

# การเขียนโค้ด **Arduino Sketch** และจำลองการทำงานสำหรับ **ATtiny85 (8-bit Microcontroller)**

- เรียนรู้การใช้ภาษา **C/C++** ภายใต้บิบทการใช้งานของดิจิทัล-อิเล็กทรอนิกส์
- เรียนรู้ตัวอย่างเขียนโค้ด **Arduino Sketch** ด้วยภาษา **C/C++** สำหรับชิป **ATtiny85** ซึ่งเป็นไมโครคอนโทรลเลอร์ขนาดเล็ก
- ฝึกต่อวงจรและมีเครื่องจำลองการทำงาน และตรวจสอบความถูกต้องเบื้องต้น โดยใช้ซอฟต์แวร์ฟรี (**AUTODESK Tinkercad Circuits**)
- ทดลองต่อวงจรจริงบนบอร์ด และอัปโหลดโปรแกรมไปยังชิปไมโครคอนโทรลเลอร์

# แนะนำไมโครคอนโทรลเลอร์ ATtiny85

- เป็นชิปไมโครคอนโทรลเลอร์ในตระกูล **ATtiny (TinyAVR)** ของบริษัท **Atmel / Microchip**
- ภายในมีชีพียูขนาด 8 บิต และมีสถาปัตยกรรมแบบ **RISC**
- มีหน่วยความจำภายใน (ค่อนข้างน้อยมาก) **SRAM 512 ไบต์** **Flash 8 กิโลไบต์ (KB)** และ **EEPROM 512 ไบต์**
- ใช้ตัวถังของไอซีที่มีเพียง 8 ขา เช่น ตัวถังแบบ **DIP-8**
- มีตัวสร้างสัญญาณ **Clock** ภายใน (**internal oscillator**) ที่มีความถี่ **8 MHz**
- สามารถเขียนโคด-จำลองการทำงานร่วมกับวงจรอิเล็กทรอนิกส์ได้ โดยใช้ **AUTODESK Tinkercad – Circuits**
- รองรับการเขียนโคด โดยใช้ชุดคำสั่ง (**API**) ของ **Arduino**

# ไมโครคอนโทรลเลอร์ ATtiny85

- ATtiny85 มีราคาไม่แพง (~50 บาท) ขนาดเล็ก
- ใช้ตัวถังแบบ **DIP-8** สามารถนำไปเสียบลงบนเบรดบอร์ดได้
- สามารถใช้ตัวสร้างสัญญาณ **Clock** ภายใน โดยไม่ต้องต่อวงจร **Crystal Oscillator** ภายนอก (ไม่ต้องใช้ขา **XTAL1 / XTAL2**)
- เลือกใช้ความถี่ เช่น **1 MHz, 8 MHz** หรือ **16 MHz** ได้
  - ถ้าใช้ความถี่ **16 MHz** จะต้องใช้แรงดันไฟเลี้ยง **5V**
  - ถ้าใช้ความถี่ **1 MHz** หรือ **8 MHz** สามารถเลือกใช้แรงดันไฟเลี้ยง **3.3V** หรือ **5V** ได้

# คุณสมบัติโดยสรุปเกี่ยวกับ ATtiny85

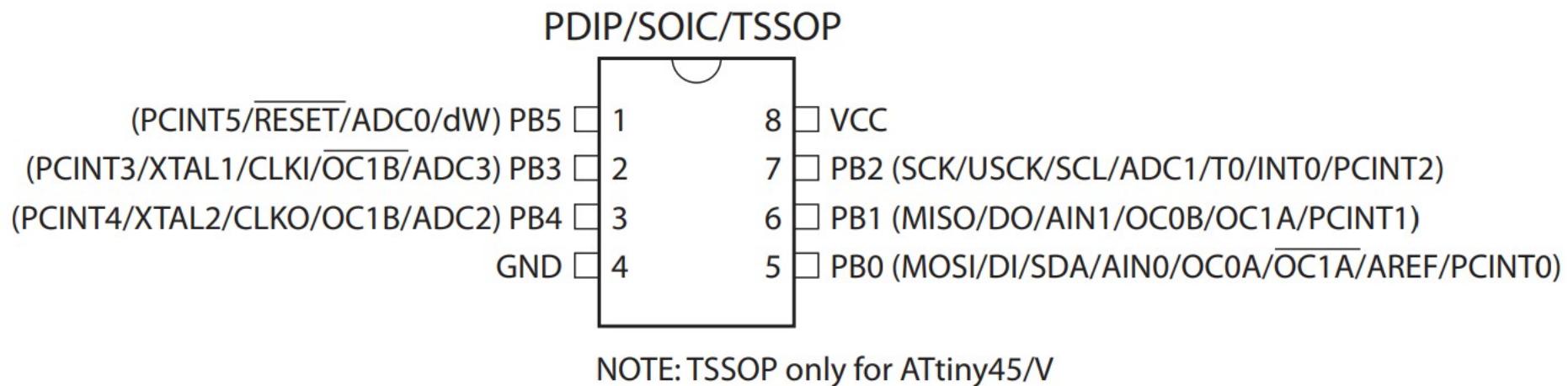
Parametrics	
Name	Value
ชนิดข้อมูล	▶ Program Memory Type Flash
ความเร็วของชิปีย์ (สูงสุด)	▶ Program Memory Size (KB) 8 ← ขนาดหน่วยความจำสำหรับโปรแกรม
ขนาดหน่วยความจำ EEPROM	▶ CPU Speed (MIPS/DMIPS) 20 ← ▶ SRAM (bytes) 512 ← ขนาดหน่วยความจำ SRAM
จำนวนสัญญาณ PWM	▶ Data EEPROM/HEF (bytes) 512 ← ▶ Digital Communication Peripherals 1-SPI, 1-I2C ← จำนวนวงจรสำหรับสื่อสารข้อมูลแบบ SPI & I2C
จำนวนวงจรเปรียบเทียบแรงดันไฟฟ้า	▶ Capture/Compare/PWM Peripherals 5PWM ← ▶ Timers 2 x 8-bit ← จำนวนวงจรตัวนับ 8-bit Timers
ช่วงแรงดันไฟเลี้ยง	▶ Number of Comparators 1 ← ▶ Temperature Range (°C) -40 to 85 ← ช่วงอุณหภูมิในการทำงานของไอซี
	▶ Operating Voltage Range (V) 1.8 to 5.5 ← จำนวนขาของไอซี
	▶ Pin Count 8 ←

# ตำแหน่งขาของ ATtiny85 (PinOut)

ขาหมายเลข 1 ตรงกับ PB5 ปกติจะถูกใช้เป็นขา /RESET และใช้สำหรับ ISP Programmer ในการโปรแกรมชิป ดังนั้นจึงไม่นิยมใช้เป็นขา I/O ทั่วไป

ขา Digital I/Os ได้แก่ PB0..PB5  
ขา Analog Inputs ได้แก่ ADC0 .. ADC3

Pinout ATtiny25/45/85



[http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-2586-AVR-8-bit-Microcontroller-ATtiny25-ATtiny45-ATtiny85\\_Datasheet.pdf](http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-2586-AVR-8-bit-Microcontroller-ATtiny25-ATtiny45-ATtiny85_Datasheet.pdf)

# ตัวอย่างที่ 1: LED Blink (Circuit View)

Circuits attiny85\_led\_blink\_2 | Tin x +

tinkercad.com/things/kD79t2vq6sl-attiny85ledblink2/editel

attiny85\_led\_blink All changes saved

Simulator time: 00:00:01

Code Stop Simulation Export Share

การต่อวงจร LED ที่ขา PB0 และ  
เขียนโค้ด เพื่อทำให้ LED กระพริบ

**Power supply (Vcc = +5V)**

5.00 V  
13.5 mA

ON VOLTAGE SUPPLY

ATTINY

<https://www.tinkercad.com/>

# ตัวอย่างที่ 1: LED Blink (Component List)

รายการอุปกรณ์สำหรับวงจรในตัวอย่างที่ 1  
(ไม่รวมเบรดบอร์ดและสายไฟ)



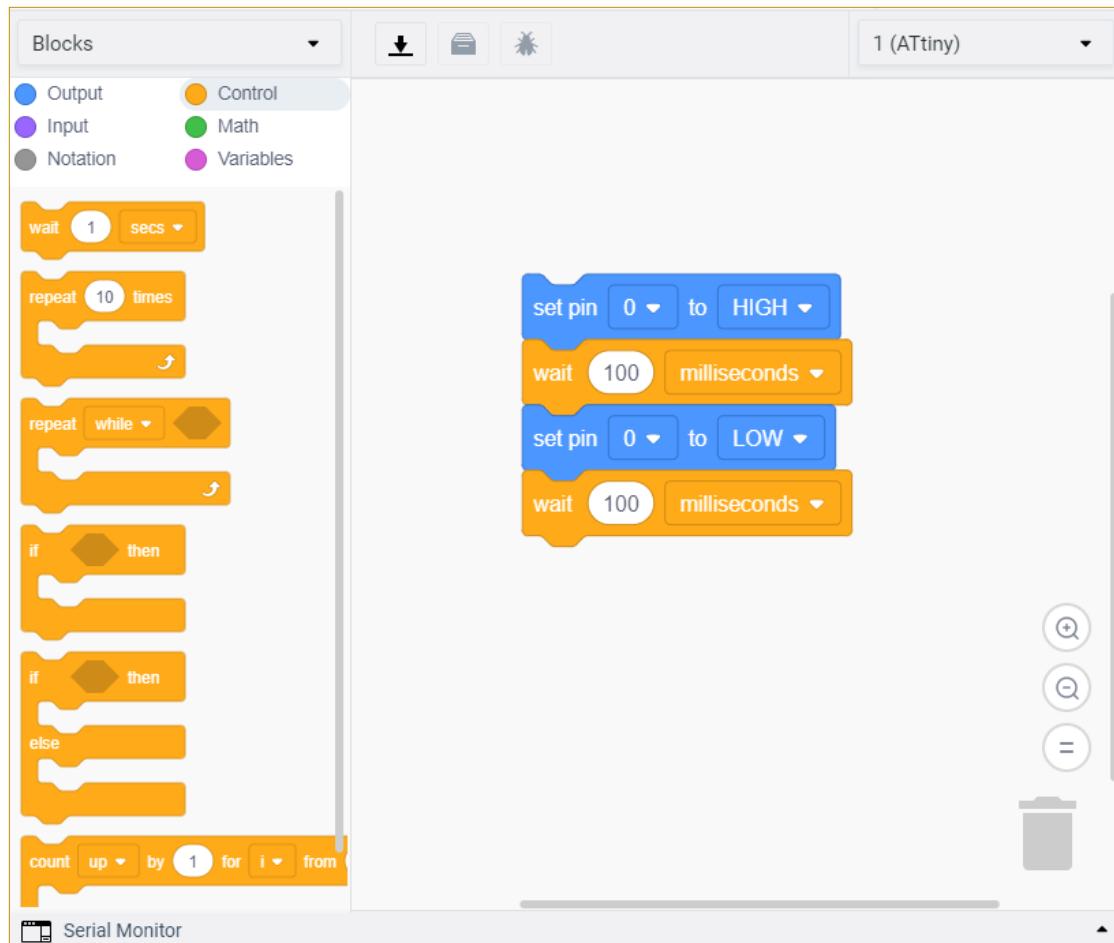
TINKERCAD attiny85\_led\_blink All changes saved

Component List Download CSV

Name	Quantity	Component
U1	1	ATtiny
P2	1	5,5 Power Supply
D1	1	Red LED
R1	1	470 Ω Resistor

# ตัวอย่างที่ 1: LED Blink (Code)

มุมมองการเขียนโค้ดด้วยการต่อบล็อก  
(**Block-based Coding**)



มุมมองการเขียนโค้ดด้วย  
ภาษาคอมพิวเตอร์ **C/C++**

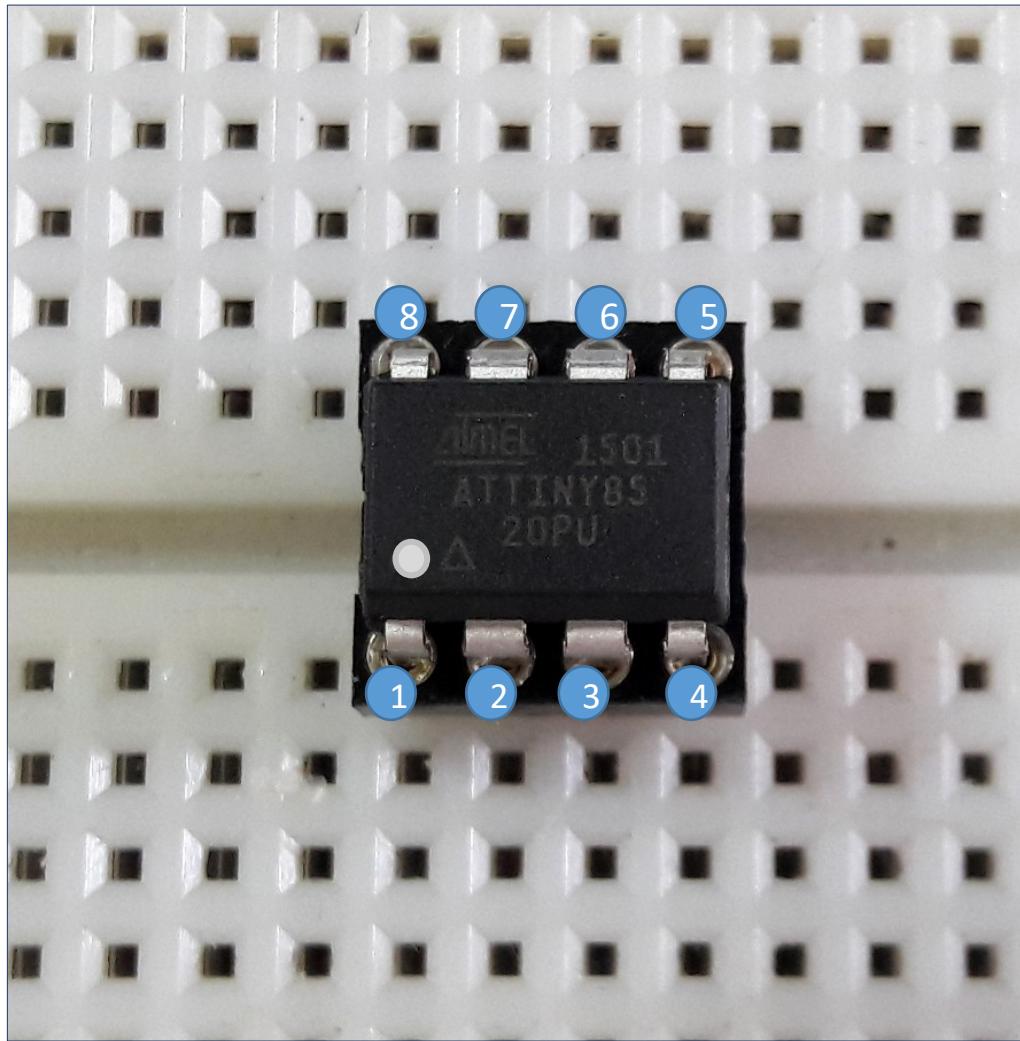
```
void setup() {  
    pinMode( 0, OUTPUT );  
}  
  
void loop() {  
    digitalWrite( 0, HIGH );  
    delay(100);  
    digitalWrite( 0, LOW );  
    delay(100);  
}
```

# การต่อวงจรเพื่อทดลองด้วยชาร์ดแวร์จริง

## รายการอุปกรณ์ (สำหรับตัวอย่าง LED Blink)

- เบอร์ดบอร์ด
- ไอซี **ATTiny85**
- หลอด **LED (Red)** ขนาด 5 มม.
- ตัวต้านทาน 470 โอห์ม สำหรับต่ออนุกรมกับ **LED**
- อุปกรณ์ **USB ISP Programmer**
- สายไฟต่อวงจร
- คอมพิวเตอร์ + ซอฟต์แวร์ **Arduino IDE**

# ATtiny85-20PU + DIP8 Socket



# Atmel ISP Programmer (USBasp)



อุปกรณ์สำหรับแปลง  
คุณเนกเตอร์ 2x5  
ให้เป็น 2x3 Pins  
(ICSP Header)

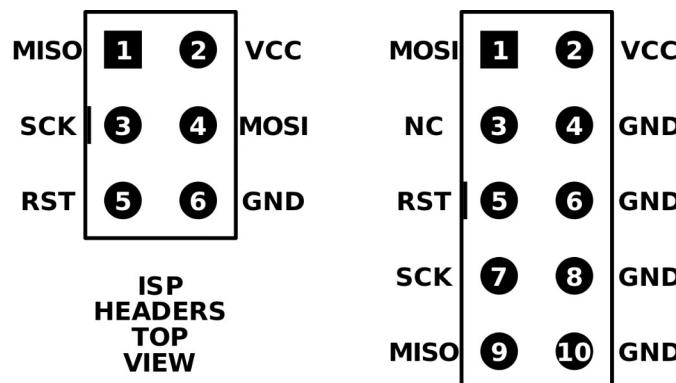
ถ้าไม่ใช้อุปกรณ์ในลักษณะนี้ ก็สามารถใช้ Arduino Uno  
ทำหน้าที่เป็น Arduino ISP Programmer ได้  
<https://www.arduino.cc/en/tutorial/arduinoISP>

ตัวอย่างอุปกรณ์ USB ISP Programmer (Clone ราคาถูก) ใช้สำหรับการอัปโหลด  
เฟิร์มแวร์ (Firmware) จากคอมพิวเตอร์ไปยัง ATTiny85 ในวงจร

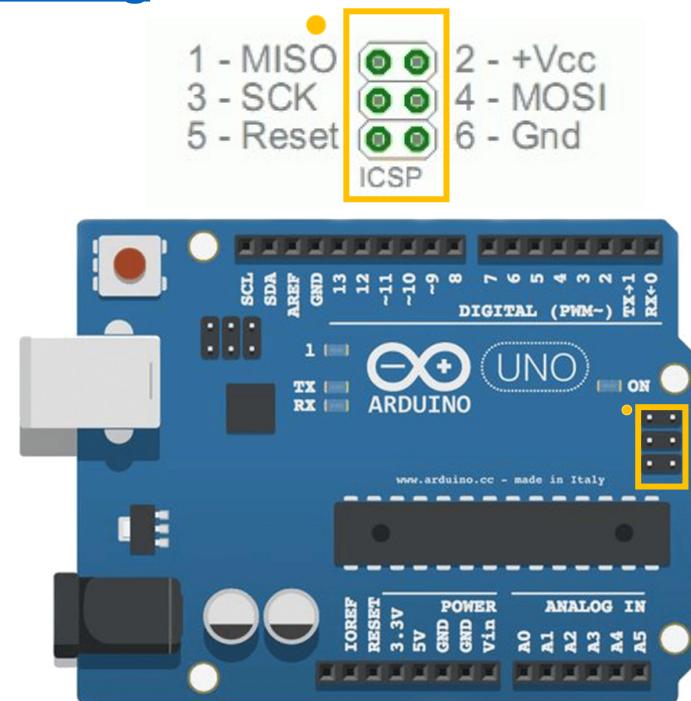
# ISP / ICSP for Arduino / AVR

**“In-system programming (ISP), also called in-circuit serial programming (ICSP), is the ability of some programmable logic devices, microcontrollers, and other embedded devices to be programmed while installed in a complete system, rather than requiring the chip to be programmed prior to installing it into the system.”**

[https://en.wikipedia.org/wiki/In-system\\_programming](https://en.wikipedia.org/wiki/In-system_programming)

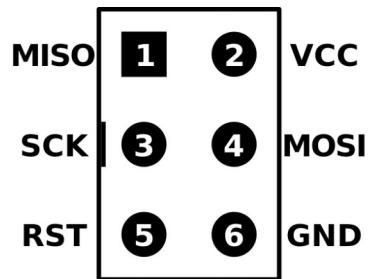


**6-pin and 10-pin AVR ISP headers**



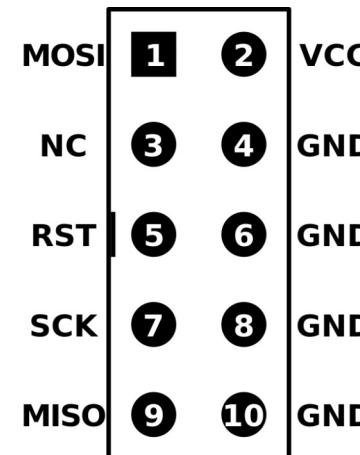
# ICSP-to-ATtiny85 Wiring

**6-pin AVR  
ISP header**

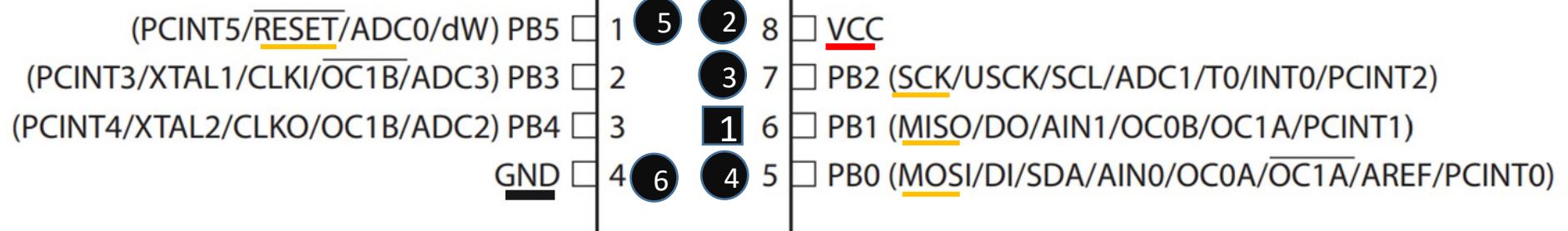


ISP  
HEADERS  
TOP  
VIEW

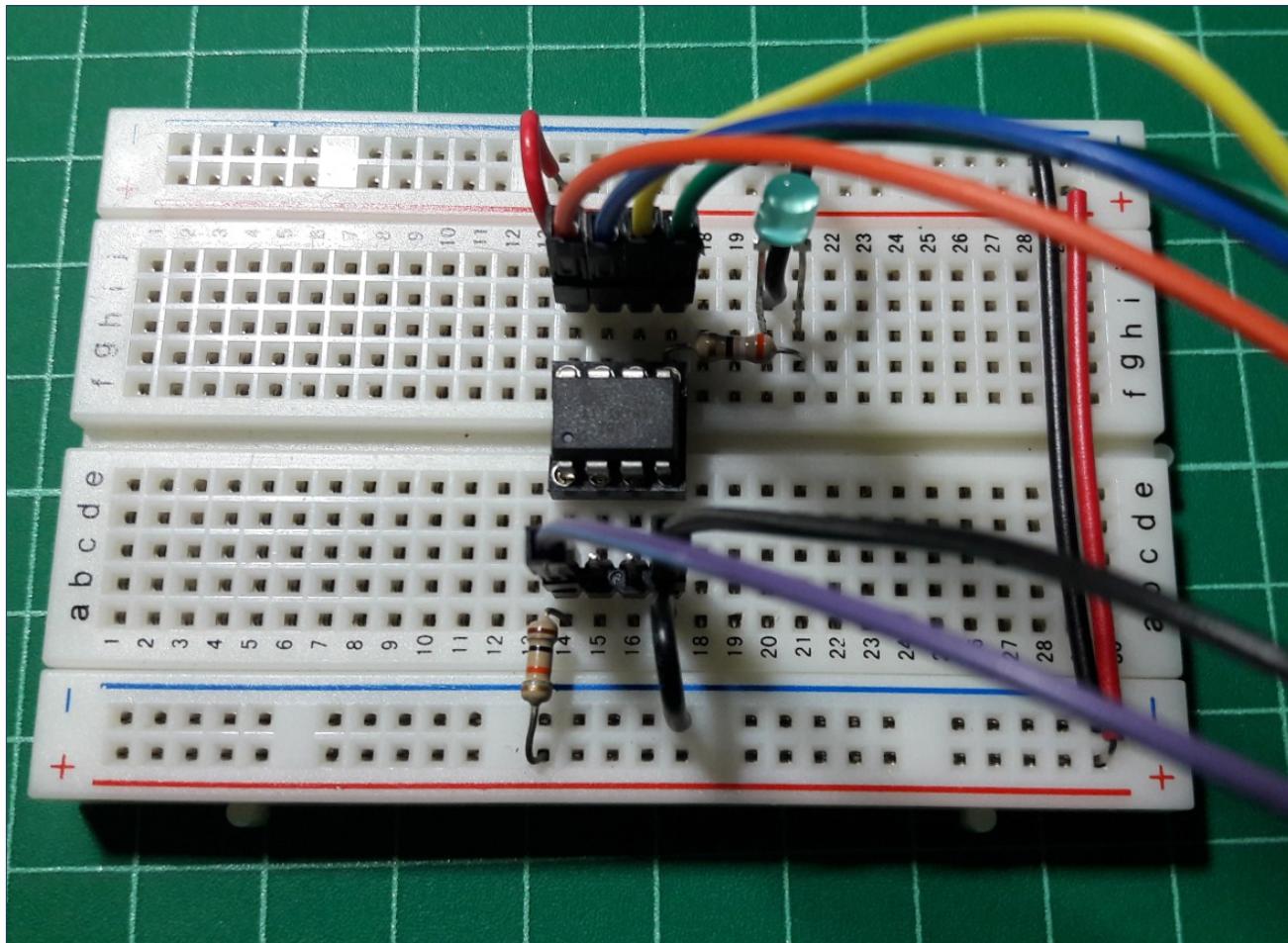
**10-pin AVR  
ISP header**



**PDIP/SOIC/TSSOP**



# ตัวอย่างการต่อวงจรบนเบรดบอร์ด



**ข้อสังเกต:** ในการต่อวงจรทดลอง สามารถใช้แรงดันไฟเลี้ยง **+5V** จาก **ISP Programmer** ได้ ถ้าใช้ปริมาณกระแสไม่มาก แต่ถ้าจะนำไปใช้งานโดยทั่วไป แนะนำให้ใช้แหล่งจ่ายแรงดันคงที่ หรือวงจรควบคุมแรงดันคงที่ เช่น ไอซี **7805**

# การใช้ซอฟต์แวร์ WokWi Simulator

The screenshot shows the WokWi Simulator interface. On the left, there is a code editor window titled "arduino\_attiny85\_demo-1.ino" containing the following Arduino sketch:

```
1 const int LED_PIN = PB0; // LED output pin
2
3 void setup() {
4     pinMode( LED_PIN, OUTPUT );
5 }
6
7 void loop() {
8     digitalWrite( LED_PIN, !digitalRead(LED_PIN) );
9     delay(500);
10}
11
```

On the right, the "Simulation" tab is active, displaying a schematic diagram of a circuit. The circuit consists of an ATTINY85 microcontroller at the bottom, connected to two blue and red LEDs via green wires. Each LED is connected in series with a resistor. The resistors are represented by brown components with three colored bands (brown, black, orange). The simulation status bar at the top right shows "00:06.750" and "99%".

# การคอมไพล์โค้ดและอัปโหลดลงชิป ATTiny85

The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** attiny85\_led\_blink | Arduino 1.8.9
- Menu Bar:** File Edit Sketch Tools Help
- Toolbar:** Includes icons for Open, Save, Upload, and Download.
- Code Editor:** Displays the following sketch code:

```
int LED_PIN = PB0;

void setup() {
  pinMode( LED_PIN, OUTPUT );
}

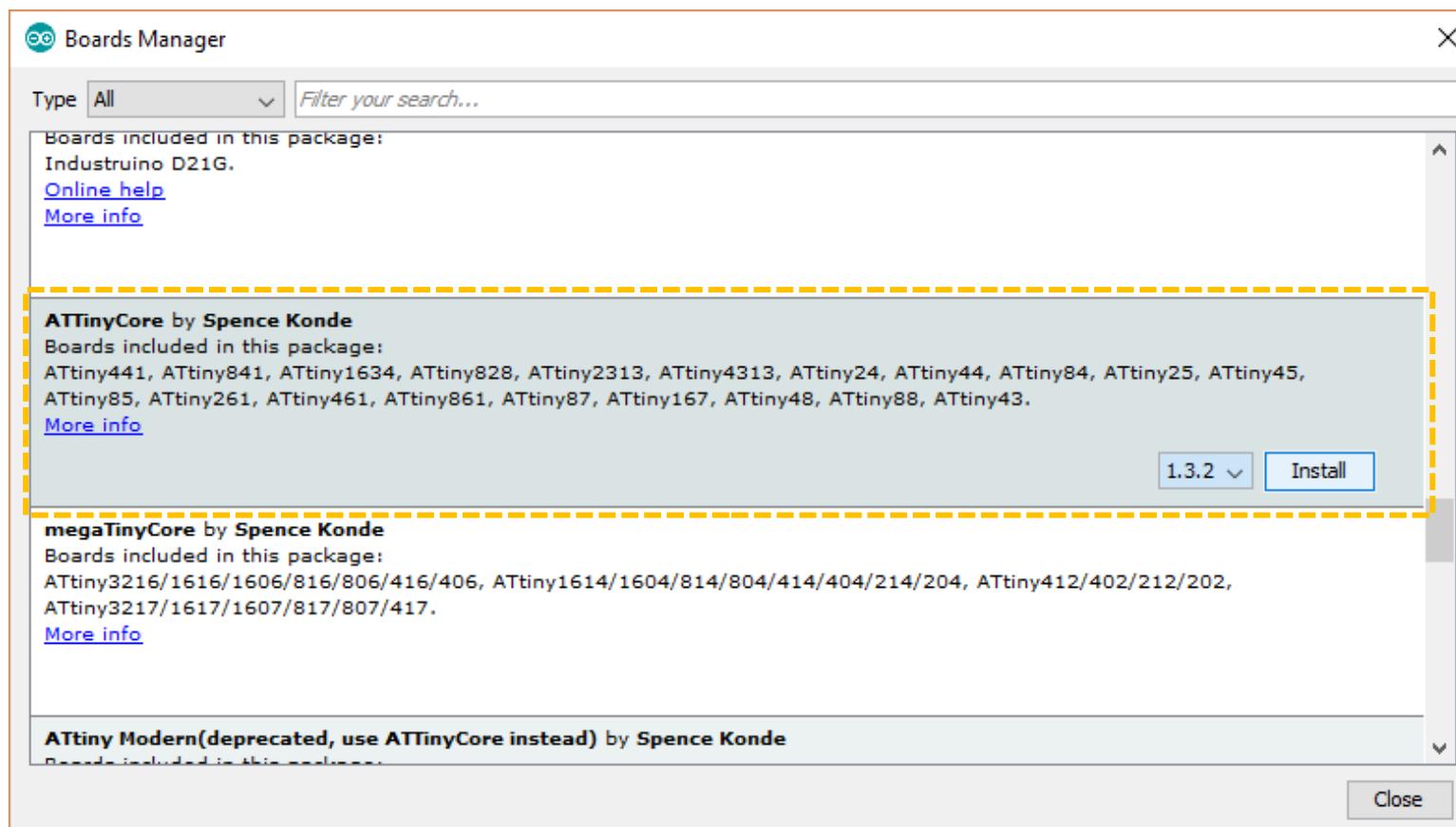
void loop() {
  digitalWrite( LED_PIN, HIGH );
  delay(100);
  digitalWrite( LED_PIN, LOW );
  delay(100);
}
```
- Status Bar:** Done uploading.
- Serial Monitor:** Shows the output of the avrdude command:

```
avrduude: set SCK frequency to 187500 Hz
avrduude: set SCK frequency to 187500 Hz
```
- Bottom Navigation:** Includes a page number (13) and a status bar indicating the connection: ATtiny25/45/85 on COM99.

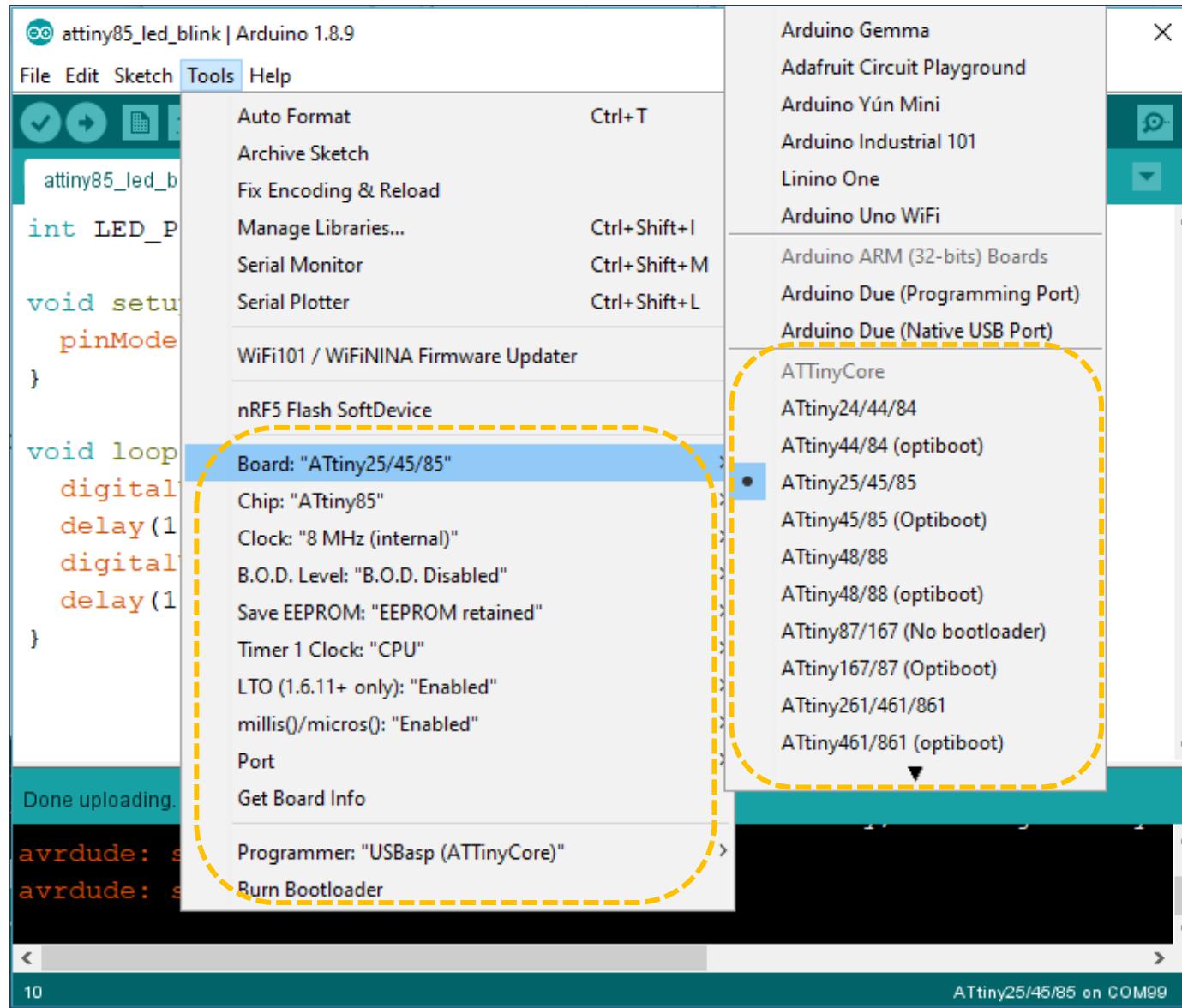
นำโค้ดที่ได้จำลองและทดสอบการทำงานแล้วมาสร้างเป็น Sketch ใน Arduino IDE (Windows 10)

# การติดตั้ง ATTinyCore ใน Arduino IDE

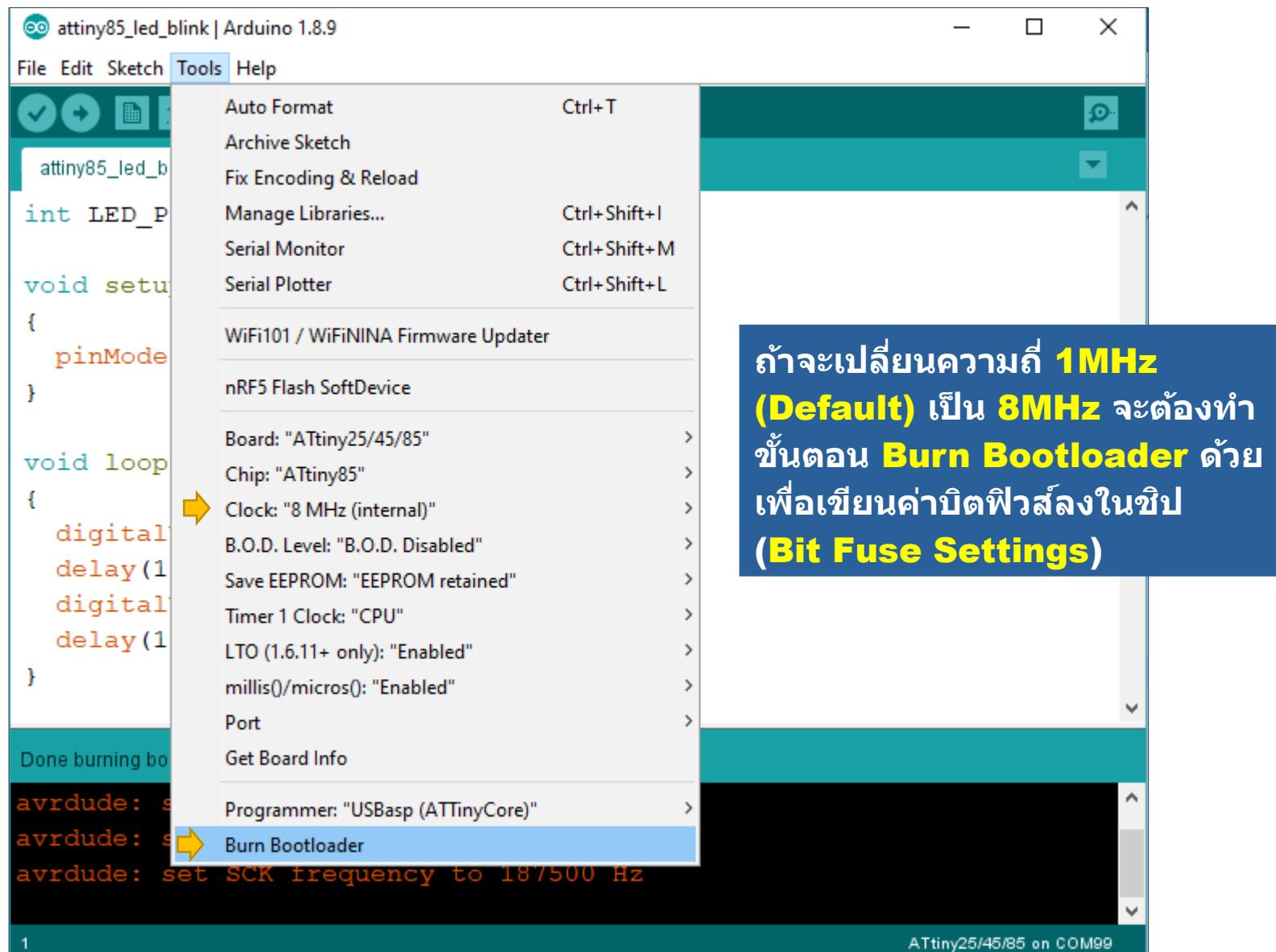
ในส่วน **Preferences** ของ **Arduino IDE** ให้ใส่ **URL** ในช่อง **Additional Boards Manager URLs:** [http://drazzy.com/package\\_drazzy.com\\_index.json](http://drazzy.com/package_drazzy.com_index.json)  
(ถ้ามี **URL** อื่น ๆ ให้ไข้สัญลักษณ์ , เป็นตัวแบ่ง) จากนั้นทำการคำสั่งจากเมนู **Tools > Boards Manager** เพื่อดาวน์โหลดและติดตั้ง **ATTinyCore** ตามรูปด้านล่าง



# การเลือกใช้ ATTiny85 ใน Arduino IDE

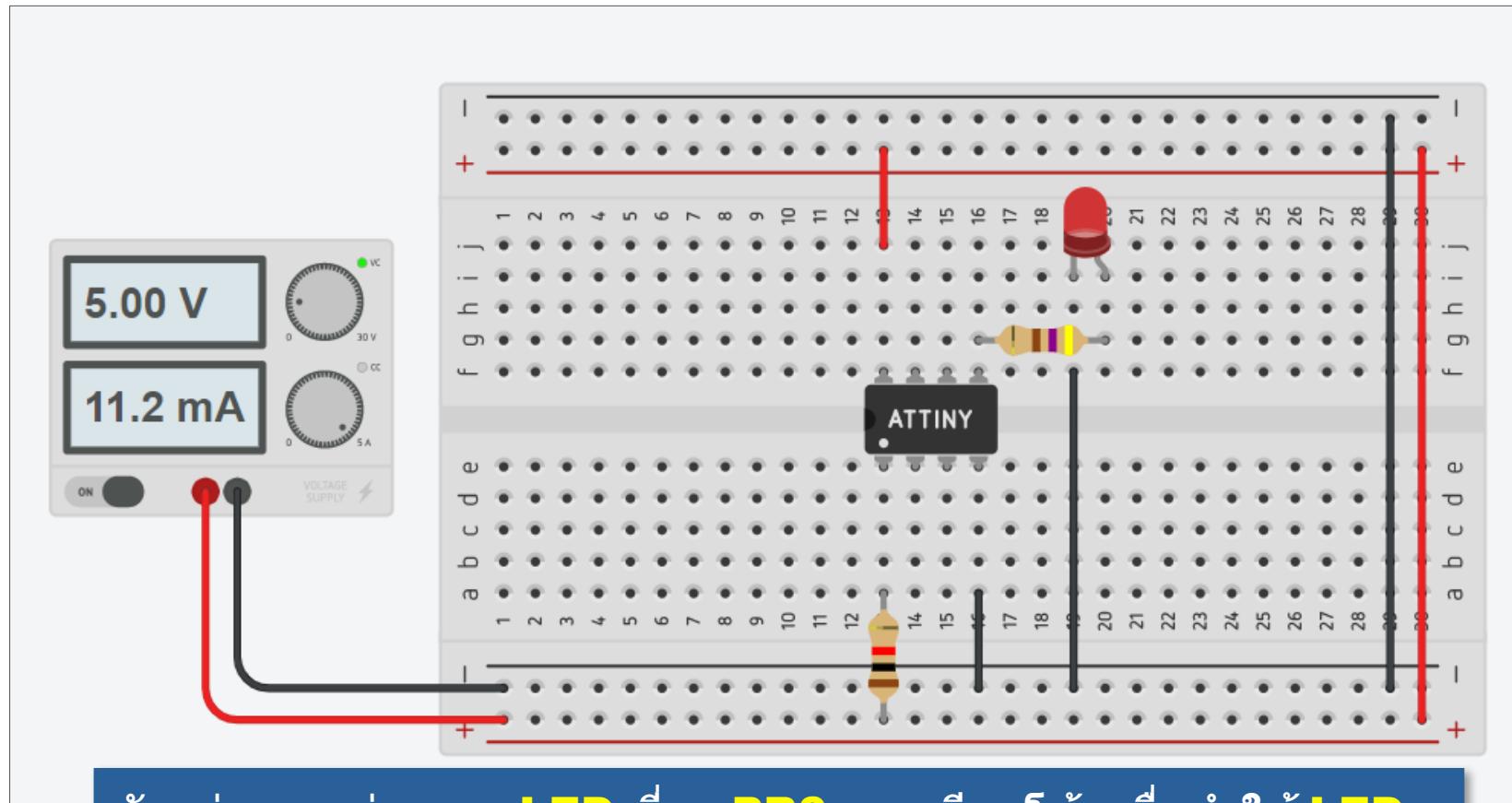


# การเลือกใช้ความถี่ 8 MHz (Internal)



# ตัวอย่างที่ 2: PWM LED Dimming

โจทย์ฝึกหัด: จงวดผังวงจรเพื่อต่อวงจรบนเบรดบอร์ดในตัวอย่างที่ 2



ตัวอย่างการต่อวงจร LED ที่ขา PB0 และเขียนโค้ดเพื่อทำให้ LED  
ค่อย ๆ เพิ่มหรือลดความสว่าง

ข้อสังเกต: ต่อวงจรเหมือนตัวอย่างที่ 1 ถ้าต่อวงจรใช้งานจริง แนะนำให้ต่อ  
ตัวต้านทาน 10k แบบ Pull-up ที่ขานหมายเลข 1 (ขาเรียกเข้า)

# ตัวอย่างที่ 2: PWM LED Dimming

โจทย์ฝึกหัด: จงต่อวงจรทดลองและทดสอบการทำงานของโค้ดในตัวอย่างที่ 2

ใช้ตัวแปร **value**  
เป็นตัวนับ ที่มีค่าอยู่  
ในช่วง **0..255** เพื่อ  
ใช้กับคำสั่ง  
**analogWrite()**  
และกำหนดความ  
สว่างของ วงจร  
**LED** ที่ขา **PB0**

ค่าของตัวแปร  
**value** เป็นจะเพิ่มขึ้น<sup>เพิ่ม</sup>  
จนถึง **255** และจะ<sup>ลดลง</sup>  
ลดลงถึง **0** ครั้งละ  
**16**

```
const int LED_PIN = PB0; // LED output pin

int value = 0; // value for the PWM duty cycle
int dir = 1; // counting direction (DOWN=0 or UP=1)

void setup() {
    pinMode( LED_PIN, OUTPUT );
}

void loop() {
    analogWrite( LED_PIN, value ); // update PWM output
    value += dir ? 16 : -16; // update next value
    if ( value > 255 ) {
        value = 255;
        dir = 0; // change direction: count down
    }
    else if ( value < 0 ) {
        value = 0;
        dir = +1; // change direction: count up
    }
    delay(100);
}
```

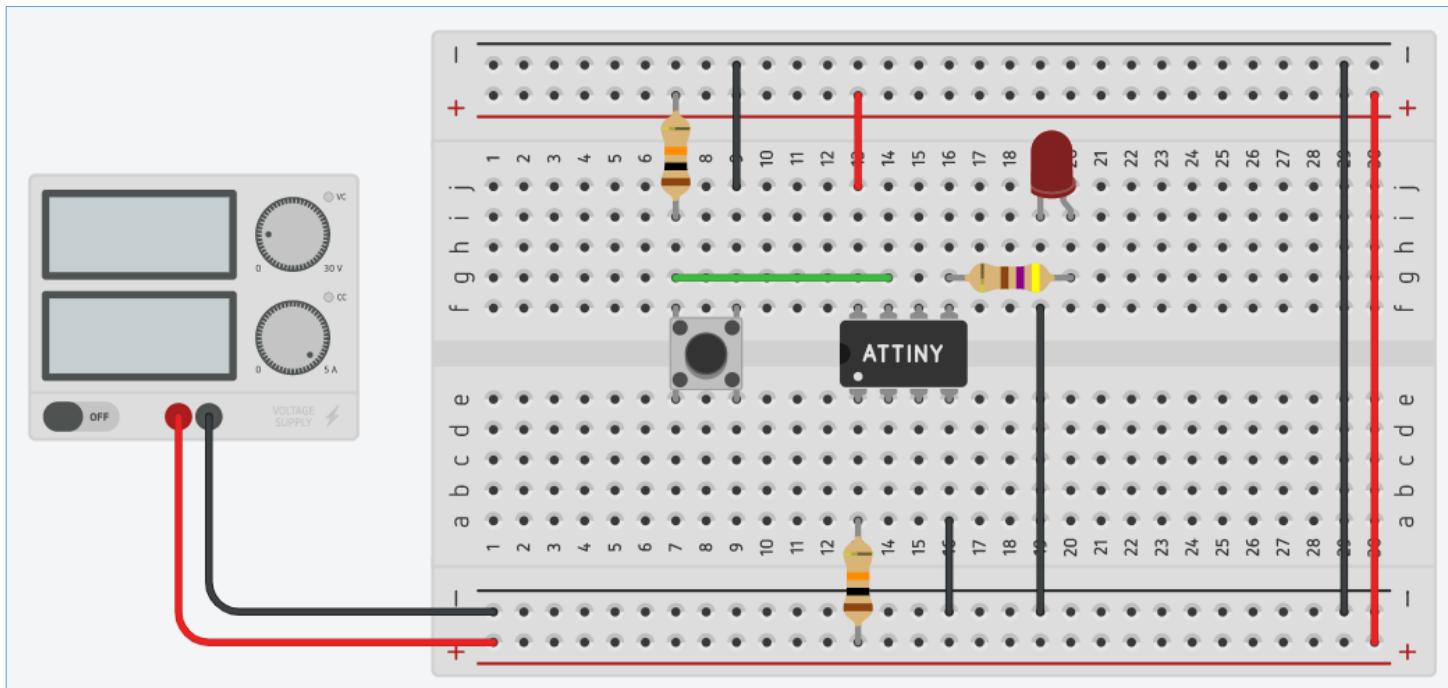
อัตราการเพิ่มขึ้น  
หรือลดลงสำหรับ<sup>เพิ่ม</sup>  
ค่าของตัวแปร  
**value** ข้าหรือเร็ว<sup>ลดลง</sup>  
ขึ้นอยู่กับคำสั่ง<sup>เพิ่ม</sup>  
**delay(...)**

# ตัวอย่างที่ 3: Push Button - LED Toggle

โจทย์ฝึกหัด: จงวาดผังวงจร (Schematic) สำหรับวงจรบนเบรดบอร์ดในตัวอย่างที่ 3

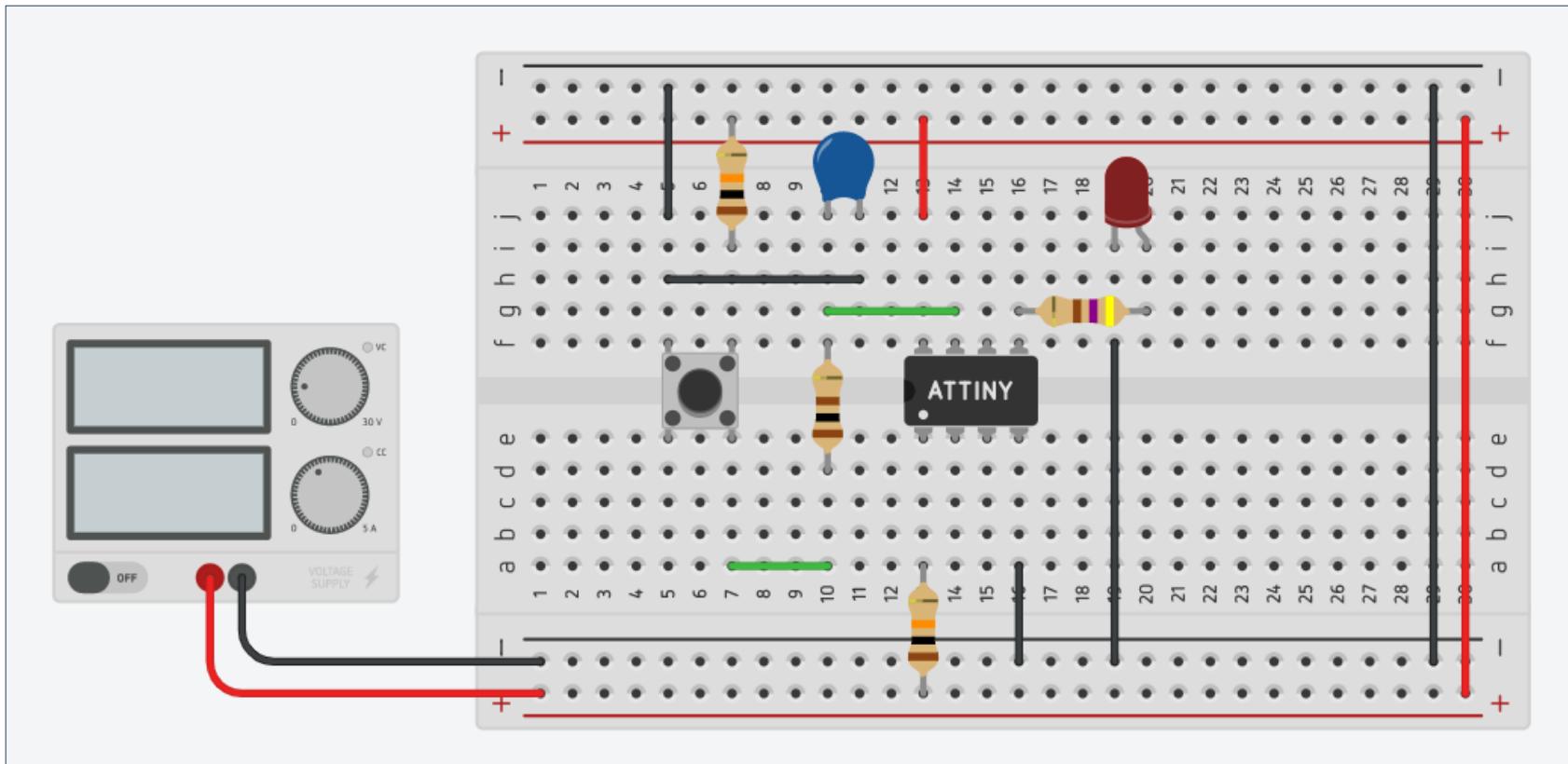
คำถาม: ถ้าทดลองต่อวงจรจริง และไม่ใส่ตัวต้านทาน Pull-up ที่ปุ่มกด การทำงานของวงจรนี้ จะมีพฤติกรรมที่แตกต่างจากเดิมหรือไม่ ?

ตัวอย่างการต่อวงจร LED ที่ขา PB0 เป็นเอาต์พุต มีวงจรปุ่มกดแบบ Active-Low ที่ขา PB2 เมื่อมีการกดปุ่มแล้วปล่อยในแต่ละครั้ง ให้สลับสถานะของเอาต์พุตหนึ่งครั้ง



# ตัวอย่างที่ 3: Push Button - LED

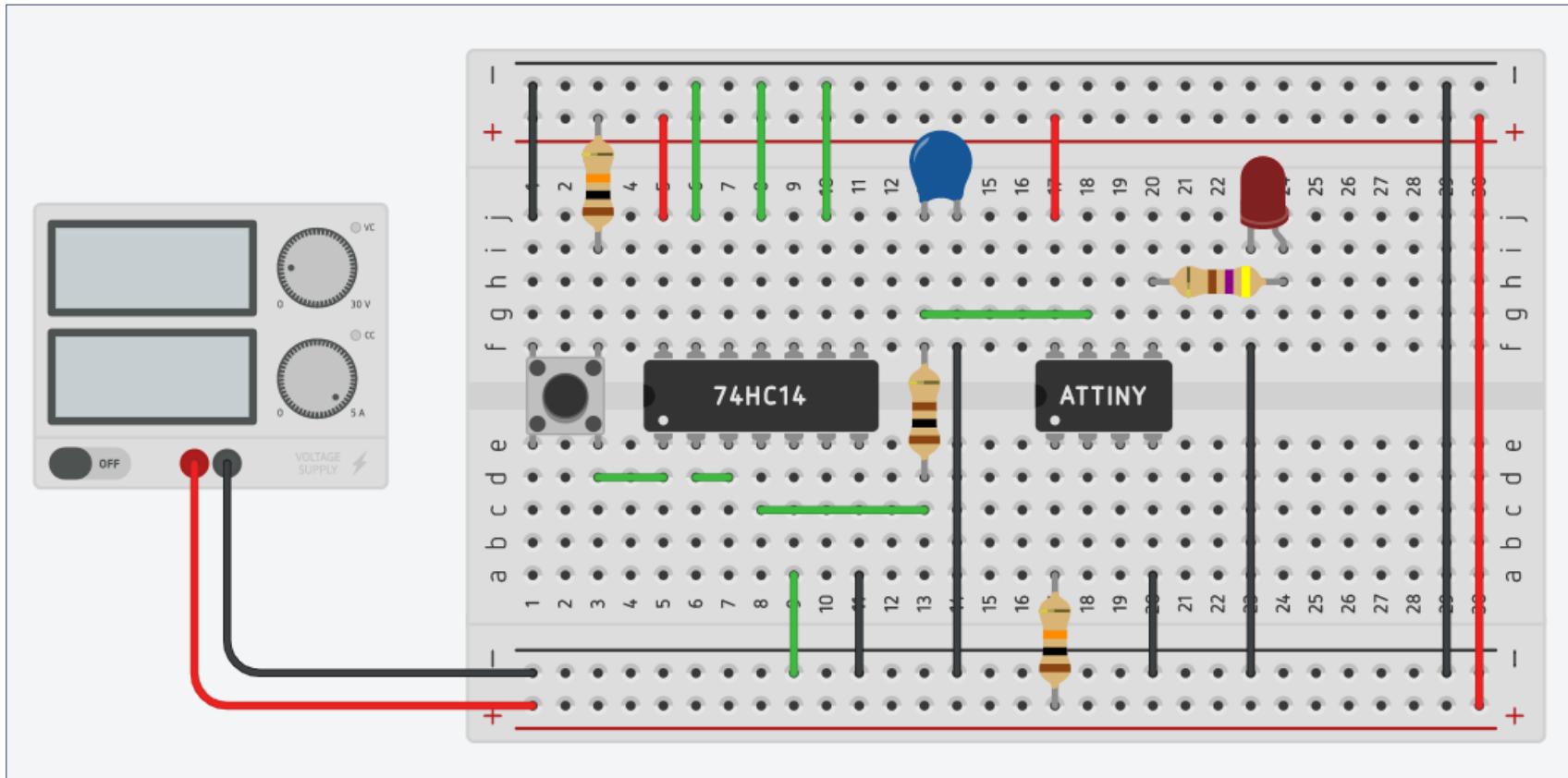
โจทย์ฝึกหัด: จงวาดผังวงจร (Schematic) สำหรับวงจรบนบอร์ดข้างล่างนี้



ตัวอย่างการต่อตัวเก็บประจุ (เช่น **100nF .. 1uF**) และตัวต้านทาน (เช่น **100 โอห์ม**)  
เพิ่มที่ขาสัญญาณของปุ่มกด เพื่อช่วยลดปัญหาการเกิด **Switch Bounce** (การกระเด้งของสัญญาณจากปุ่มกด)

# ตัวอย่างที่ 3: Push Button - LED

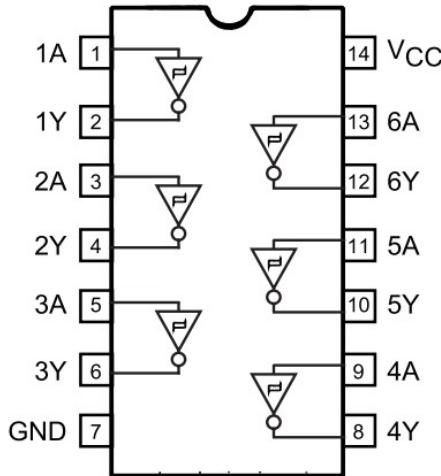
โจทย์ฝึกหัด: จงวาดผังวงจร (Schematic) สำหรับวงจรบนเบรดบอร์ดข้างล่างนี้



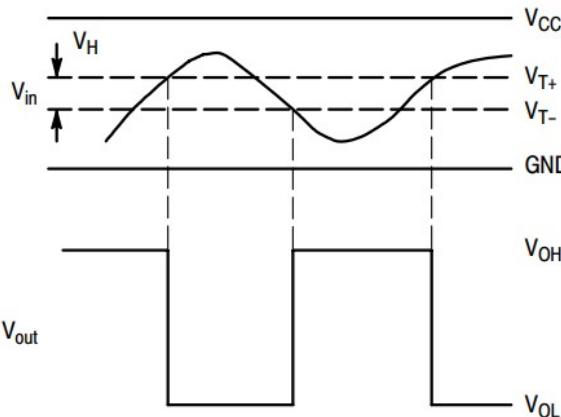
อีกตัวอย่างหนึ่งสำหรับเทคนิคในการลดปัญหาการเกิด **Switch Bounce** คือ การต่อตัวเก็บประจุ ตัวต้านทาน และไอซี **74HC14 (Schmitt-Trigger Inverters)**

# เอกสารอ้างอิงสำหรับไอซี 74HC14

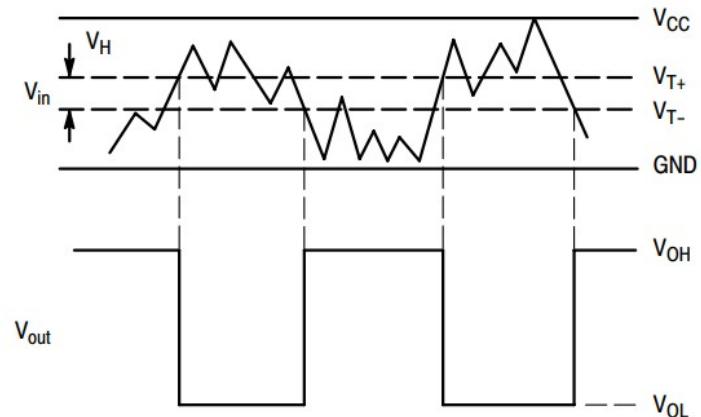
จุดยืดหยุ่น: จงอธิบายหลักการทำงานของไอซี 74HC14 โดยศึกษาจากเอกสาร **Datasheet** ของผู้ผลิต และอธิบายความหมายของคำว่า **Hysteresis, Threshold Voltages (V<sub>+</sub> และ V<sub>-</sub>)**



(a) A Schmitt-Trigger Squares Up Inputs With Slow Rise and Fall Times



(b) A Schmitt-Trigger Offers Maximum Noise Immunity



Reference: <https://www.st.com/resource/en/datasheet/m74hc14.pdf>

<http://www.mouser.com/ds/2/308/74HC14.REV1-34947.pdf>

# ตัวอย่างที่ 3: Push Button - LED

โจทย์ฝึกหัด: จงต่อวงจรทดลองและทดสอบการทำงานของโค้ดในตัวอย่างที่ 3 (1)

เขียนโค้ดเพื่อวนลูป  
ให้อ่านค่าจากอินพุต  
จากวงจรปุ่มกด (ขา  
**PB2**) และตรวจสอบ  
สถานะ

ปุ่มกดทำงานแบบ  
**Active-Low** ดังนั้น  
ในขณะที่กดปุ่ม จะ  
ได้ ค่าของอินพุต  
เป็น 0 แต่ถ้าไม่กด  
ปุ่ม จะได้ 1

ถ้ายังกดปุ่มค้างไว้  
ให้ร้อนกว่า จะ  
ปล่อยปุ่ม และจึง  
สลับสถานะของ  
เอาต์พุต (ขา **PB0**)

```
const int LED_PIN = PB0; // LED output pin
const int BTN_PIN = PB2; // Push button input pin

boolean state = false; // LED output state

void setup() {
    pinMode( LED_PIN, OUTPUT );
    digitalWrite( LED_PIN, state );
}

void loop() {
    if ( !digitalRead( BTN_PIN ) ){ // is the button pressed?
        // wait until the button released
        while ( !digitalRead( BTN_PIN ) ) {
            delay(10);
        }
        state = !state; // toggle the state
        digitalWrite( LED_PIN, state ); // update output
    }
}
```

การวนลูปซ้ำเพื่อ  
ค่อยอ่านค่าอินพุต  
แล้วทำขึ้นตอนตาม  
เงื่อนไข เป็นรูป  
แบบการทำงานที่  
เรียกว่า **Polling-based I/O**

# ตัวอย่างที่ 3: Push Button - LED Toggle

โจทย์ฝึกหัด: จงต่อวงจรทดลองและทดสอบการทำงานของโค้ดในตัวอย่างที่ 3 (2)

ตัวอย่างนี้สาธิตการเปิดใช้งาน “อินเทอร์รัพท์ภายนอก” (Ext. Interrupt) โดยใช้คำสั่ง `attachInterrupt()` หมายเลข 0 ซึ่งตรงกับขา PB2 ของ Attiny85

ไม่มีการวนลูป เพื่ออ่านค่าอินพุตที่ขา PB2 สำหรับปุ่มกด

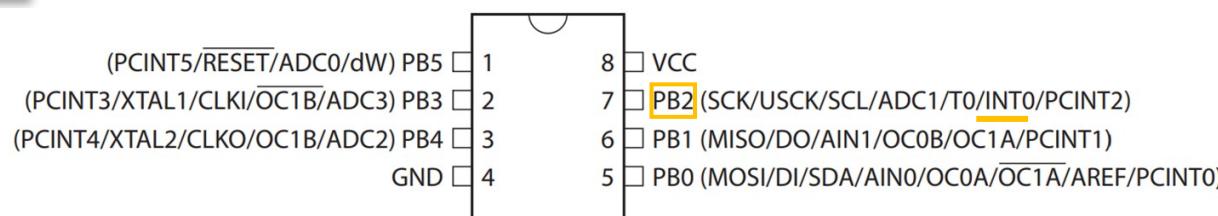
```
const int LED_PIN = PB0; // LED output pin
const int EXT_INT = 0; // use external interrupt: INT0

volatile boolean state = false; // LED output state

void callback() {
    state = !state; // toggle state
}
void setup() {
    pinMode( LED_PIN, OUTPUT );
    digitalWrite( LED_PIN, state );
    // enable external interrupt 0, falling edge
    attachInterrupt( EXT_INT, callback, FALLING );
}
void loop() {
    digitalWrite( LED_PIN, state ); // update state
}
```

ฟังก์ชัน `callback()` จะถูกเรียกเมื่อเกิดปุ่ม (โลจิกจะเปลี่ยนค่าจาก 1 เป็น 0 หรือเรียกว่า **Falling**) และถือว่า เกิดเหตุการณ์จากภายนอก

การทำงานรูปแบบนี้เรียกว่า **Interrupt-driven I/O**



# ตัวอย่างที่ 3: Push Button - LED

โจทย์ฝึกหัด: จงต่อวงจรทดลองและทดสอบการทำงานของโค้ดในตัวอย่างที่ 3 (3)

```
const int LED_PIN = PB0; // LED output pin
const int EXT_INT = 0; // use INT0
volatile boolean is_btn_pressed = false;
boolean state = false, blinking = false;
uint32_t ts; // timestamp (in msec)

void callback() {
    is_btn_pressed = true; // set flag
}

void setup() {
    pinMode( LED_PIN, OUTPUT );
    digitalWrite( LED_PIN, state );
    attachInterrupt( EXT_INT, callback, FALLING );
    ts = millis();
}
```

โค้ดนี้สาธิตการทำให้ **LED** กระพริบ และสามารถเปลี่ยนโหมดการกระพริบได้ เมื่อกดปุ่มหนึ่งครั้ง จะสลับโหมด (เปิดหรือปิด) การกระพริบ **LED**

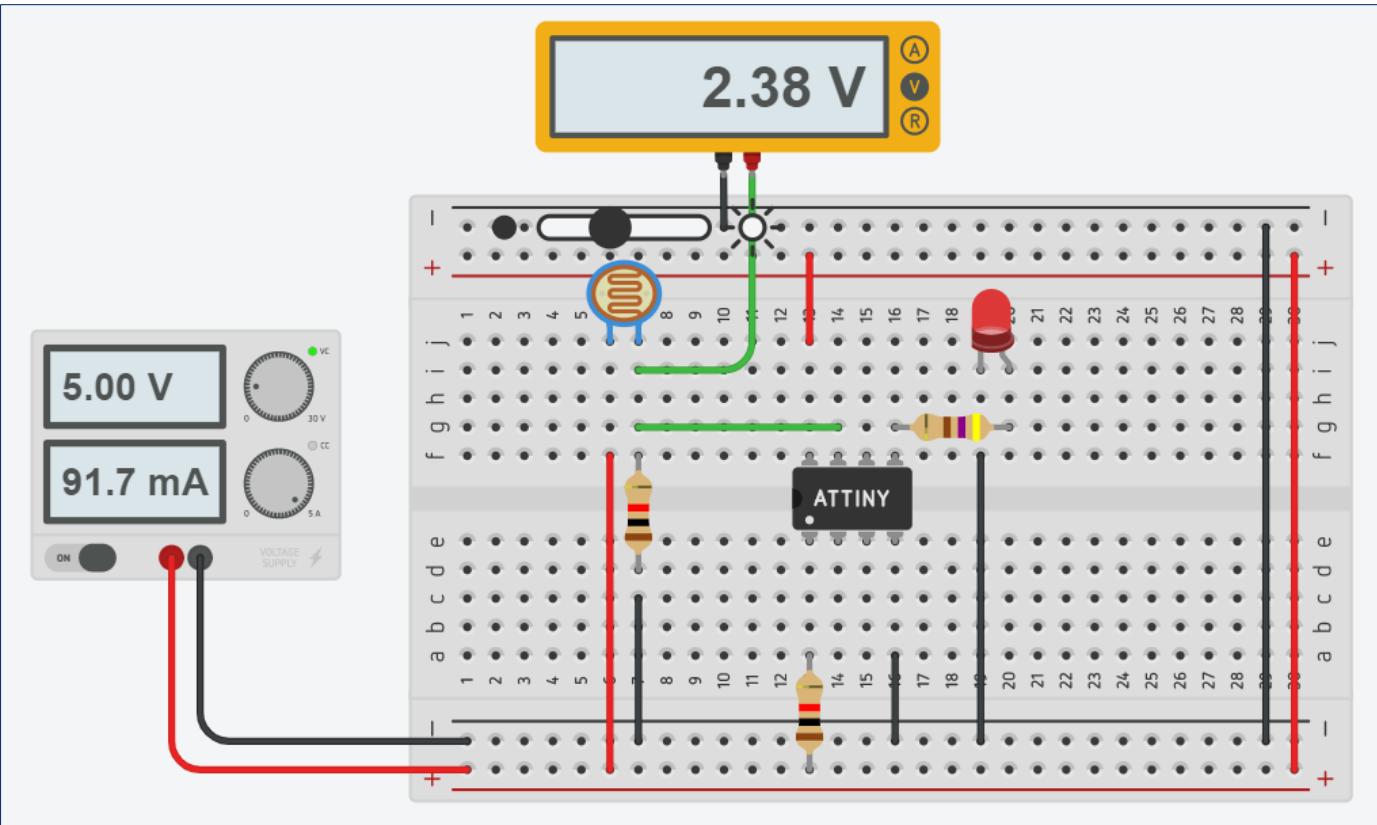
```
void loop() {
    if ( is_btn_pressed ) { // the button was pressed.
        is_btn_pressed = false; // clear flag
        blinking = !blinking; // toggle blink mode
    }
    if ( blinking ) { // LED blinking enabled
        if ( millis() - ts >= 100 ) {
            ts = millis();
            state = !state; // toggle LED state
        }
    } else { // LED blinking disabled
        state = false; // LED state = OFF
    }
    digitalWrite( LED_PIN, state ); // update output
}
```

คำสั่ง **attachInterrupt()** เปิดการใช้งานอินเทอร์รัพท์ภายนอก (ใช้อินเทอร์รัพท์หมายเลข 0 และตรงกับขา **PB2**) และเรียกฟังก์ชัน **callback()** ทุกครั้งที่เกิดเหตุการณ์ “ขอบขาลง” (**Falling Edge**) ที่ขา **PB2**

# ตัวอย่างที่ 4: LDR - LED

โจทย์ฝึกหัด: จงวาดผังวงจร (Schematic) สำหรับวงจรบนเบรดบอร์ดในตัวอย่างที่ 4

ตัวอย่างการต่อวงจร LED ที่ขา PB0 เป็นเอาต์พุต มีวงจร LDR (Photoresistor) และตัวต้านทานต่ออนุกรมทำหน้าที่เป็นเซ็นเซอร์แสงและใช้เป็นอินพุต-แอนะล็อกที่ขา PB2 (ADC1) เมื่อแสงน้อย (มืด) จะทำให้ LED สว่าง



# ตัวอย่างที่ 4: LDR - LED

โจทย์ฝึกหัด: จงต่อวงจรทดลองและทดสอบการทำงานของโค้ดในตัวอย่างที่ 4

โค้ดตัวอย่างนี้ใช้คำสั่ง  
`analogRead()` เพื่อ<sup>อ่านค่าอินพุตที่ขา</sup>  
**PB2 / A1**  
ซึ่งจะได้ค่าในช่วง  
**0..1023**

ถ้าใช้แรงดันไฟเลี้ยง  
**5V** หรือ **5000 mV**  
ก็สามารถคำนวณเพื่อ<sup>แปลงค่าอินพุตให้เป็น</sup>  
ค่าในหน่วยมิลลิโวลต์  
**(mV)** ได้

อ่านค่าอินพุตจากขา  
แอนะล็อกได้มากขึ้น  
ถ้าปริมาณแสงเพิ่มขึ้น

```
#include <inttypes.h>
const int LED_PIN = PB0; // LED output pin
const int AIN_PIN = A1; // analog input pin
const uint16_t level_low = 2500;
const uint16_t level_high = 2700;
boolean state = false;

void setup() {
    pinMode( LED_PIN, OUTPUT);
    digitalWrite( LED_PIN, state );
}

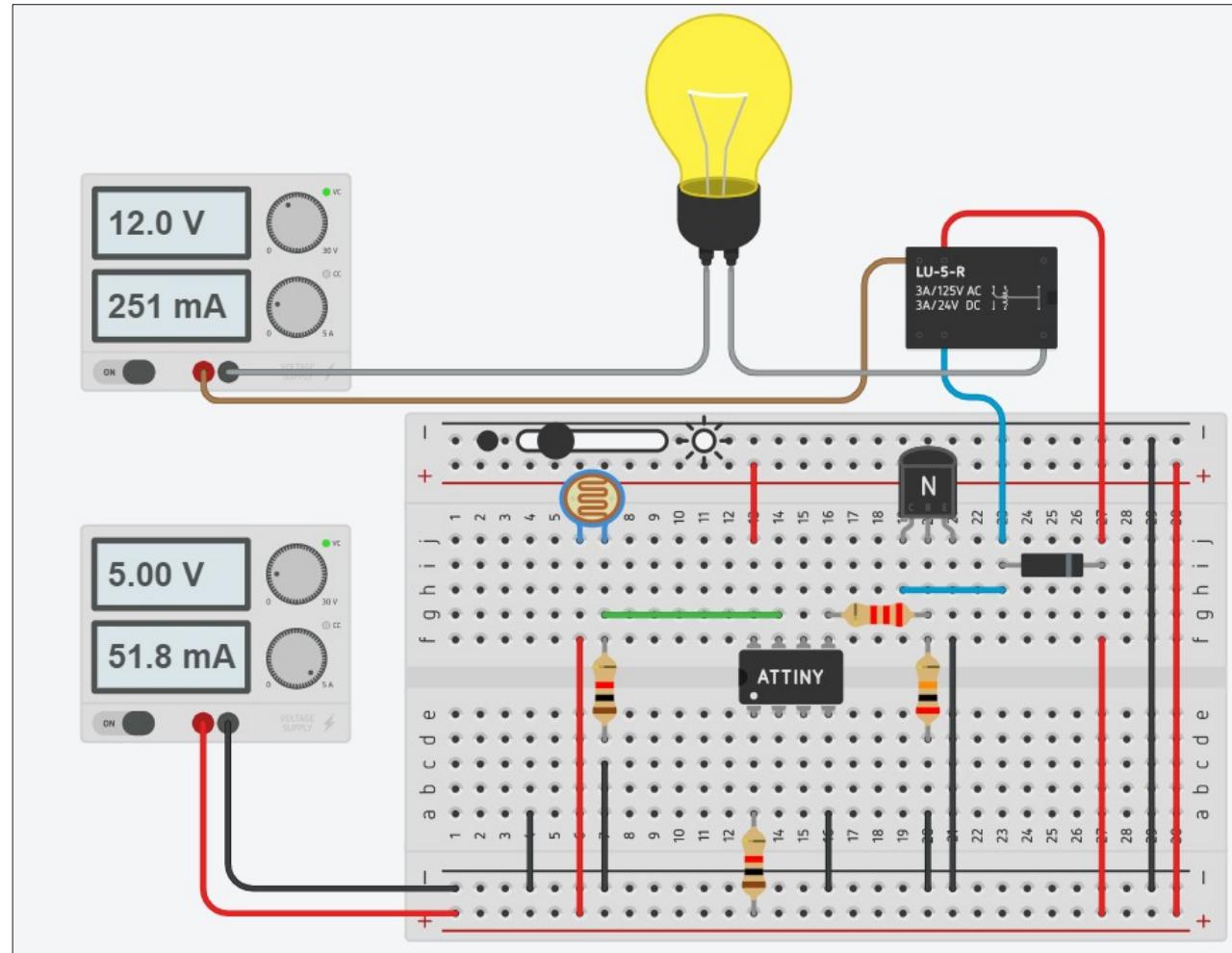
void loop() {
    uint16_t value = analogRead( AIN_PIN );
    uint16_t mV = (value * 5000UL) / 1024;
    if ( state && mV > level_high ) {
        state = false; // turn LED OFF
    }
    else if ( !state && mV < level_low ) {
        state = true; // turn LED ON
    }
    digitalWrite( LED_PIN, state);
}
```

**LED** จะเปลี่ยนสถานะเป็น **ON** เมื่อค่าอินพุตน้อยกว่า **level\_low** และเปลี่ยนสถานะเป็น **OFF** เมื่อค่าอินพุตเพิ่มขึ้นมากกว่า **level\_high**

# ตัวอย่างที่ 5: LDR – Relay – Light Bulb

โจทย์ฝึกหัด: จงวัดผังวงจรสำหรับวงจรบนบอร์ดในตัวอย่างที่ 5

ตัวอย่างการต่อวงจรเปิด-ปิดหลอดไฟแสงสว่าง โดยอัตโนมัติ เมื่อแสงน้อย ด้วยรีเลย์ (โมเดล LU-5-R) ซึ่งใช้แรงดัน 5V สำหรับค oy l (Coil Voltage) ใช้ทรานซิสเตอร์ (NPN) ควบคุมการทำงานของรีเลย์



# ຕ້າຍ່າງຂ້ອມູລເຊີງເທັນນິກສໍາຮັບຮືອງ LU-5-R

## CONTACT RATING

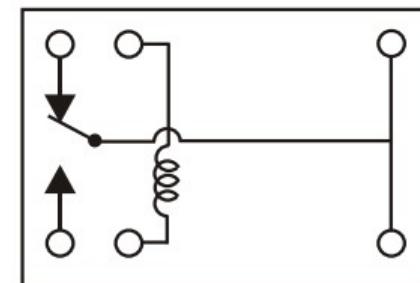
1 Form C (1PDT)	AC 120V	1A
	DC 24V	2A

## COIL DATA(0.2W~0.36W, at 25°C )

Coil Nominal Voltage (VDC)	Resistance Tol.±10% (Ohms )	Nominal Current (mA)	Maximum Pick Up Voltage (V)	Minimum Drop Out Voltage (V)
3	25	120.0	2.25	0.3
5	125	40.0	3.75	0.5
6	180	33.3	4.5	0.6
9	405	22.2	6.75	0.9
12	720	16.7	9.0	1.2
24	2,880	8.3	18.0	2.4

BOTTOM VIEW

N.C.



COMMON

N.O.

N.C. = Normally Closed

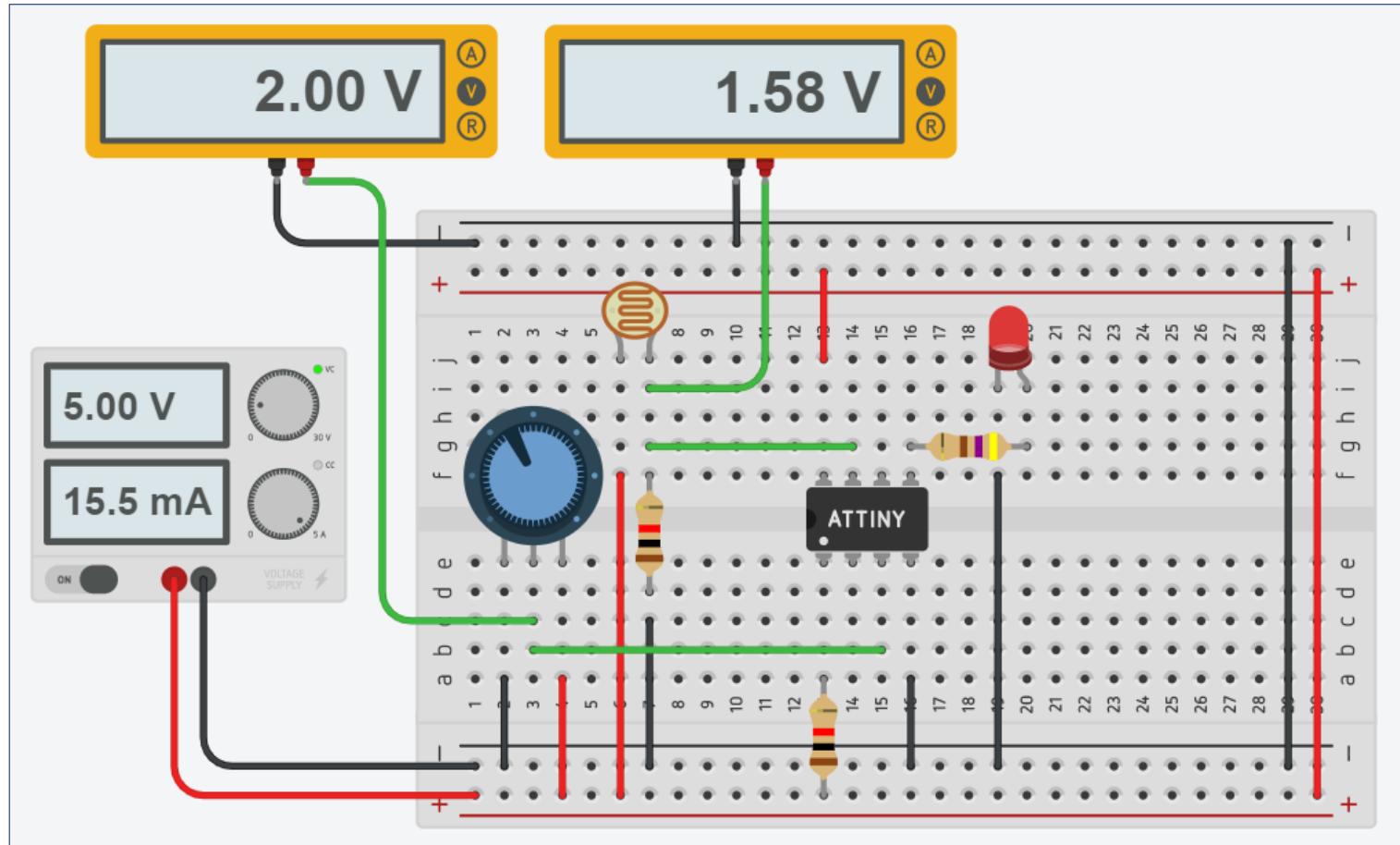
N.O. = Normally Open

Reference: <https://datasheet.octopart.com/LU-5-R-Rayex-datasheet-10584258.pdf>

# ตัวอย่างที่ 6: Voltage Level Comparator

โจทย์ฝึกหัด: จงวัดผังวงจรสำหรับวงจรบันเบรอร์ดในตัวอย่างที่ 6

ตัวอย่างการอ่านค่าอินพุต-แอนะล็อก 2 ช่อง (**A** และ **B**) นำมาเปรียบเทียบกันแล้วกำหนดสถานะของเอาต์พุต (**LED**)



# ตัวอย่างที่ 6: Voltage Level Comparator

โจทย์ฝึกหัด: จงต่อวงจรทดลองและทดสอบการทำงานของโค้ดในตัวอย่างที่ 6

คำสั่ง `analogRead()` ใช้สำหรับอ่านค่าแรงดันอินพุต ที่ขา **ADC1 (A1)** และ **ADC2 (A2)** และนำมาเปรียบเทียบกัน ในตัวอย่างนี้ ช่อง **A** หมายถึงขา **A1** ได้แรงดันอินพุตจากวงจร **LDR** และช่อง **B** คือขา **A2** ได้แรงดันอินพุตจากวงจรตัวต้านทานปรับค่าได้ ถ้า **A < B** จะทำให้ **LED** สว่าง

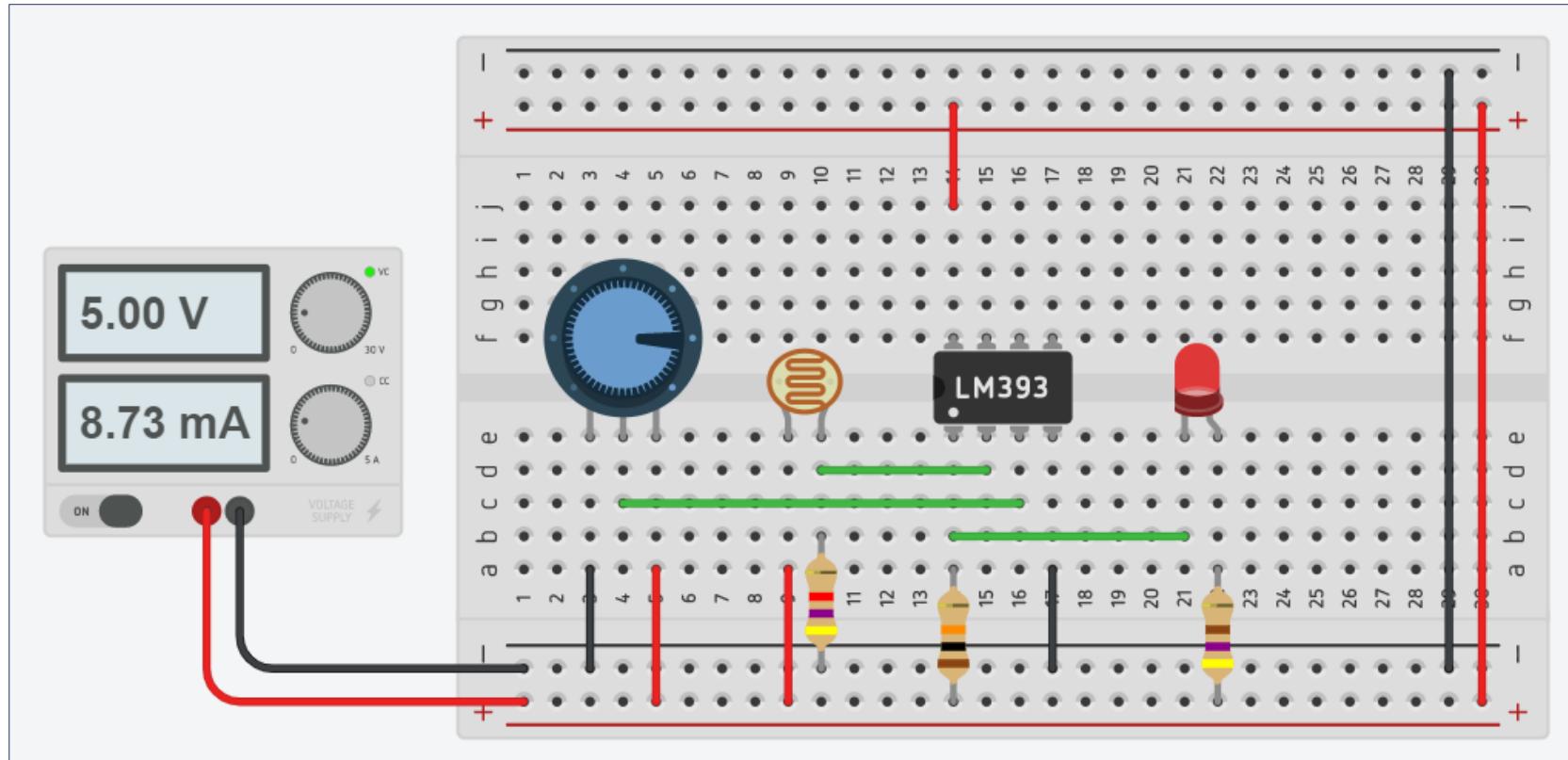
```
const int LED_PIN = PB0; // LED output pin
const int A_PIN = A1; // analog input A, use ADC1 pin
const int B_PIN = A2; // analog input B, use ADC2 pin

void setup() {
    pinMode( LED_PIN, OUTPUT );
    digitalWrite( LED_PIN, LOW );
}

void loop() {
    int a = analogRead( A_PIN ); // read input A
    int b = analogRead( B_PIN ); // read input B
    digitalWrite( LED_PIN, (a<b) ); // update output
    delay(100);
}
```

# การใช้ไอซีเปรียบเทียบแรงดัน

โจทย์ฝึกหัด: จงวาดผังวงจรสำหรับวงจรบันเบรอร์ดตามรูปตัวอย่างข้างล่างนี้

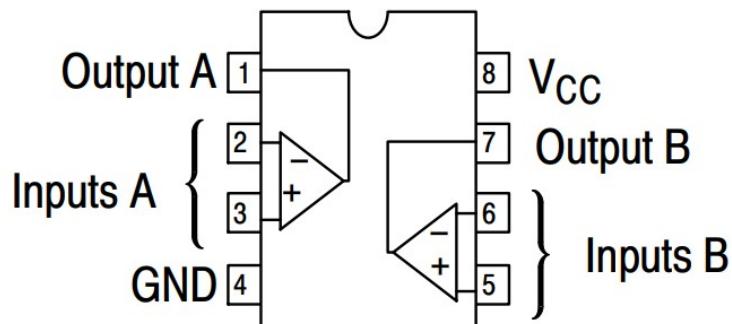


ในตัวอย่างที่ 6 เป็นการใช้ ATTiny85 และเขียนโค้ดให้หน้าที่อ่านค่าแรงดันแอนะล็อก 2 ช่อง และเปรียบเทียบกัน เพื่อกำหนดสถานะเอาต์พุตให้ LED ตัวอย่างนี้สาธิตการใช้ ไอซี LM393 ซึ่งเป็นไอซีเปรียบเทียบแรงดัน (Voltage Comparator) แทนการใช้ ATTiny85

# ตำแหน่งขาของ LM393

ไอซี LM393 มีวงจรเปรียบเทียบแรงดันที่เป็นสัญญาณแอนะล็อกอยู่ภายใน 2 ชุด (**Dual Voltage Comparator**) ทำงานได้อิสระจากกัน วงจรเปรียบเทียบแต่ละชุด มีขาอินพุต + () และ - และขาเอาต์พุต ถ้าแรงดันที่ขา + สูงกว่าที่ขา - จะให้เอาต์พุตเป็น 1 แต่ถ้าต่ำกว่าจะได้เป็น 0

## PIN CONNECTIONS

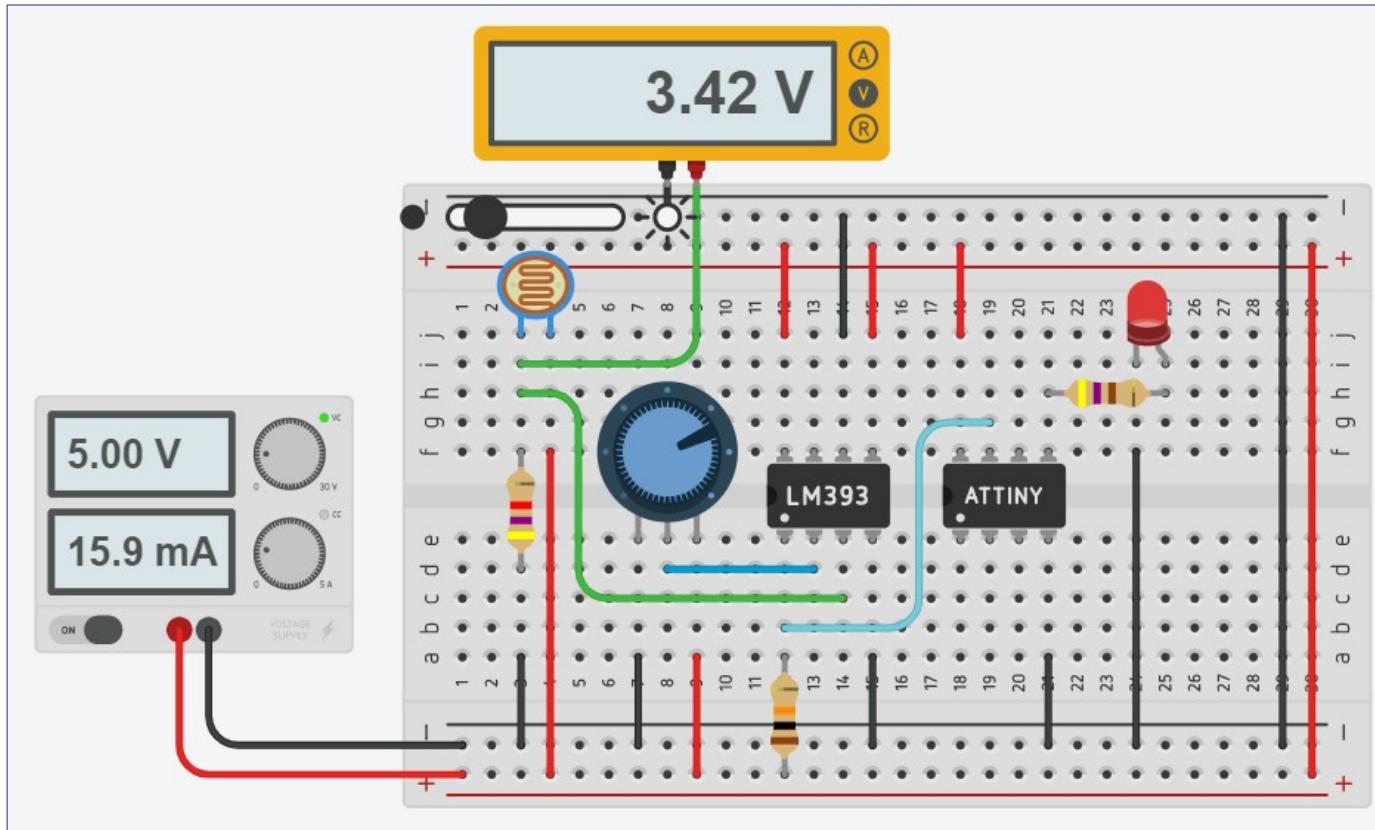


(Top View)

Reference: <https://www.onsemi.com/PowerSolutions/document/LM393-D.PDF>

# ตัวอย่างที่ 7: LDR – LM393 – LED

โจทย์ฝึกหัด: จงวัดผังวงจรสำหรับวงจรบันเบรอร์ดในตัวอย่างที่ 7



ตัวอย่างสาธิตการต่อวงจรโดยใช้ไอซี LM393 เป็นตัวเปรียบเทียบแรงดัน และตรวจสอบการเปลี่ยนแปลงปริมาณแสงด้วย LDR และใช้ตัวต้านทานปรับค่าได้ในการตั้งค่าเปรียบเทียบ ถ้าแสงน้อย จะทำให้ LED สว่าง

# ตัวอย่างที่ 7: LDR – LM393 – LED

โจทย์ฝึกหัด: จงต่อวงจรทดลองและทดสอบการทำงานของโค้ดในตัวอย่างที่ 7

```
const int LED_PIN = PB0; // LED output pin
const int TRIG_PIN = PB2; // digital input pin
const int EXT_INT = 0; // use INT0 / PB2 pin

volatile boolean flag = false;
int value; // input value (0 or 1)

void callback() {
    flag = true; // set flag
}

void setup() {
    pinMode( LED_PIN, OUTPUT );
    digitalWrite( LED_PIN, false );
    attachInterrupt( EXT_INT, callback, CHANGE );
    value = digitalRead( TRIG_PIN );
    digitalWrite( LED_PIN, !value );
}
```

```
void loop() {
    if ( flag ) {
        flag = false; // clear flag
        value = digitalRead( TRIG_PIN );
        digitalWrite( LED_PIN, !value );
    }
    delay(100);
}
```

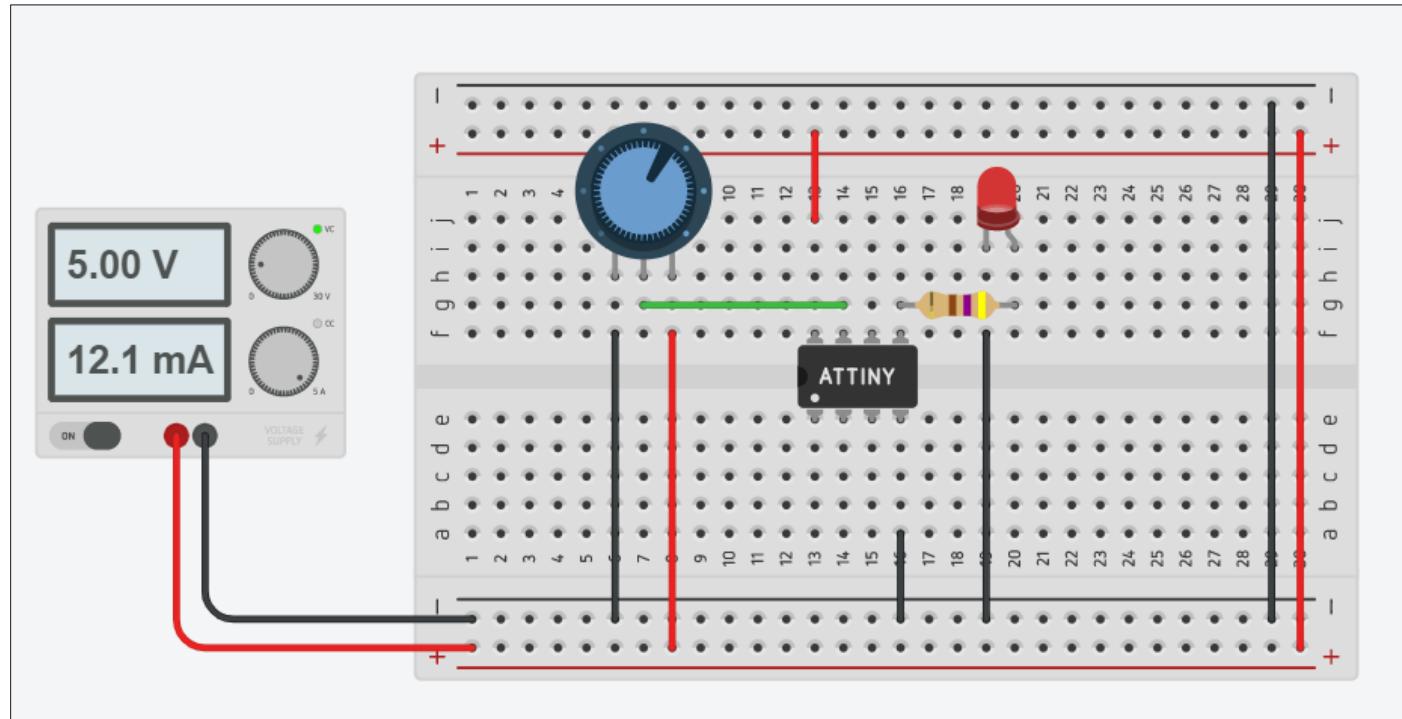
สัญญาณเอาต์พุตจาก **LM393** จะถูกใช้เป็นสัญญาณอินพุต-ดิจิทัล ที่ขา **PB2** ซึ่งตรงกับอินเทอร์รูปท์หมายเลข **INT0**

เมื่อมีการเปลี่ยนแปลงค่าของอินพุตที่ **PB2** จะมีการเรียกฟังก์ชัน **callback()** และตัวแปร **flag** จะเป็น **true** เพื่อบ่งบอกว่า มีการเปลี่ยนแปลงที่อินพุต และจะต้องทำการอัปเดตเอาต์พุต

# ตัวอย่างที่ 8: Trimpot - LED Dimming

โจทย์ฝึกหัด: จงวัดผังวงจรสำหรับวงจรบนเบรดบอร์ดในตัวอย่างที่ 8

ตัวอย่างการใช้ตัว  
ต้านทานปรับค่าได้  
**(Potentiometer หรือ  
Trimpot)** เช่น 10k  
เพื่อปรับแรงดันอินพุต-  
แอนะล็อก และใช้กำหนด  
ระดับความสว่างของ  
LED



Trimpot = Trimmer Potentiometer

# ตัวอย่างที่ 8: Trimpot - LED Dimming

โจทย์ฝึกหัด: จงต่อวงจรทดลองและทดสอบการทำงานของโค้ดในตัวอย่างที่ 8

คำสั่ง `analogWrite()` ใช้สำหรับสร้างสัญญาณเอาต์พุตแบบ **PWM (Pulse Width Modulation)** ซึ่งเป็นสัญญาณแบบมีความแต่ปรับช่วงกว้างที่เป็นลอจิก 1 (**High**) เทียบกับค่าของสัญญาณโดยคิดเป็นเปอร์เซ็นต์: **0=0%** ถึง **255=100%**

คำสั่ง `analogRead()` อ่านได้ค่าในช่วง **0..1023** ถ้าจะนำค่าที่ได้ไปใช้กับคำสั่งจะต้องหารด้วย **4** เพื่อให้ค่าอยู่ในช่วง **0..255**

```
const int LED_PIN = PB0; // digital input pin PB0
const int AIN_PIN = A1; // analog input pin A1 (ADC1)

int value = 0; // 0..255

void setup() {
    analogWrite( LED_PIN, value ); // set output to 0
}

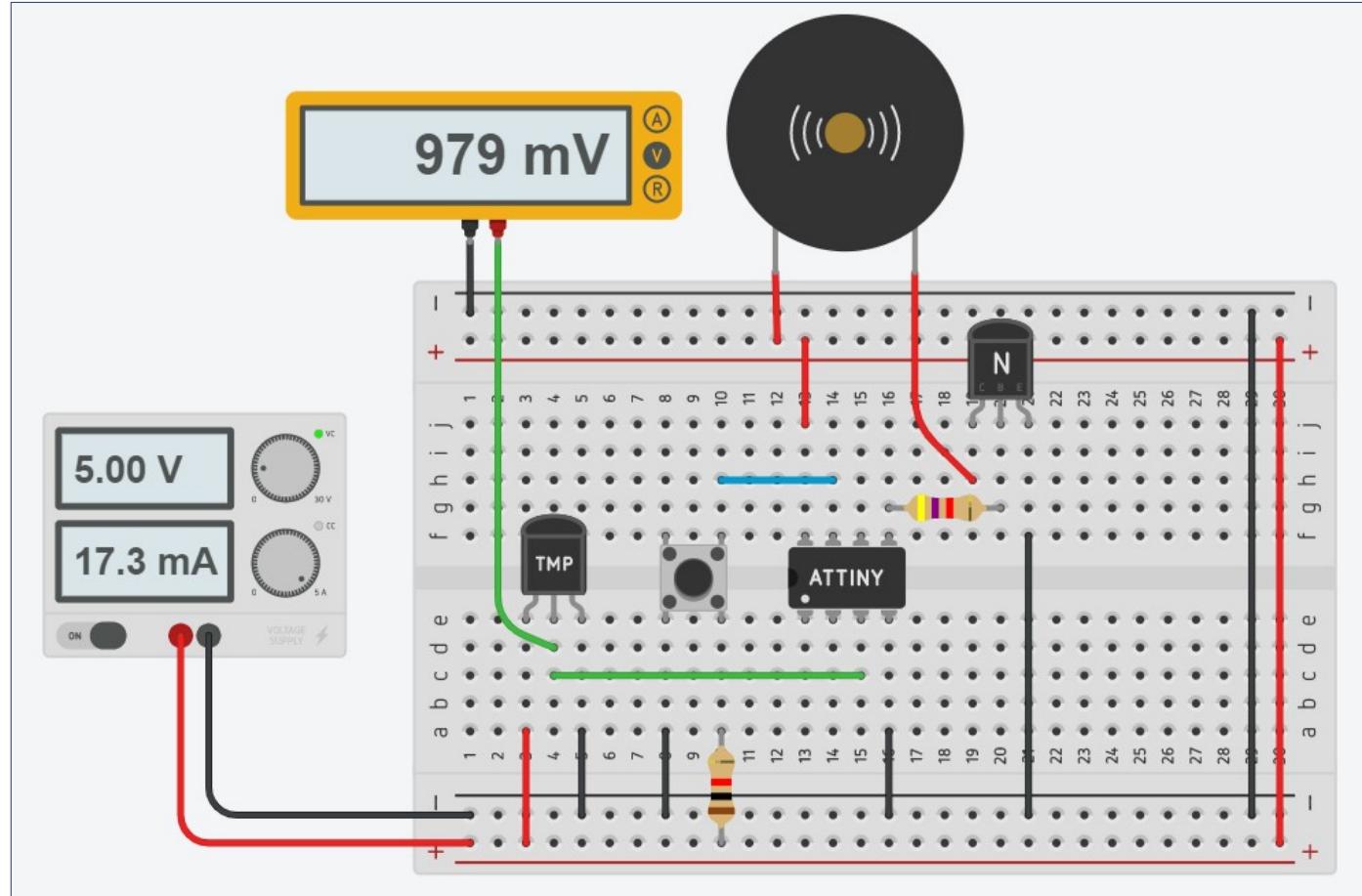
void loop() {
    int new_value = analogRead( AIN_PIN )/4;
    if ( new_value != value ) { // input value changed ?
        value = new_value; // update value
        analogWrite( LED_PIN, value ); // update PWM
        delay(100);
    }
}
```

# ตัวอย่างที่ 9: TMP36 - Buzzer

โจทย์ฝึกหัด: จงวัดผังวงจรสำหรับวงจรบนเบรดบอร์ดในตัวอย่างที่ 9

ตัวอย่างการใช้ไอซี  
**TMP36 (Analog Temperature Sensor IC)** สำหรับ  
วัดอุณหภูมิในช่วง  
**-40°C .. 125°C** และ  
ให้อาตโนมัติเป็นแรงดัน  
ไฟฟ้า **0V @ -50°C**  
**ถึง 1.75V @ -125°C**  
(สเกลแบบเชิงเส้น)

วงจรใช้ทรานซิสเตอร์  
แบบ **NPN** เปิด-ปิด  
การทำงานของบีซ  
เซอร์เสียง



# ตัวอย่างที่ 9: TMP36 - Buzzer

โจทย์ฝึกหัด: จงต่อวงจรทดลองและทดสอบการทำงานของโค้ดในตัวอย่างที่ 9

```
const int BUZ_PIN = PB0; // digital output pin (buzzer)
const int TMP_PIN = A2; // analog input pin (TMP36)
const int EXT_INT = 0; // use INT0
const int LEVEL = 40; // compare level (deg.C)

volatile boolean enable = true;

void callback() {
    enable = !enable; // enable/disable buzzer alarm
}

void setup() {
    pinMode( BUZ_PIN, OUTPUT );
    attachInterrupt( EXT_INT, callback, FALLING );
}
```

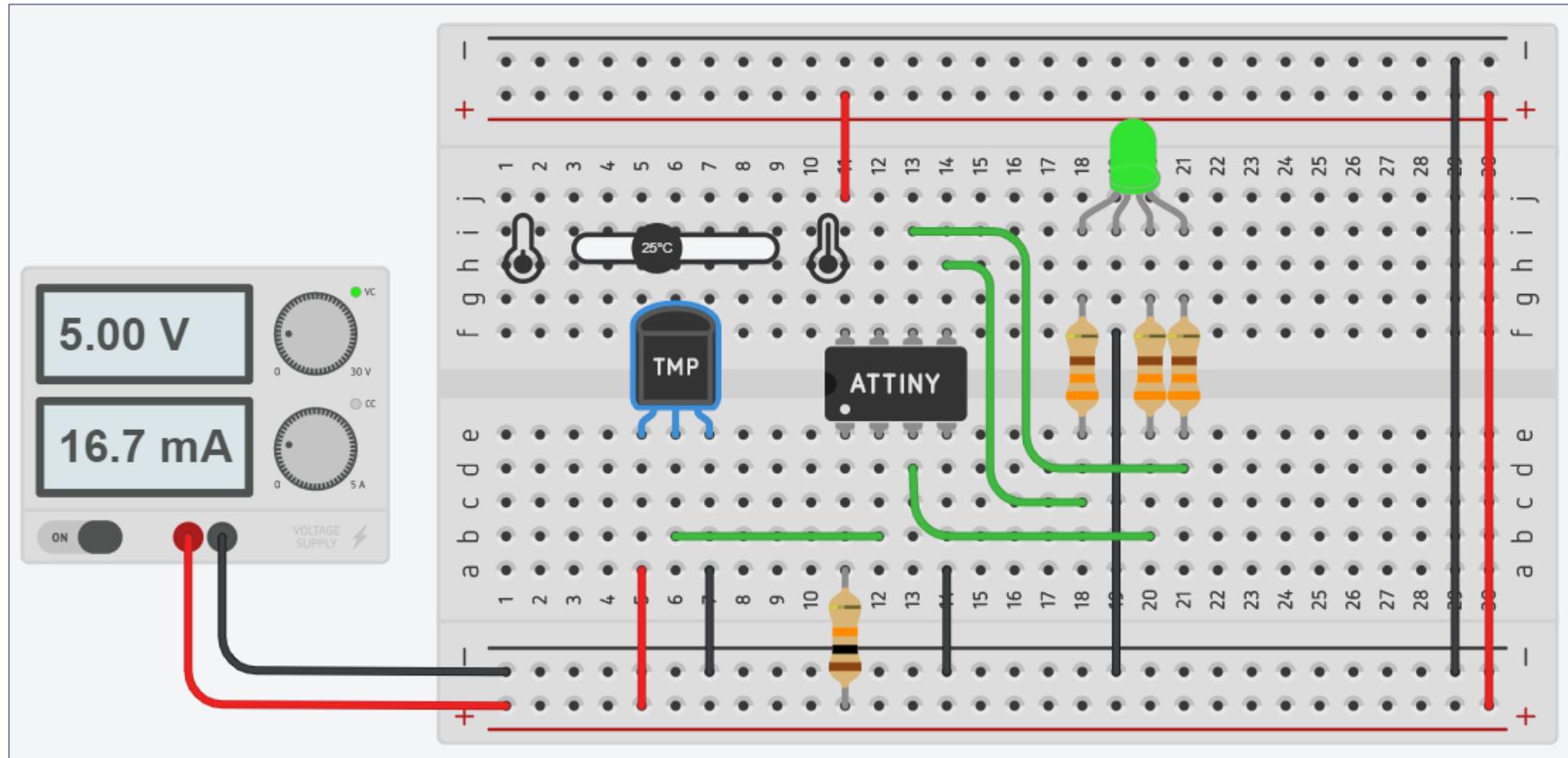
```
// TMP36: 0V @ -50°C, 1.75V @ 125°C
// Temp [deg°C] = (voltage [mV] - 500)/10

void loop() {
    int value = analogRead( TMP_PIN );
    int mV = (value * 5000L) / 1024;
    int temp = (mV - 500) / 10;
    int output = enable && (temp >= LEVEL);
    digitalWrite( BUZ_PIN, output );
}
```

ในตัวอย่างนี้ ถ้าค่าอุณหภูมิเกิน 40 องศา จะทำให้บีซเซอร์เสียง (Sound Buzzer) มีเสียงดังเพื่อเป็นการแจ้งเตือน และสามารถกดปุ่มเพื่อเปิดหรือปิดการแจ้งเตือนด้วยเสียงได้

# ตัวอย่างที่ 10: TMP36 – RGB LED

โจทย์ฝึกหัด: จงวาดผังวงจรสำหรับวงจรบนบอร์ดในตัวอย่างที่ 10



ในตัวอย่างนี้ สาธิตการวัดอุณหภูมิด้วยไอซี **TMP36** และใช้ **RGB LED (Common Cathode)** แสดงช่วงของค่าที่วัดได้ เช่น สีน้ำเงินสำหรับค่าที่ต่ำกว่า 20 องศา หรือ สีแดง สำหรับค่าที่สูงกว่า 40 องศา

# ตัวอย่างที่ 10: TMP36 – RGB LED

โจทย์ฝึกหัด: จงต่อวงจรทดลองและทดสอบการทำงานของโค้ดในตัวอย่างที่ 10

```
const int RGB_PINS[] = {PB0, PB1, PB4};  
const int TMP36_PIN = A3;  
  
typedef union _RGB {  
    struct { byte r,g,b; } s;  
    byte v[3];  
} RGB;  
  
RGB rgb;  
  
void setup() {  
    for ( int i=0; i < 3; i++ ) {  
        pinMode( RGB_PINS[i], OUTPUT );  
        digitalWrite( RGB_PINS[i], 0 );  
    }  
}
```

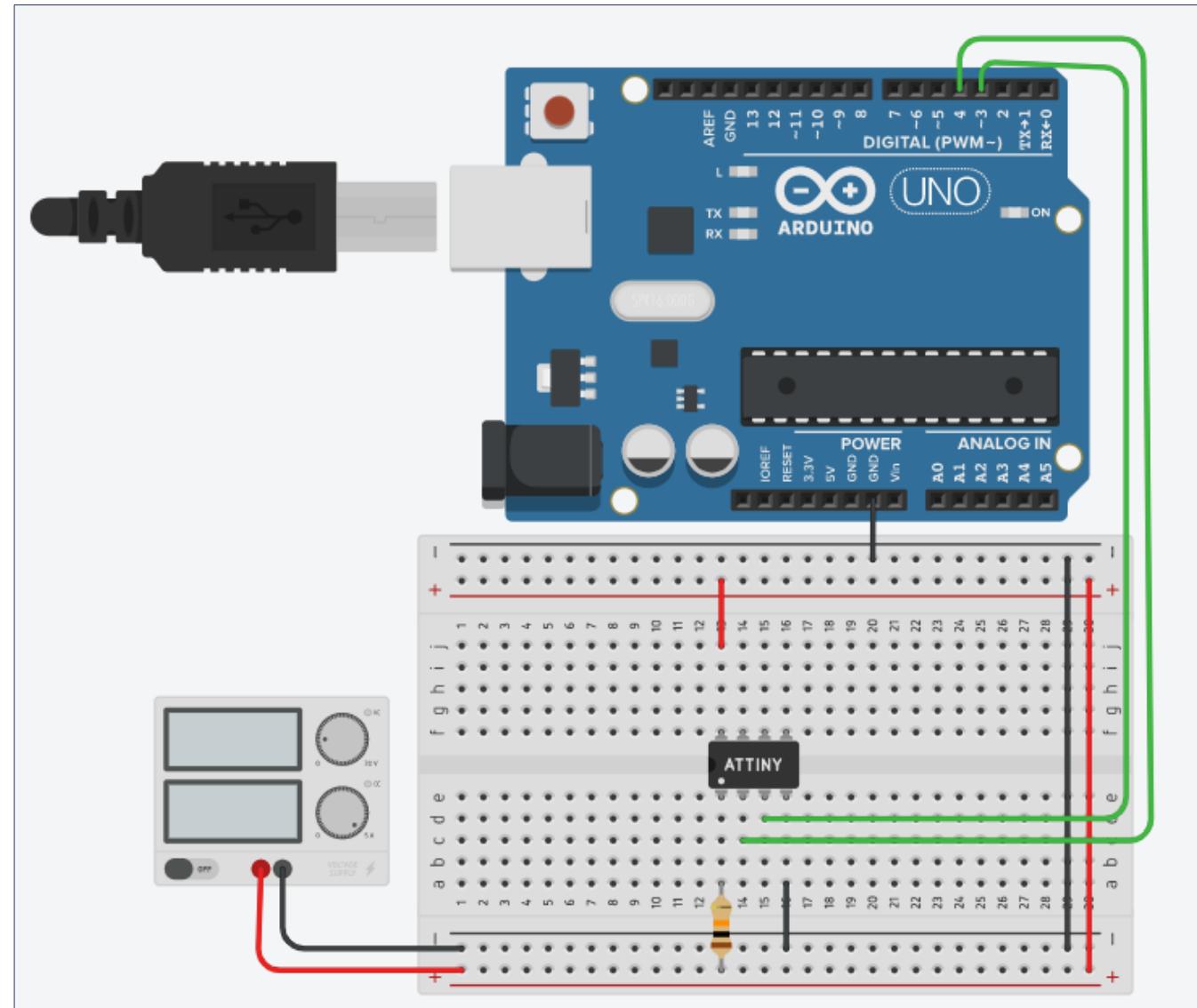
```
void loop() {  
    // read analog input from TMP36 sensor  
    int value = analogRead( TMP36_PIN );  
    int mV = (value * 5000L) / 1024;  
    int temp = (mV - 500) / 10; // value in deg.C  
    memset( rgb.v, 0x00, 3 );  
    if (temp < 10) {           // below 10 deg.C  
        rgb.s.b = 1;          // blue color  
    } else if (temp < 20) {    // 10..20 deg.C  
        rgb.s.b = 1; rgb.s.g = 1;  
    } else if (temp < 30) {    // 20..30 deg.C  
        rgb.s.g = 1;          // green color  
    } else if (temp < 40) {    // 30..40 deg.C  
        rgb.s.g = 1; rgb.s.r = 1;  
    } else {                  // 40+ deg.C  
        rgb.s.r = 1;          // red color  
    }  
    for ( int i=0; i < 3; i++ ) {  
        digitalWrite( RGB_PINS[i], rgb.v[i] );  
    }  
}
```

# ตัวอย่างที่ 11: Uno - Attiny85 Serial Link

ตัวอย่างสาธิตการเชื่อมต่อเพื่อสื่อสารข้อมูลแบบบิตอนุกรม (Serial) ระหว่างบอร์ด Arduino Uno และ ATTiny85 บนเบรเดบอร์ด

ขา 3 และ 4 ใช้สำหรับขา Rx และ Tx ของ SoftSerial ตามลำดับ

การต่อสายระหว่างขา Rx และ Tx ของทั้งสองบอร์ด จะต้องไข้สายกัน (Rx → Tx และ Tx <- Rx)



# ตัวอย่างที่ 11: Uno - Attiny85 Serial Link

TIN KER CAD uno\_attiny85\_serial\_link Simulator time: 00:00:01.773 All changes saved Code Stop Simulation Export Share 2 (Arduino Uno R3)

The circuit diagram illustrates a serial communication setup between an Arduino Uno and an ATTINY85 microcontroller. The Arduino Uno is connected to the breadboard via its pins 3 (RX) and 4 (TX). The ATTINY85 is also connected to the breadboard. A 5V power source is connected to the Arduino's 5V pin and the ATTINY85's Vcc pin. A ground connection is made from the Arduino's GND pin to the breadboard and the ATTINY85's GND pin. The ATTINY85 is programmed to count and send the values to the Arduino Uno.

```
#include <SoftwareSerial.h>
SoftwareSerial mySerial( 3, 4 ); //rx, tx
#define BUF_LEN 64
char buf[ BUF_LEN+1 ];
int index = 0;

void setup() {
  Serial.begin(115200);
  mySerial.begin(4800);
}

void loop() {
  while (mySerial.available() > 0) {
    if ( index < BUF_LEN ) {
      buf[index++] = mySerial.read();
    } else break;
  }
  if (index > 0) {
    buf[index] = '\0';
    index = 0;
    Serial.print(buf);
  }
}
```

cnt: 001  
cnt: 002  
cnt: 003  
cnt: 004  
cnt: 005  
cnt: 006  
cnt: 007  
cnt: 008

Send Clear

# ตัวอย่างที่ 11: Uno – Attiny85 Serial Link

```
#include <SoftwareSerial.h>
SoftwareSerial mySerial( 3, 4 ); //rx, tx

void setup() {
    mySerial.begin(4800);
    mySerial.println( "ATTiny85..." );
}

char sbuf[16]; // char buffer

void loop() {
    static byte cnt = 0;
    sprintf( sbuf, "cnt: %03d", ++cnt );
    mySerial.println( sbuf );
    delay(200);
}
```

Attiny85

Uno ทำหน้าที่ค่อยรับข้อมูลจาก Attiny85 ด้วยวิธี **SoftSerial** (ตั้งค่า **baud 4800**) และเมื่อได้รับข้อความ ให้ จะส่งต่อออกทาง **Serial (Hardware)** และเปิดดูข้อมูลได้ใน **Serial Monitor**

```
#include <SoftwareSerial.h>
SoftwareSerial mySerial( 3, 4 ); //rx, tx

#define BUF_LEN 64
char buf[ BUF_LEN+1 ];
int index = 0;

void setup() {
    Serial.begin(115200);
    mySerial.begin(4800);
}

void loop() {
    while (mySerial.available() > 0) {
        if ( index < BUF_LEN ) {
            buf[index++] = mySerial.read();
        } else break;
    }
    if (index > 0) {
        buf[index] = '\0';
        index = 0;
        Serial.print(buf);
    }
}
```

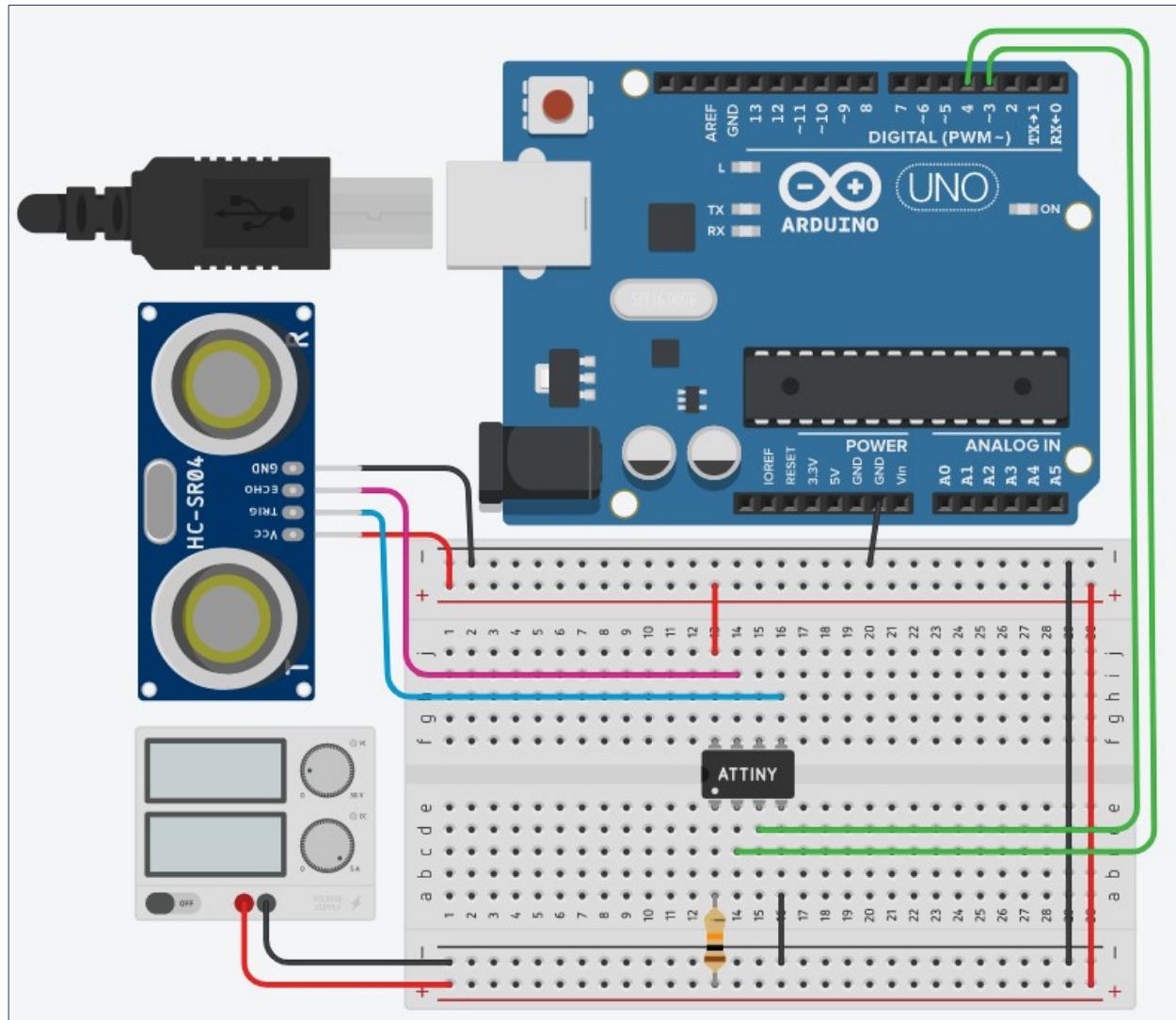
Uno

# ตัวอย่างที่ 12: Ultrasonic Distance Sensor

**ໂຈທຍື່ັກໜັດ: ໃຫ້ສຶກຂາກເກຮົາການທຳງານຂອງໂມດູລ **HC-SR04** ຈາກເອກສາຣູຜູ້ຜລິດ**

ตัวอย่างนี้สาธิตการอ่านค่า  
ระยะห่างจากวัตถุด้วยโมดูล  
**HC-SR04 (Ultrasonic  
Distance Sensor)** ซึ่งมี  
ขา **TRIGGER** และ **ECHO**

ในการอ่านค่าจาก **HC-SR04** แต่ละครั้ง จะต้องสร้างสัญญาณพัลส์ (**Pulse**) เป็นสัญญาณกระแสเพื่อเริ่มต้นการวัดด้วยคลื่นเสียง ความถี่สูง และเอาต์พุตที่ได้จะเป็นสัญญาณพัลส์ที่ขา **ECHO** ซึ่งความกว้างของพัลส์เป็นระยะเวลาเดินทางของเสียงเดินทางไปและสะท้อนกลับมา (หน่วย: ไมโครวินาที)



# ตัวอย่างที่ 12: Ultrasonic Distance Sensor

ข้อสังเกต: บอร์ด **Uno** ทำหน้าที่รับข้อมูลจาก **Attiny85** ที่วัดระยะห่างจากวัตถุด้วย **HC-SR04** และส่งต่อข้อมูลไปยัง **Serial Monitor** และค่าที่ได้มีหน่วยเป็นเซนติเมตร (cm.)

TIN KER CAD uno\_attiny85\_serial\_ultrasonic All changes saved

Simulator time: 00:00:10.112

Code Stop Simulation Export Share

Ultrasonic Distance Sensor Name 2

Arduino Uno R3

Text

```
1 #include <SoftwareSerial.h>
2 SoftwareSerial mySerial( 3, 4 ); //rx, tx
3
4 #define BUF_LEN 64
5 char buf[ BUF_LEN+1 ];
6 int index = 0;
7
8 void setup() {
9   Serial.begin(115200);
10  mySerial.begin(4800);
11 }
12 void loop() {
13   while (mySerial.available() > 0) {
14     if ( index < BUF_LEN ) {
15       buf[index++] = mySerial.read();
16     } else break;
17   }
18   if (index > 0) {
19     buf[index] = '\0';
20     index = 0;
21     Serial.print(buf);
22   }
23 }
```

Serial Monitor

40  
40  
40  
40  
40  
40  
39

The circuit diagram shows an Attiny85 microcontroller connected to an Arduino Uno. The Attiny85 is powered by a 5.00 V power source at 38.2 mA. Its pins 14 and 15 are connected to the Arduino Uno's digital pins 3 and 4 respectively. The Arduino Uno is connected to a HC-SR04 ultrasonic distance sensor, which is also powered by the same 5.00 V source. The sensor's trig pin (TRIG) is connected to digital pin 3, and its echo pin (ECHO) is connected to digital pin 4. The Arduino Uno's digital pins 3 and 4 are connected to the TX and RX pins of a SoftwareSerial object named mySerial. The mySerial object is configured to operate at 4800 baud rate. The code reads data from the HC-SR04 sensor via the mySerial port and prints it to the Serial Monitor. The output in the Serial Monitor shows distance values ranging from 40 to 39 cm.

# ตัวอย่างที่ 12: Ultrasonic Distance Sensor

```
#include <SoftwareSerial.h>
SoftwareSerial mySerial( 3, 4 ); //rx, tx
const int TRIG_PIN = PB0, ECHO_PIN = PB2;
#define SOUND_SPEED (340) // m/s

unsigned long readSensor() {
    digitalWrite( TRIG_PIN, HIGH );
    delayMicroseconds(10);
    digitalWrite( TRIG_PIN, LOW );
    return pulseIn( ECHO_PIN, HIGH, 50000 );
}

void setup() {
    mySerial.begin(4800);
    pinMode( ECHO_PIN, INPUT );
    pinMode( TRIG_PIN, OUTPUT );
    digitalWrite( TRIG_PIN, LOW );
}

void loop() {
    unsigned long duration = readSensor(); // usec
    int d = (SOUND_SPEED*duration)/2/10000;
    mySerial.println( d ); // distance in cm.
    delay(1000);
}
```

Attiny85

```
#include <SoftwareSerial.h>
SoftwareSerial mySerial( 3, 4 ); //rx, tx

#define BUF_LEN 64
char buf[ BUF_LEN+1 ];
int index = 0;

void setup() {
    Serial.begin(115200);
    mySerial.begin(4800);
}

void loop() {
    while (mySerial.available() > 0) {
        if ( index < BUF_LEN ) {
            buf[index++] = mySerial.read();
        } else break;
    }
    if (index > 0) {
        buf[index] = '\0';
        index = 0;
        Serial.print(buf);
    }
}
```

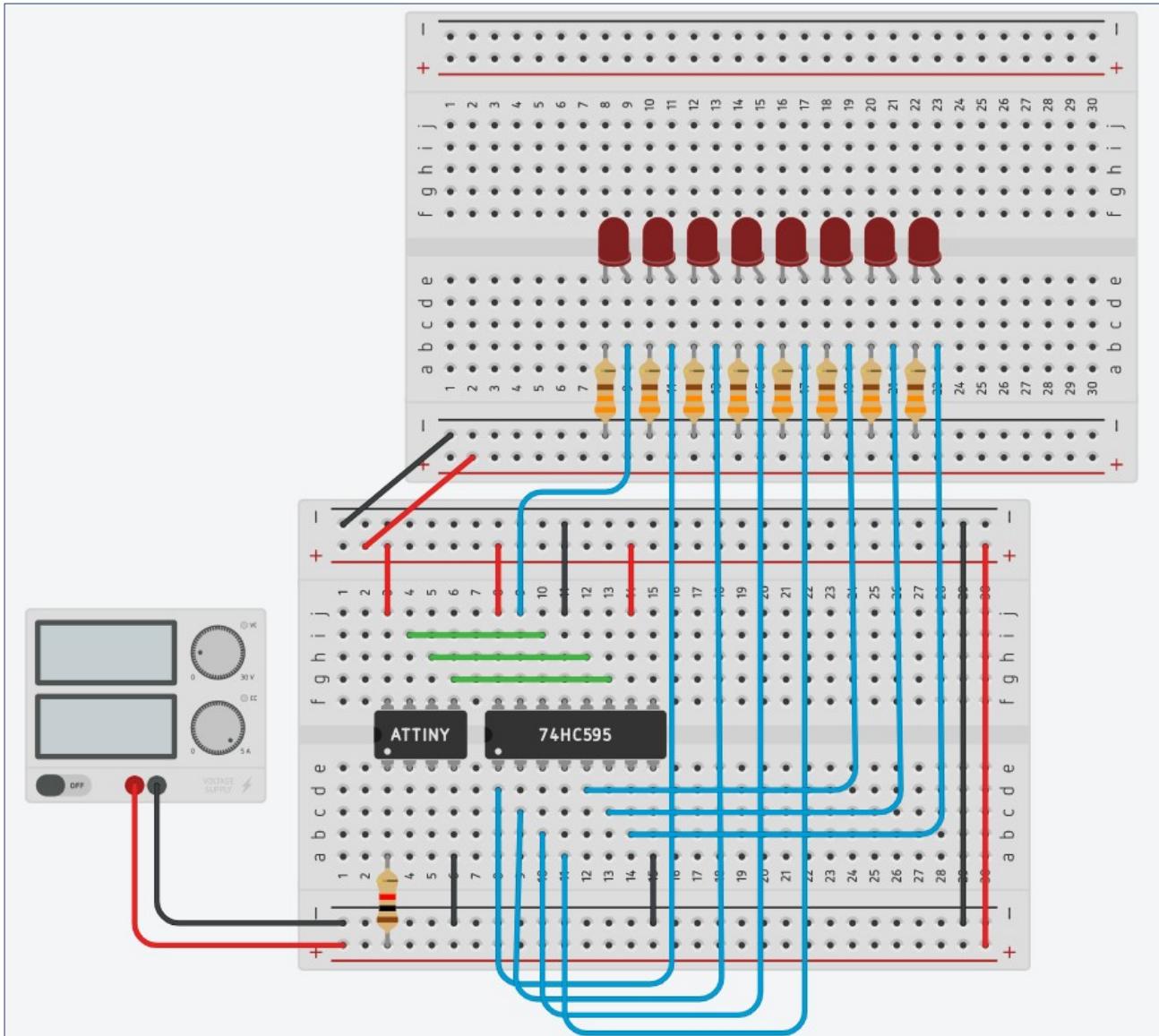
Uno

ข้อสังเกต: คำสั่ง **pulseIn()** ของ **Arduino API** ใช้สำหรับวัดความกว้างของสัญญาณพลส์

# ตัวอย่างที่ 13: 74HC595 – LEDs

โจทย์ฝึกหัด: ให้ศึกษา  
การทำงานของไอซี  
**74HC595** จากเอกสาร  
ของผู้ผลิต

ตัวอย่างการต่อวงจรเพื่อ  
สาธิตการเลื่อนข้อมูล 8  
บิต (หรือ 1 ไบต์) โดยส่ง  
จาก **Attiny85** ออกไป  
ยังไอซี **74HC595** ซึ่งมี  
เอาต์พุต 8 ขา นำไปต่อ  
กับวงจร **LED** แบบ 8  
ตำแหน่ง



# ตัวอย่างที่ 13: 74HC595 – LEDs (Code 1/2)

```
const int SH_CP_PIN = PB0; // shift clk pin
const int ST_CP_PIN = PB1; // storage clk pin
const int DS_PIN = PB2; // serial data pin

byte data = 0x01; // data byte

void setup() {
    pinMode( SH_CP_PIN, OUTPUT );
    pinMode( ST_CP_PIN, OUTPUT );
    pinMode( DS_PIN, OUTPUT );
    digitalWrite( SH_CP_PIN, LOW );
    digitalWrite( ST_CP_PIN, LOW );
}
```

```
void shiftDataOut( byte data ) {
    int bit;
    digitalWrite( SH_CP_PIN, LOW );
    for ( int i=0; i < 8; i++ ) { // shift bit, MSB first
        bit = data & 0x80 ? HIGH : LOW;
        digitalWrite( DS_PIN, bit );
        data = data << 1;
        digitalWrite( SH_CP_PIN, HIGH );
        digitalWrite( SH_CP_PIN, LOW );
    }
    digitalWrite( ST_CP_PIN, HIGH );
    digitalWrite( ST_CP_PIN, LOW );
}

void loop() {
    shiftDataOut( data ); // shift one byte to 74HC595
    // rotate-shift-left by 1-bit position
    data = (data << 1) | (data >> 7);
    delay(100);
}
```

ATTiny85 สื่อสารกับ 74HC595 โดยใช้สัญญาณ

- 1) Shift Register Clock (SH\_CP)
- 2) Storage Register Clock (ST\_CP) และ
- 3) Serial Data Input (DS)

# ตัวอย่างที่ 13: 74HC595 – LEDs (Code 2/2)

โค้ดตัวอย่างนี้สาธิตการใช้คำสั่ง **shiftOut()** ซึ่งเป็นคำสั่งของ **Arduino API** สำหรับเลื่อนข้อมูลขนาดหนึ่งไบต์ทีละบิตออกจากไป โดยกำหนดลำดับของการเลื่อนบิต (**Bit Order**) คือ ให้บิต **MSB** ออกไป ก่อน (**MSB First**)

```
const int SH_CP_PIN = PB0; // shift clk pin
const int ST_CP_PIN = PB1; // storage clk pin
const int DS_PIN = PB2; // serial data pin

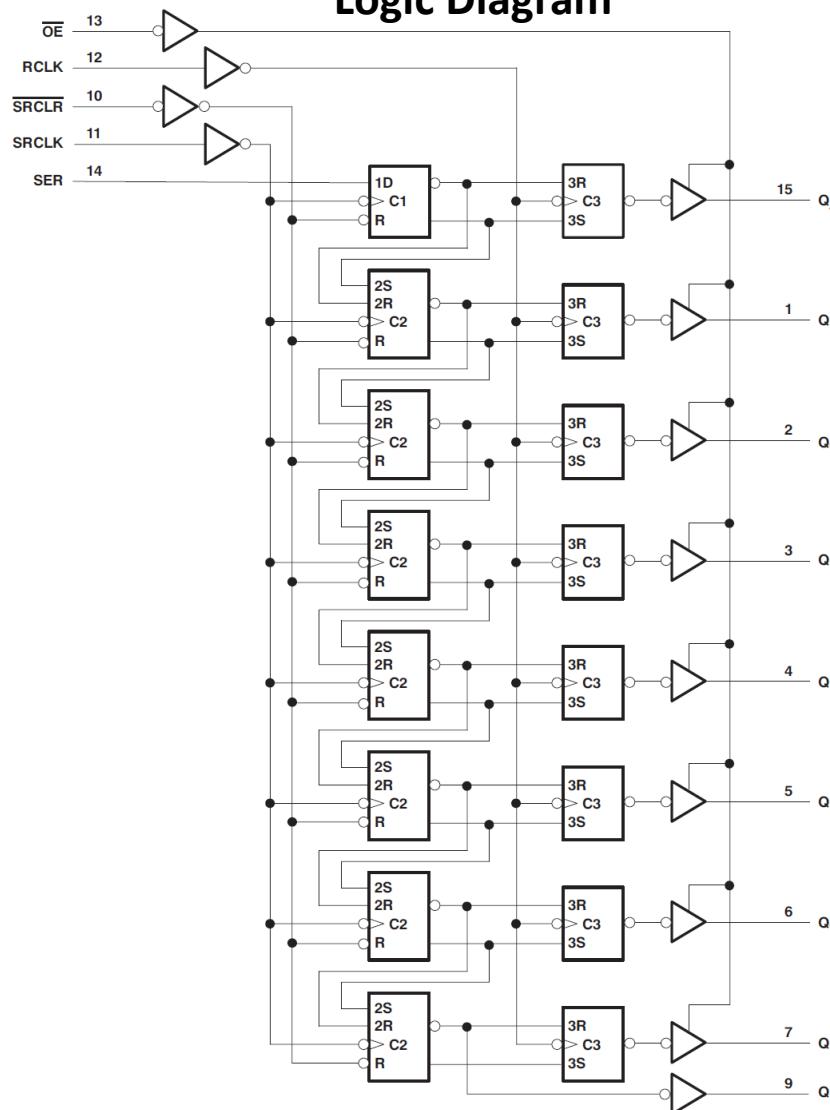
byte data = 0x01; // data byte

void setup() {
    pinMode( SH_CP_PIN, OUTPUT );
    pinMode( ST_CP_PIN, OUTPUT );
    pinMode( DS_PIN, OUTPUT );
    digitalWrite( SH_CP_PIN, LOW );
    digitalWrite( ST_CP_PIN, LOW );
}

void loop() {
    shiftOut( DS_PIN, SH_CP_PIN, MSBFIRST, data );
    digitalWrite( ST_CP_PIN, HIGH );
    digitalWrite( ST_CP_PIN, LOW );
    // rotate-shift-left by 1-bit position
    data = (data << 1) | (data >> 7);
    delay(200);
}
```

# เอกสารอ้างอิงเกี่ยวกับ 74HC595

Logic Diagram



Pin Assignments

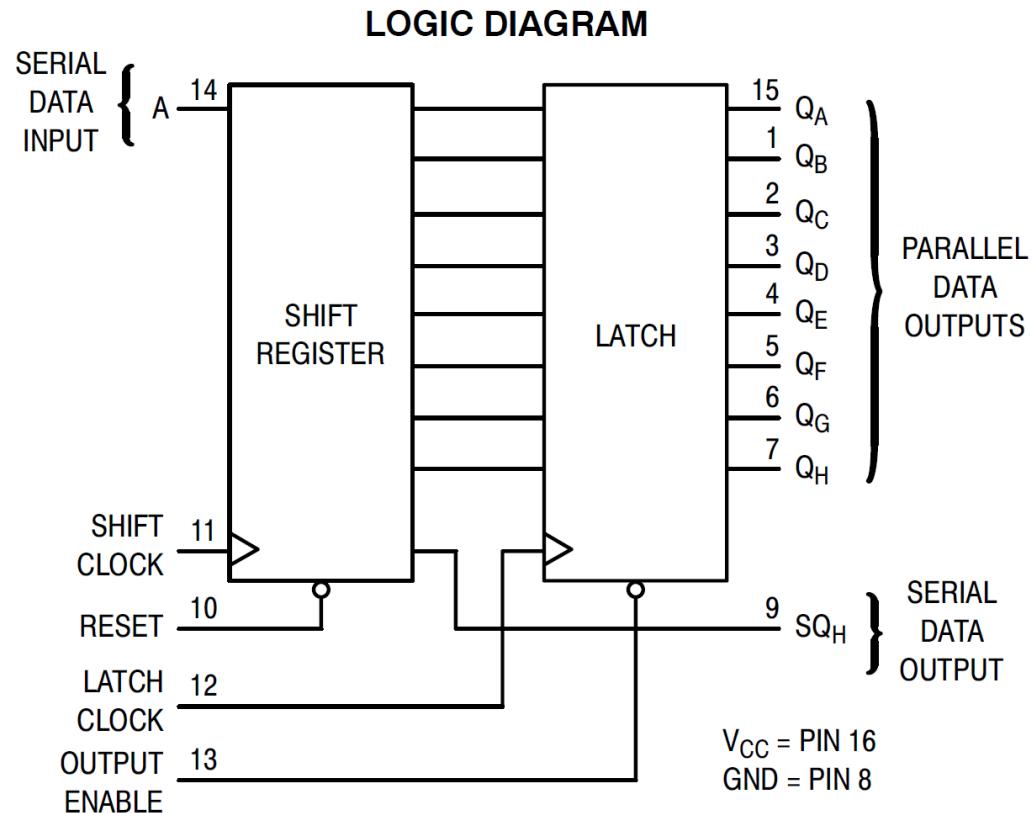
Q <sub>B</sub>	1	16	V <sub>CC</sub>
Q <sub>C</sub>	2	15	Q <sub>A</sub>
Q <sub>D</sub>	3	14	SER
Q <sub>E</sub>	4	13	OE
Q <sub>F</sub>	5	12	RCLK
Q <sub>G</sub>	6	11	SRCLK
Q <sub>H</sub>	7	10	SRCLR
GND	8	9	Q <sub>H</sub> '

# เอกสารอ้างอิงเกี่ยวกับ 74HC595

## PIN ASSIGNMENT

Q <sub>B</sub>	1	16	V <sub>CC</sub>
Q <sub>C</sub>	2	15	Q <sub>A</sub>
Q <sub>D</sub>	3	14	A
Q <sub>E</sub>	4	13	OUTPUT ENABLE
Q <sub>F</sub>	5	12	LATCH CLOCK
Q <sub>G</sub>	6	11	SHIFT CLOCK
Q <sub>H</sub>	7	10	RESET
GND	8	9	SQ <sub>H</sub>

SHIFT CLOCK = Shift Register Clock  
LATCH CLOCK = Storage Register Clock



# เอกสารอ้างอิงเกี่ยวกับ 74HC595

## Pin Descriptions

### INPUTS

#### A (Pin 14)

Serial Data Input. The data on this pin is shifted into the 8-bit serial shift register.

### CONTROL INPUTS

#### Shift Clock (Pin 11)

Shift Register Clock Input. A low– to–high transition on this input causes the data at the Serial Input pin to be shifted into the 8-bit shift register.

#### Reset (Pin 10)

Active–low, Asynchronous, Shift Register Reset Input. A low on this pin resets the shift register portion of this device only. The 8-bit latch is not affected.

#### Latch Clock (Pin 12)

Storage Latch Clock Input. A low–to–high transition on this input latches the shift register data.

#### Output Enable (Pin 13)

Active–low Output Enable. A low on this input allows the data from the latches to be presented at the outputs. A high on this input forces the outputs ( $Q_A$ – $Q_H$ ) into the high–impedance state. The serial output is not affected by this control unit.

### OUTPUTS

#### $Q_A$ – $Q_H$ (Pins 15, 1, 2, 3, 4, 5, 6, 7)

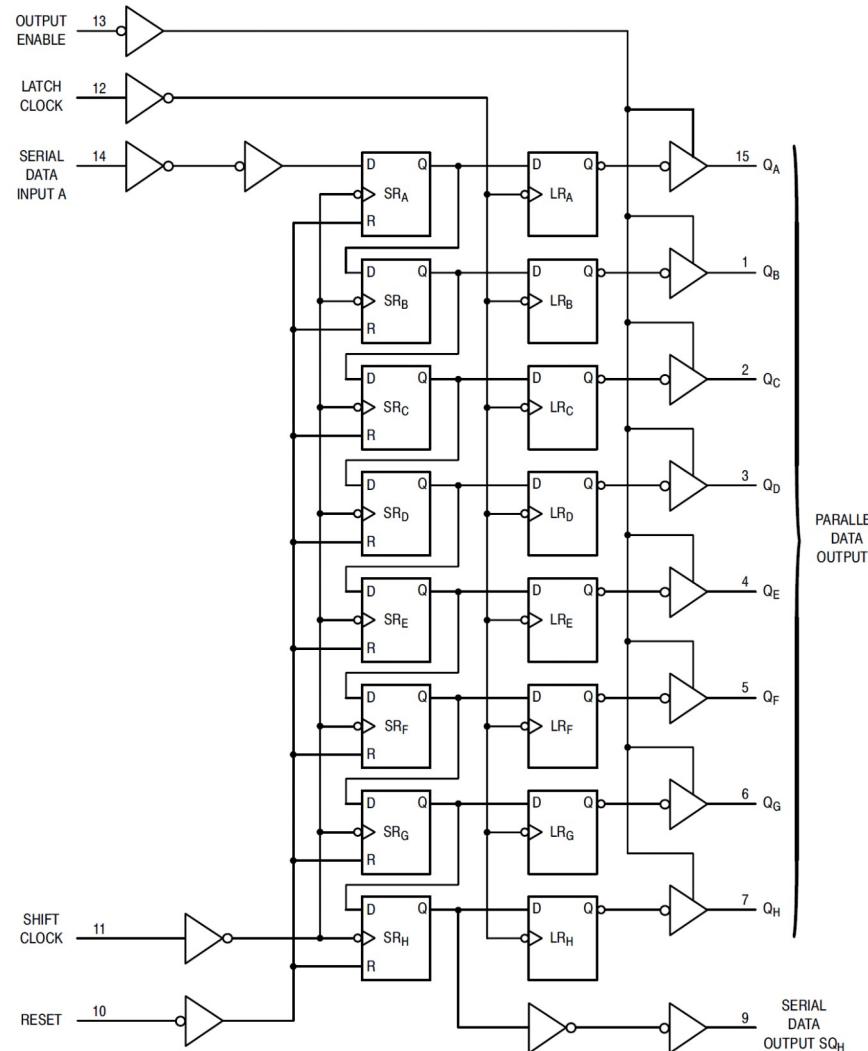
Noninverted, 3-state, latch outputs.

#### $SQ_H$ (Pin 9)

Noninverted, Serial Data Output. This is the output of the eighth stage of the 8-bit shift register. This output does not have three–state capability.

# ເອກສາຣອ້າງອີງເກື່ຍວກັນ 74HC595

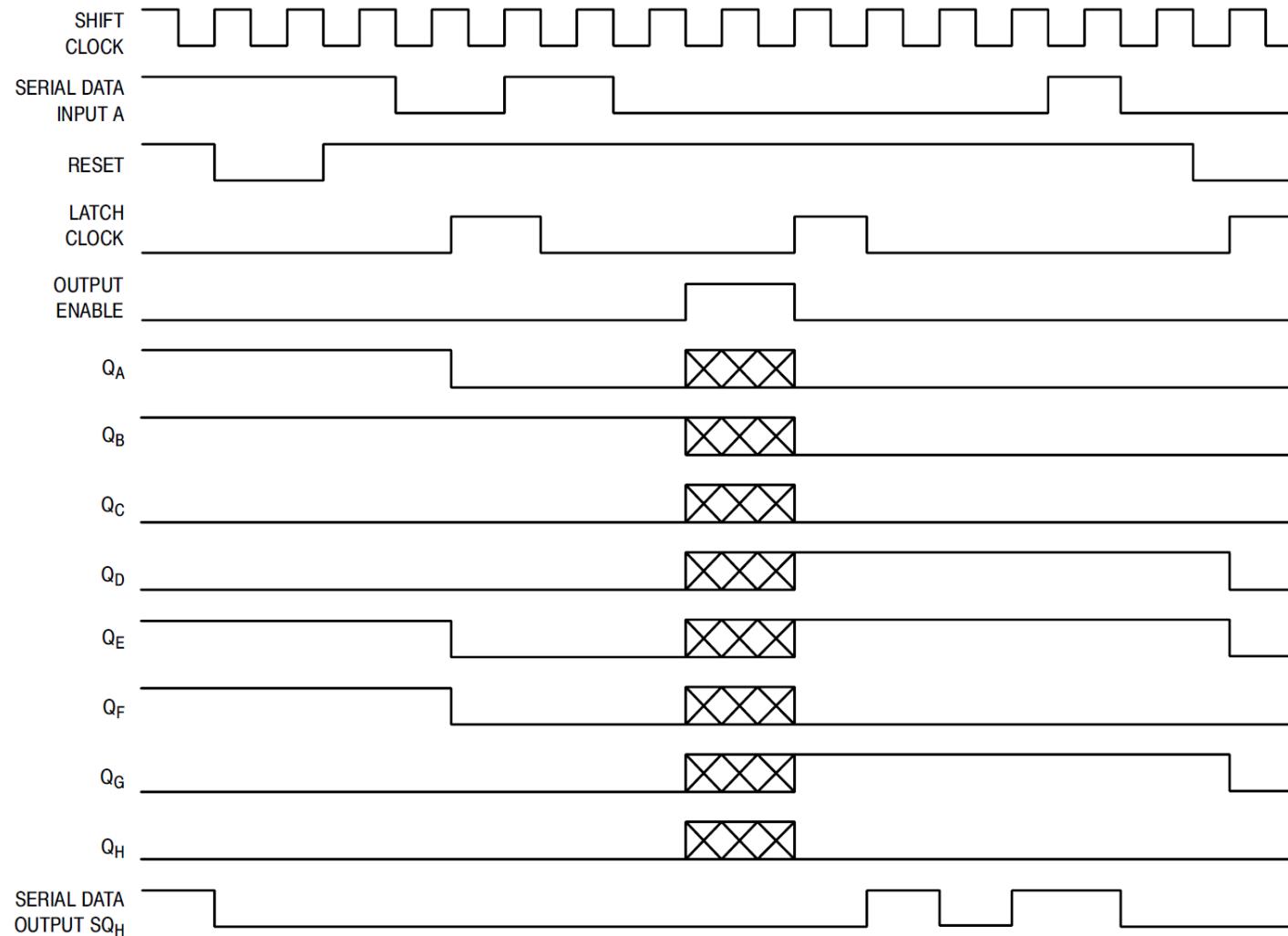
## Logic Diagram



Reference: <https://www.onsemi.com/pub/Collateral/MC74HC595-D.PDF>

# เอกสารอ้างอิงเกี่ยวกับ 74HC595

## Timing Diagram



NOTE: implies that the output is in a high-impedance state.

Reference: <https://www.onsemi.com/pub/Collateral/MC74HC595-D.PDF>

# เอกสารอ้างอิง: ATtiny85 Datasheet



## Atmel 8-bit AVR Microcontroller with 2/4/8K Bytes In-System Programmable Flash

ATtiny25/V / ATtiny45/V / ATtiny85/V

### Features

- High Performance, Low Power AVR® 8-Bit Microcontroller
- Advanced RISC Architecture
  - 120 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
- Non-volatile Program and Data Memories
  - 2/4/8K Bytes of In-System Programmable Program Memory Flash
    - Endurance: 10,000 Write/Erase Cycles
  - 128/256/512 Bytes In-System Programmable EEPROM
    - Endurance: 100,000 Write/Erase Cycles
  - 128/256/512 Bytes Internal SRAM
  - Programming Lock for Self-Programming Flash Program and EEPROM Data Security
- Peripheral Features
  - 8-bit Timer/Counter with Prescaler and Two PWM Channels
  - 8-bit High Speed Timer/Counter with Separate Prescaler
    - 2 High Frequency PWM Outputs with Separate Output Compare Registers
    - Programmable Dead Time Generator
  - USI – Universal Serial Interface with Start Condition Detector
  - 10-bit ADC
    - 4 Single Ended Channels
    - 2 Differential ADC Channel Pairs with Programmable Gain (1x, 20x)
    - Temperature Measurement
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
- Special Microcontroller Features
  - debugWIRE On-chip Debug System
  - In-System Programmable via SPI Port
  - External and Internal Interrupt Sources
  - Low Power Idle, ADC Noise Reduction, and Power-down Modes
  - Enhanced Power-on Reset Circuit
  - Programmable Brown-out Detection Circuit
  - Internal Calibrated Oscillator
- I/O and Packages
  - Six Programmable I/O Lines
  - 8-pin PDIP, 8-pin SOIC, 20-pad QFN/MLF, and 8-pin TSSOP (only ATtiny45/V)
- Operating Voltage
  - 1.8 - 5.5V for ATtiny25V/45V/85V
  - 2.7 - 5.5V for ATtiny25/45/85
- Speed Grade
  - ATtiny25V/45V/85V: 0 - 4 MHz @ 1.8 - 5.5V, 0 - 10 MHz @ 2.7 - 5.5V
  - ATtiny25/45/85: 0 - 10 MHz @ 2.7 - 5.5V, 0 - 20 MHz @ 4.5 - 5.5V
- Industrial Temperature Range
- Low Power Consumption
  - Active Mode:
    - 1 MHz, 1.8V: 300 µA
  - Power-down Mode:
    - 0.1 µA at 1.8V

Rev. 2586Q-AVR-08/2013