ง ให้ไฟ (LED) ติด
- กดแล้วปล่อยอีกครั้ง ให้ไฟดับ

เฉลย

```
#include <avr/io.h>
#include <in_out.h>
#include <segment.h>

void main()
{
  out_d(6, 0);
  while (1)
  {
    while (in_d(7) == 0) {}
    toggle_d(6);
    while (in_d(7) == 1) {}
  }
}
```

LED ต่ออยู่กับ D6

สวิตช์ต่อกับ D7

ตัวอย่าง - ควบคุมไฟด้วยสวิตช์

- เชื่อมวงจร LED สองสีเข้ากับ PD6
- เชื่อมวงจรสวิตช์ช่อง HIGH เข้ากับ PD7

```
#include <avr/io.h>
#include <in_out.h>

void main()
{
  while (1)
  {
    if (in_d(7) == 1)
      out_d(6,1);
    else
      out_d(6,0);
  }
}
```

หรือ

```
#include <avr/io.h>
#include <in_out.h>

void main()
{
  while (1)
  {
    out_d(6,in_d(7));
  }
}
```

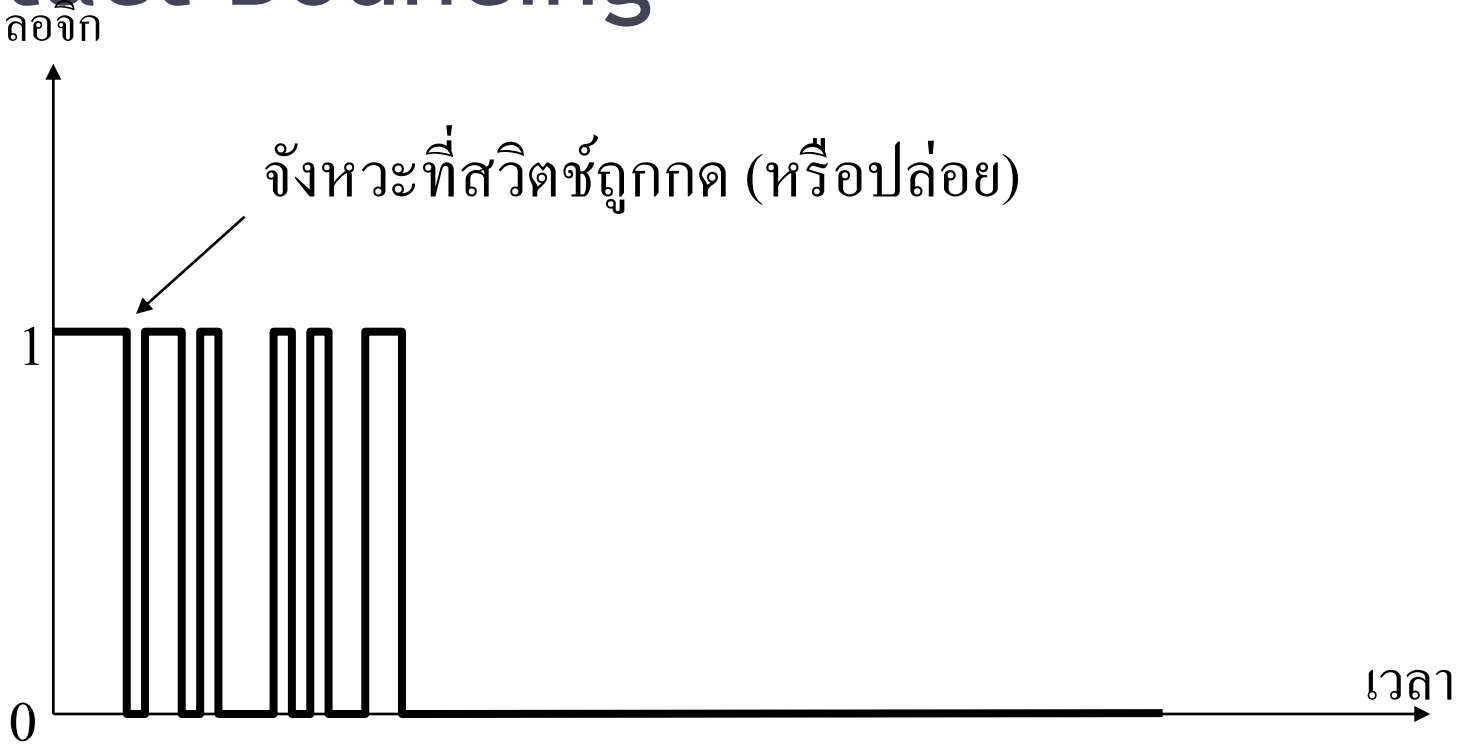
- ข้อควรระวัง
  - อย่าใช้หัวต่อบนแผงวงจร LED สองสีพร้อมกันทั้งคู่

ตัวอย่าง - ตัวนับจำนวนครั้งการกดสวิตช์และแสดงผลออกทาง LED 7 ส่วน

```
#include <avr/io.h>
#include <in_out.h>
#include <segment.h>
#include <sleep.h>
void main()
{
  uint16_t i = 0;
  while (1)
  {
    while (in_d(7) == 0){} sleep(10);
    while (in_d(7) == 1){} sleep(10);
    i++; segment(i);
  }
}
```

- ทดลองใช้งานและสังเกตว่าตัวเลขปรากฏตามจำนวนครั้งที่กดจริงหรือไม่

Contact Bouncing



การสร้างคำสั่งตรวจสอบสวิตช์

- สร้างเป็นมาโคร
  - เขียนไว้ก

```
#define wait_d(x) \
  while (in_d(x) == 0){} sleep(10); \
  while (in_d(x) == 1){} sleep(10);
```

- สร้างเป็นฟังก์ชัน

```
void wait_d(uint8_t x)
{
  while (in_d(x) == 0){} sleep(10);
  while (in_d(x) == 1){} sleep(10);
}
```

วงจรลำโพงเปียโซ

- ใช้งานร่วมกับชุดคำสั่งในไลบรารี sound.h

- ไฟล์เฮดเดอร์ที่ใช้
  - **avr/io.h, in_out.h, sound.h**
- คำสั่งในไลบรารี
  - **beep_a(*p*)** – กำเนิดเสียงความถี่ 2000Hz นาน 10 ms ที่ขา p ของพอร์ต A (ใช้กับพอร์ต B, C และ D ได้เช่นกันโดยใช้ **beep_b**, ฯลฯ)
  - **sound_a(*p*, *f*, *t*)** – กำเนิดเสียงความถี่ ***f*** Hz นาน ***t*** มิลลิวินาทีที่ขา ***p*** ของพอร์ต A (ใช้กับพอร์ตอื่นได้เช่นกันโดยใช้ **sound_b**, ฯลฯ)

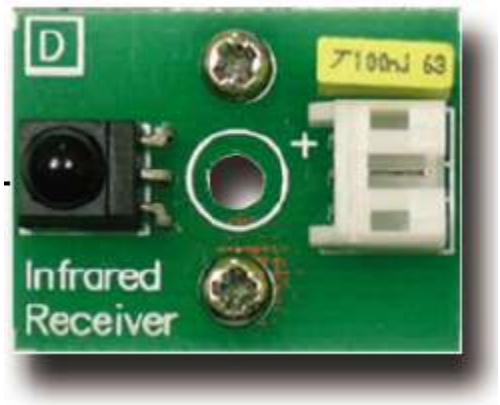
ตัวอย่าง - ประตูเซ่น

```
#include <avr/io.h>
#include <in_out.h>
#include <sleep.h>
#include <sound.h>

void main()
{
  while (1)
  {
    if (in_d(7) == 1)
    {
      sound_d(5, 659, 500);
      sound_d(5, 523, 500);
    }
  }
}
```

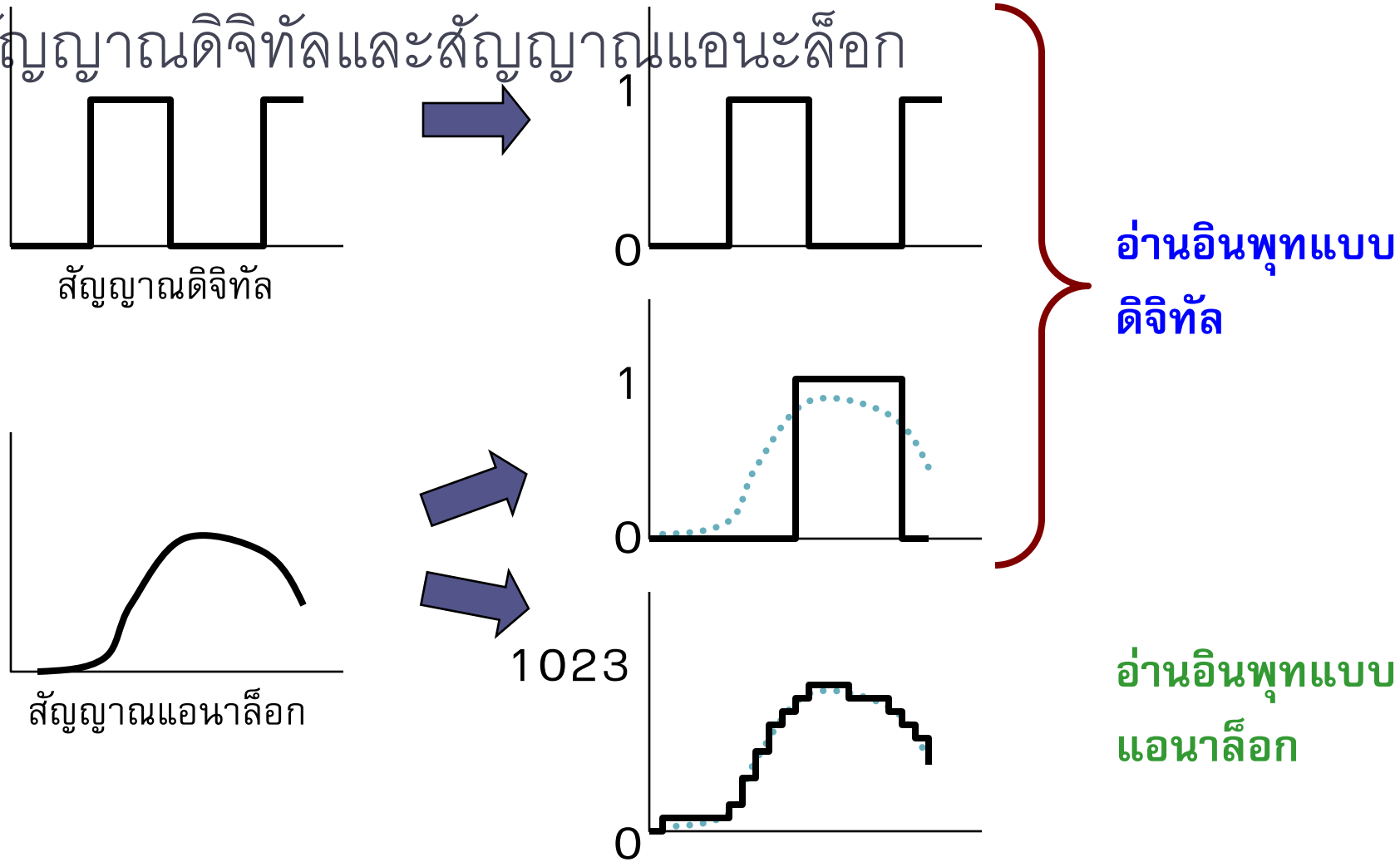
ลำโพงต่อเข้าที่พอร์ต PD5

ตัวรับแสงอินฟราเรด 38kHz

- เป็นอุปกรณ์รับข้อมูลดิจิทัล
- ให้โลจิก ‘0’ ถ้าตรวจจับคลื่นอินฟราเรดย่านความถี่ 38kHz ได้



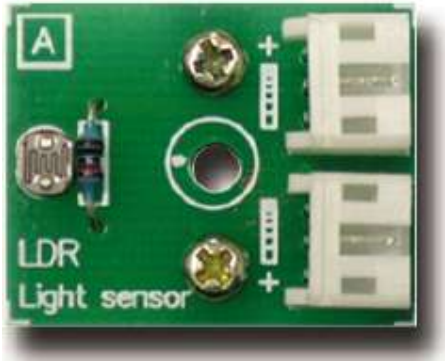
สัญญาณดิจิทัลและสัญญาณแอนะล็อก

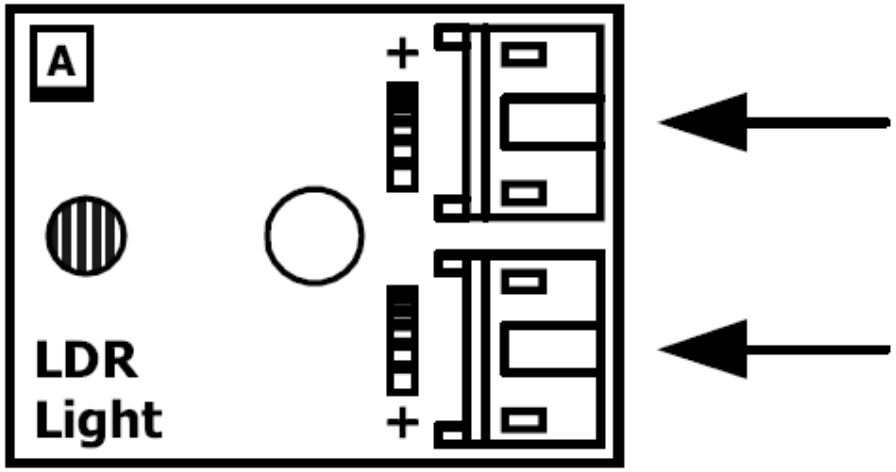


การใช้งานอุปกรณ์แบบแอนะล็อก

- Mega16 มีวงจรแปลงสัญญาณอะนาล็อกเป็นดิจิตอลภายในตัว
  - ใช้งานผ่านพอร์ต PA0 – PA7
  - ความละเอียด 10 บิต (0-1023)
- อุปกรณ์แอนะล็อกที่มีให้
  - วงจรตรวจจับแสงสว่าง, แสงอินฟราเรด
  - วงจรตรวจจับเสียง
  - วงจรตรวจจับแสงสะท้อน
  - วงจรตัวต้านทานปรับค่าได้

วงจรตรวจจับแสงสว่าง



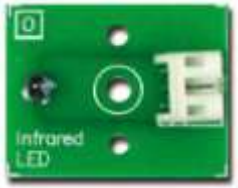


เมื่อแสงมากขึ้น
แรงดันที่ได้จะลดลง

เมื่อแสงมากขึ้น
แรงดันที่ได้จะมากขึ้น

ชุดตรวจวัดแสงอินฟราเรด

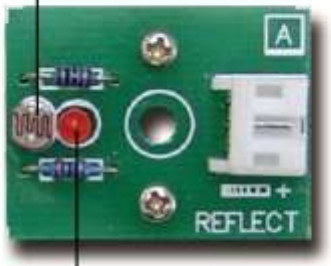
- ตัวกำเนิดแสงอินฟราเรด (เอาท์พุท)



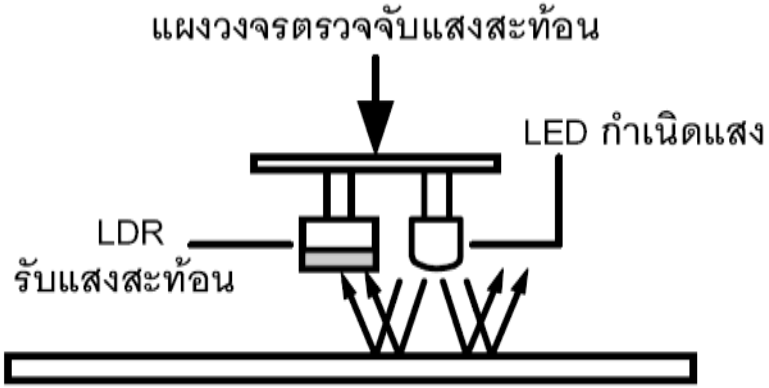
- โฟโต้ทรานซิสเตอร์อินฟราเรด (อินพุท)
  - แสงมากให้ค่าน้อย

วงจรจับแสงสะท้อน

- LDR ตัวต้านทานแปรค่าตามแสง

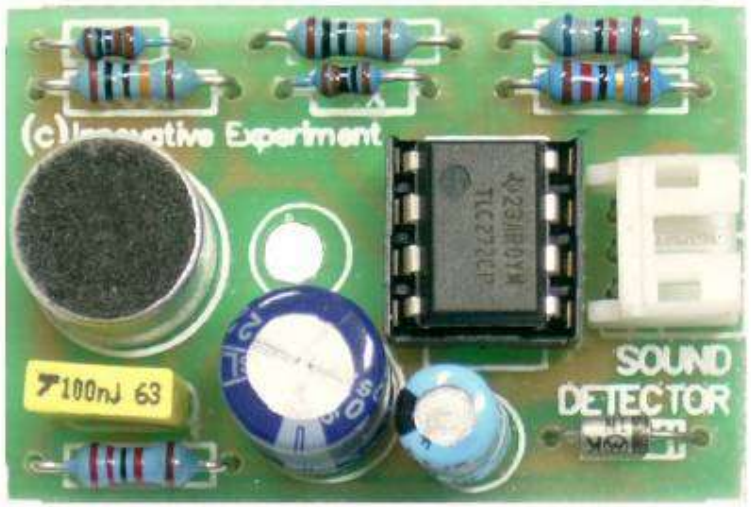


ไดโอดเปล่งแสงหรือ LED สีแดง
แบบความสว่างสูง

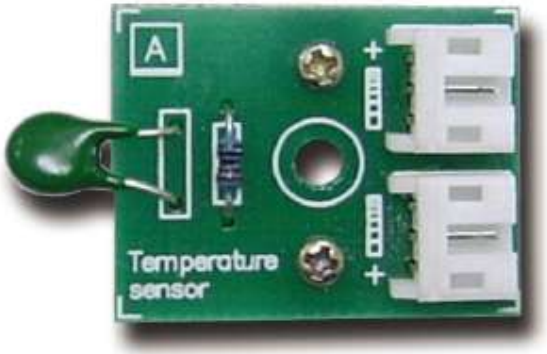


วงจรรับเสียง

- เสียงดังให้ค่ามาก



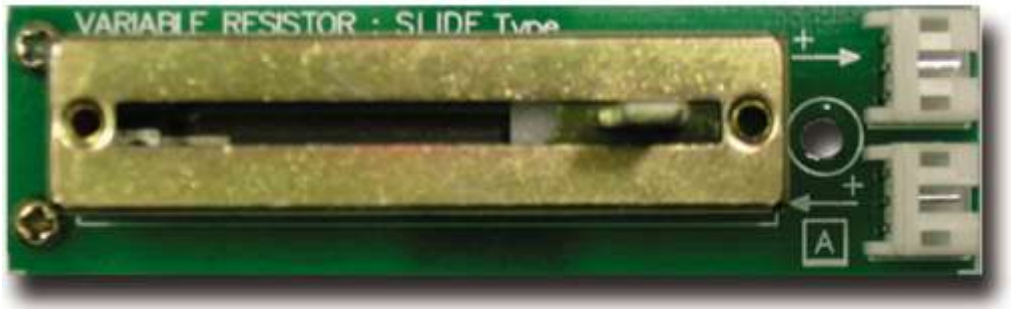
วงจรวัดอุณหภูมิ



อุณหภูมิสูงขึ้นให้ค่าน้อยลง

อุณหภูมิสูงขึ้นให้ค่ามากขึ้น

วงจรตัวต้านทานปรับค่าได้



เลื่อนมาทางขวาได้ค่ามากขึ้น

เลื่อนไปทางซ้ายได้ค่ามากขึ้น

คำสั่งอ่านค่าแอนะล็อก

- ไฟล์เฮดเดอร์ที่ใช้
  - **avr/io.h, analog.h**
- คำสั่งในไลบรารี มีเพียงคำสั่งเดียวคือ
  - **analog(*ch*)** – คืนค่าระหว่าง 0 ถึง 1023 เพื่อบ่งบอกระดับแรงดัน (สัญญาณแอนะล็อก) ที่วัดได้ที่ขา *ch* ของพอร์ต A

โปรแกรมทดสอบอุปกรณ์แอนะล็อก

- ต่อตัวต้านทานแกนเลื่อนเข้าที่พอร์ต PA0
- ทดสอบโปรแกรมต่อไปนี้

```
#include <avr/io.h>
#include <in_out.h>
#include <segment.h>
#include <analog.h>

void main()
{
  while(1) segment(in_a(0));
}
```

จากนั้นเปลี่ยนเป็นคำสั่ง analog(0)

ตัวอย่าง - *แข่งปรบมือ*

- ต่อไมโครโป
- ปรบมือจาก ให้ได้

```
#include <avr/io.h>
#include <in_out.h>
#include <segment.h>
#include <analog.h>

void main()
{
  uint16_t aval, max = 0;
  while(1)
  {
    aval = analog(0);
    if (aval > max) max = aval;
    segment(max);
  }
}
```

แบบฝึกหัด - *เปิดปิดไฟด้วยแสง*

- จำลองการควบคุมไฟภายในบ้านด้วยแสง
- *ฮาร์ดแวร์ที่ใช้:* แผงวงจร LED สองสี, แผงวงจรวัดแสง, ตัวต้านทานปรับค่าได้
- *การทำงาน:*
  - สว่าง → LED ดับ
  - มืด → LED ติด
  - กำหนดความไวด้วยวงจรความต้านทานปรับค่าได้

เฉลย - *เปิดปิดไฟด้วยแสง*

ต่อตัววัดแสงที่ PA0 (แสงน้อยค่ามาก)

ต่อตัวต้านทานที่ PA1

ต่อ LED ที่ PA5 ช่อง HIGH

```
#include <avr/io.h>
#include <in_out.h>
#include <analog.h>

void main()
{
  while(1)
  {
    if (analog(0) > analog(1))
      out_a(5,1);
    else
      out_a(5,0);
  }
}
```

แบบฝึกหัด - *เปิดปิดไฟด้วยเสียง*

- เปิดปิดไฟโดยใช้วิธีปรบมือ
- *ฮาร์ดแวร์ที่ใช้:* LED สองสี, วงจรรับเสียง, ตัวต้านทานปรับค่าได้
- *การทำงาน:*
  - ปรบครั้งแรกให้ไฟ (LED) ติด
  - ปรบอีกครั้งให้ไฟ (LED) ดับ
  - ใช้ตัวต้านทานปรับค่าได้กำหนดความไว

เฉลย - *เปิดปิดไฟด้วยเสียง*

ต่อไมโครโฟนที่ PA0

ต่อตัวต้านทานที่ PA1

ต่อ LED ที่ PA5 ช่อง HIGH

```
#include <avr/io.h>
#include <in_out.h>
#include <analog.h>
#include <sleep.h>

void main()
{
  out_a(5,0); // initially on
  while(1)
  {
    if (analog(0) > analog(1))
    {
      toggle_a(5);
      sleep(1000);
    }
  }
}
```

แบบฝึกหัด – *นับคนเข้าร้าน*

- จำลองอุปกรณ์ตรวจจับคนเดินเข้าหน้าร้าน
- เมื่อคนเดินผ่านให้เพิ่มค่าบน 7-Segment (พร้อมนิ้งหน่อง)
- เลือกใช้ตัววัดตามสมควร
  - ตัววัดแสง + ไฟฉาย
  - ตัววัดอินฟราเรด + LED แบบอินฟราเรด
  - ตัววัดแสงสะท้อน
  - ตัววัดเสียง

เฉลย -

```
#include <avr/io.h>
#include <in_out.h>
#include <segment.h>
#include <analog.h>
#include <sleep.h>
#include <sound.h>

void main()
{
  uint16_t count = 0;
  segment(count); out_d(7, 1);
  while (1)
  {
    while (analog(0) < 100) {} sleep(10);
    while (analog(0) > 100) {} sleep(10);
    count++; segment(count);
    sound_d(5, 659, 500);
    sound_d(5, 523, 1000);
  }
}
```

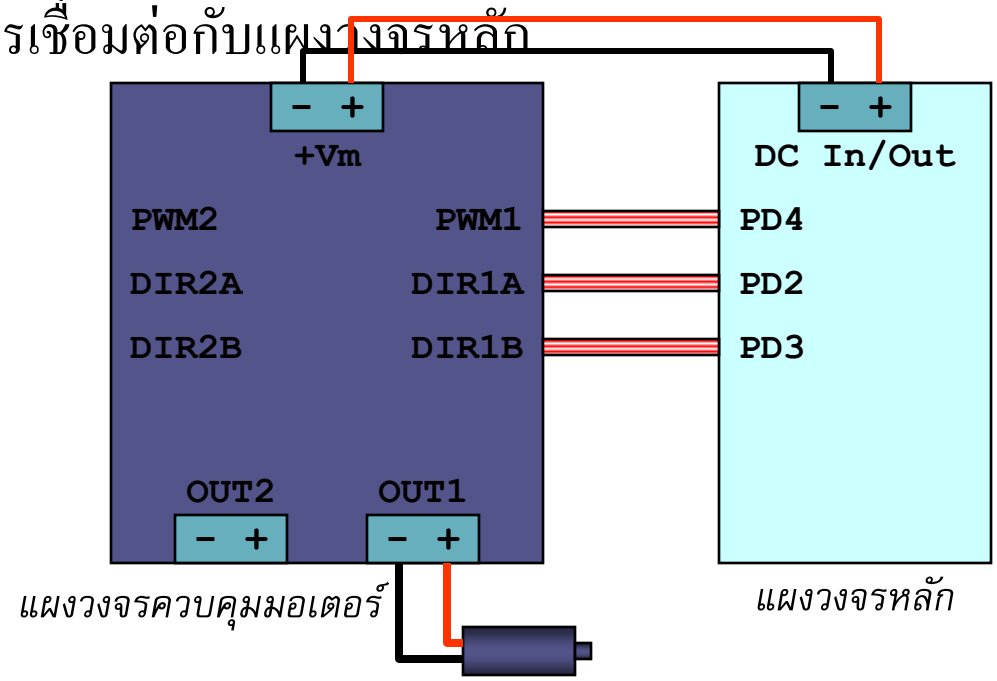
ต่อ IR LED ที่ PD7

ต่อตัวรับ IR ที่ PA0 (แสงมากค่าน้อย)

ต่อลำโพงที่ PD5

แผงวงจรควบคุมมอเตอร์ไฟตรง

- การเชื่อมต่อกับแผงวงจรหลัก



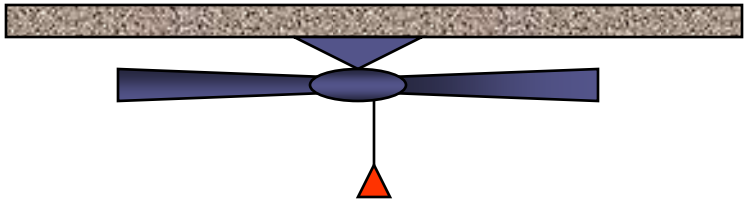
คำสั่งควบคุมมอเตอร์ไฟตรง

- ไฟล์เฮดเดอร์ที่ใช้
  - **avr/io.h, in_out.h, motor.h**
- คำสั่งในไลบรารี
  - **motor(*ch*, *pwr*)** – ขับมอเตอร์ช่อง *ch* ด้วยกำลัง *pwr*%
  - **motor_stop(*ch*)** – หยุดมอเตอร์ช่อง *ch* หรือใช้ motor_stop(ALL) เพื่อหยุดมอเตอร์ทั้งคู่

ทดสอบกา

```
#include <avr/io.h>
#include <in_out.h>
#include <motor.h>
#include <sleep.h>

void main()
{
  motor(1, 50); // เดินหน้า 50%
  sleep(10000);
  motor(1, 100); // เดินหน้า 100%
  sleep(10000);
  motor(1, -50); // ถอยหลัง 50%
  sleep(10000);
  motor_stop(1); // หยุด
  while (1) {}
}
```

ตัวอย่าง - *พัดลมติดเพดาน*



- กดสวิตช์ครั้งแรกเปิด กด

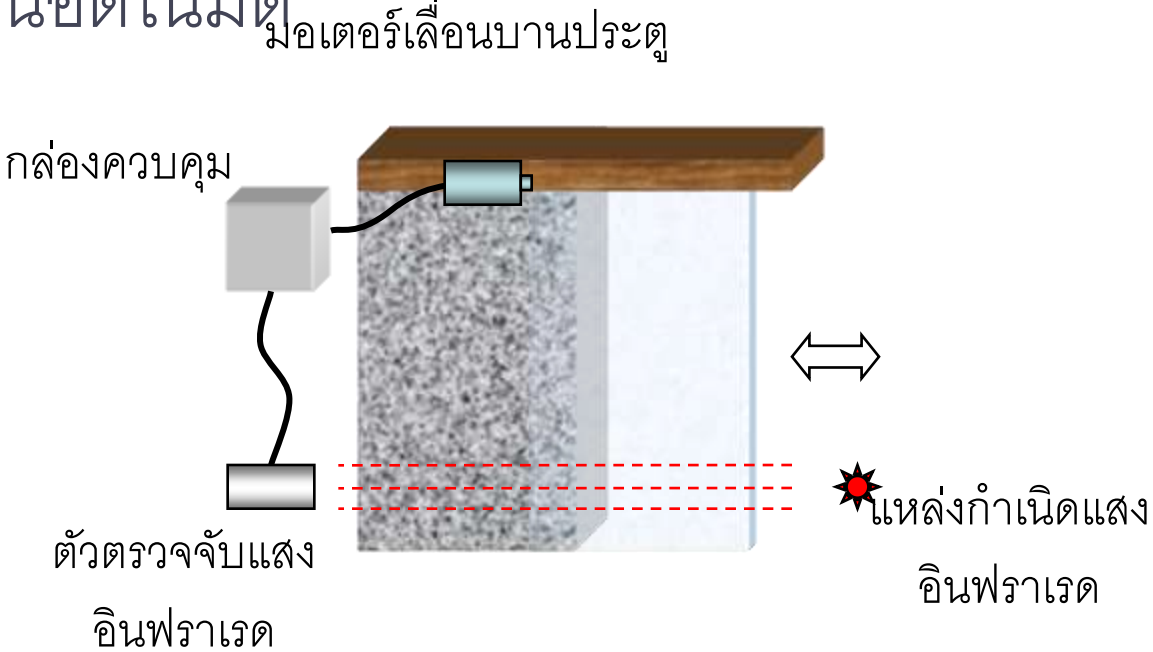
```
#include <avr/io.h>
#include <in_out.h>
#include <motor.h>
#include <sleep.h>

uint8_t switch_d(uint8_t x)
{
  if (in_d(x) == 1)
  {
    sleep(5); while (in_d(x) == 1) {}
    sleep(5); return 1;
  }
  else return 0;
}
```

```
void main()
{
  uint16_t power = 0;
  while (1)
  {
    if (switch_d(7) == 1)
    {
      power = power + 33;
      if (power > 100)
      {
        power = 0;
        motor_stop(1);
      }
      motor(1, power);
    }
  }
}
```

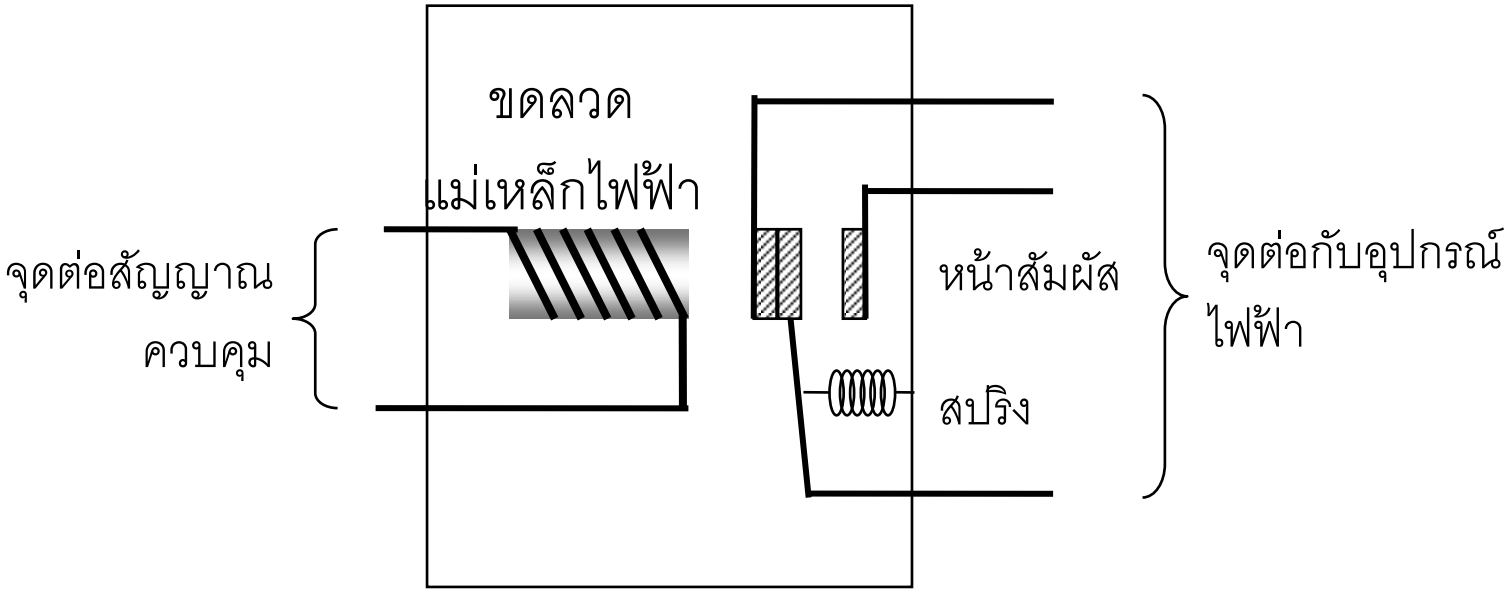
แบบฝึกหัด - *พัดลมสมองกล*

- จำลองการทำงานของพัดลมด้วยมอเตอร์ โดยปรับความแรงให้สอดคล้องกับสภาพแวดล้อม
- *การทำงาน*:
  - ใช้สวิตช์เพื่อเปิดและปิดเท่านั้น
  - อากาศเย็น (ต่ำกว่า $25^{o}C$) พัดลมหมุนช้า
  - อากาศร้อนแต่มืด พัดลมหมุนปานกลาง
  - อากาศร้อนและสว่าง พัดลมหมุนเร็ว

ประตูเลื่อนอัตโนมัติ

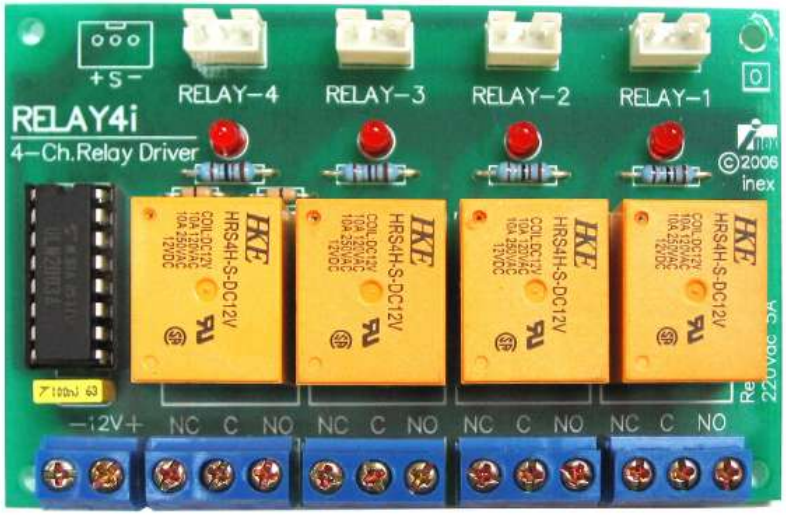


รีเลย์ (Relay)


ขดลวด
แม่เหล็กไฟฟ้า
จุดต่อสัญญาณ
ควบคุม
หน้าสัมผัส
สปริง
จุดต่อกับอุปกรณ์
ไฟฟ้า

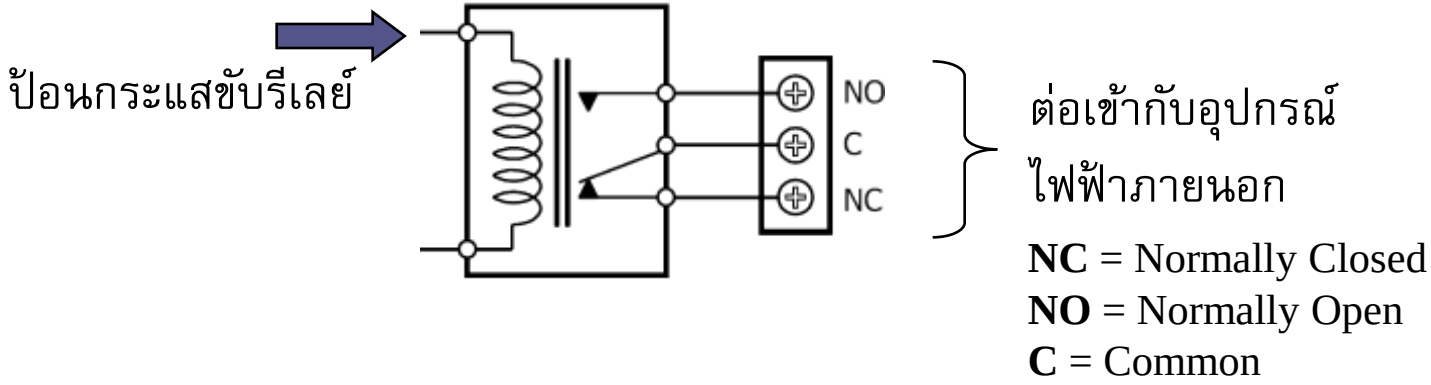
แผงวงจรควบคุมรีเลย์

- ใช้ควบคุมอุปกรณ์ไฟฟ้าภายนอก โดยเฉพาะพวกที่ต้องใช้กับแรงดันหรือกระแสสูง ๆ

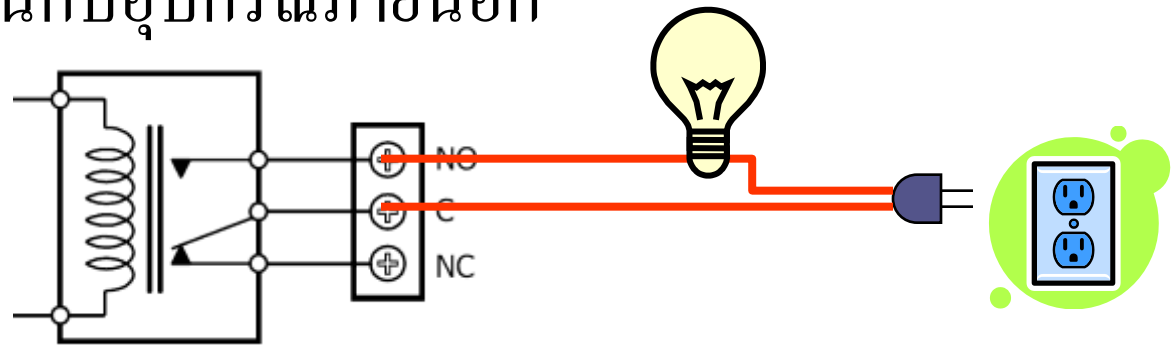


- การทำงาน

ภายในรีเลย์และการใช้งาน

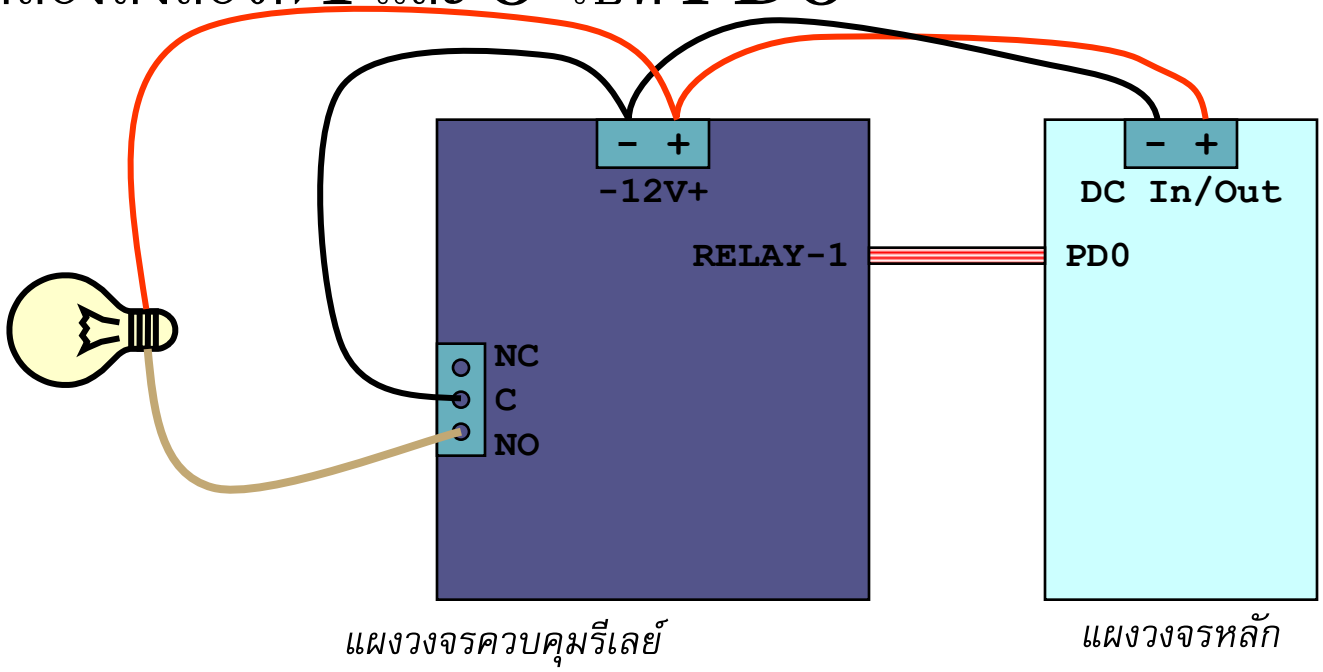


- การต่อใช้งานกับอุปกรณ์ภายนอก



ตัวอย่างการเชื่อมต่อแผงควบคุมรีเลย์

- ช่อง RELAY-n แต่ละช่องสามารถต่อกับพอร์ตใดก็ได้ ยกเว้นพอร์ต C
- ทดลองส่งลอจิก 1 และ 0 ไปที่ PD0



แบบฝึกหัด - *เปิดปิดไฟด้วยแสง*

- จำลองการควบคุมไฟภายในบ้านด้วยแสง (เหมือนแบบฝึกหัดก่อนหน้า)
- *ฮาร์ดแวร์ที่ใช้:* หลอดไฟ, แผงวงจรวัดแสง, ตัวต้านทานปรับค่าได้, แผงวงจรควบคุมรีเลย์
- *การทำงาน:*
  - สว่าง → หลอดไฟดับ
  - มืด → หลอดไฟติด
  - กำหนดความไวด้วยวงจรความต้านทานปรับค่าได้

แบบฝึกหัด - *โรงเรือนอัจฉริยะ*

- สร้างระบบปรับอากาศให้โรงเลี้ยงสัตว์
- แบบจำลอง
  - หลอดไฟ (จริง ๆ) แทนฮีตเตอร์ให้ความอบอุ่น
  - มอเตอร์ แทนปั๊มน้ำให้ความเย็น
- การทำงาน
  - ควบคุมอุณหภูมิให้อยู่ในช่วง 20-30°C
  - หากร้อนเกินไปให้เปิดปั๊มน้ำ
  - หากเย็นเกินไปให้เปิดฮีตเตอร์

คำนวณค่าแรงดัน

- จากกราฟคุณสมบัติของเทอร์มิสเตอร์
  - อุณหภูมิ 20°C ได้ความต้านทานประมาณ 12kΩ
  - อุณหภูมิ 30°C ได้ความต้านทานประมาณ 8kΩ
- ใช้สูตร $R_t = \frac{10D}{1023 - D}$ ⟺ $D = \frac{1023R_t}{10 + R_t}$
- ได้ $D_{low} = 558$, $D_{high} = 455$

เช

```
#include <avr/io.h>
#include <in_out.h>
#include <sleep.h>
#include <motor.h>
#include <analog.h>

#define LOW 558
#define HIGH 455

void main()
{
  uint16_t temp;
```

```
  while (1)
  {
    temp = analog(0);
    if (temp > LOW) // too cold
    {
      out_d(0, 1);
      motor_stop(1);
    }
    else if (temp < HIGH) // too hot
    {
      out_d(0, 0);
      motor(1, 100);
    }
    else
    {
      out_d(0, 0);
      motor_stop(1);
    }
  }
}
```

ต่อตัววัดอุณหภูมิที่ PA0

ต่อรีเลย์ที่ PD0