

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

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

แนะนำไมโครคอนโทรลเลอร์ ATtiny85

- เป็นชิปไมโครคอนโทรลเลอร์ในตระกูล **ATtiny (TinyAVR)** ของบริษัท **Atmel / Microchip**
- ภายในมีซีพียูขนาด **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**
- สามารถเขียนโค้ด-จำลองการทำงานร่วมกับวงจรอิเล็กทรอนิกส์ได้ โดยใช้ **AUTODESK Tinkercad – Circuits**
- รองรับการเขียนโค้ด โดยใช้ชุดคำสั่ง (**API**) ของ **Arduino**

ไมโครคอนโทรลเลอร์ **ATtiny85**

- **ATtiny85** มีราคาไม่แพง (~50 บาท) ขนาดเล็ก
- ใช้ตัวถังแบบ **DIP-8** สามารถนำไปเสียบลงบนบอร์ดบอร์ดได้
- 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
Program Memory Type	Flash
Program Memory Size (KB)	8
CPU Speed (MIPS/DMIPS)	20
SRAM (bytes)	512
Data EEPROM/HEF (bytes)	512
Digital Communication Peripherals	1-SPI, 1-I2C
Capture/Compare/PWM Peripherals	5PWM
Timers	2 x 8-bit
Number of Comparators	1
Temperature Range (°C)	-40 to 85
Operating Voltage Range (V)	1.8 to 5.5
Pin Count	8

ชนิดข้อมูล

ความเร็วของซีพียู
(สูงสุด)

ขนาดหน่วยความจำ
EEPROM

จำนวนสัญญาณ
PWM

จำนวนวงจรเปรียบเทียบแรงดันไฟฟ้า

ช่วงแรงดันไฟเลี้ยง

ขนาดหน่วยความจำ
สำหรับโปรแกรม

ขนาดหน่วยความจำ
SRAM

จำนวนวงจรสำหรับ
สื่อสารข้อมูลแบบ

SPI & I2C

จำนวนวงจรตัวนับ
8-bit Timers

ช่วงอุณหภูมิในการ
ทำงานของไอซี

จำนวนขาของไอซี

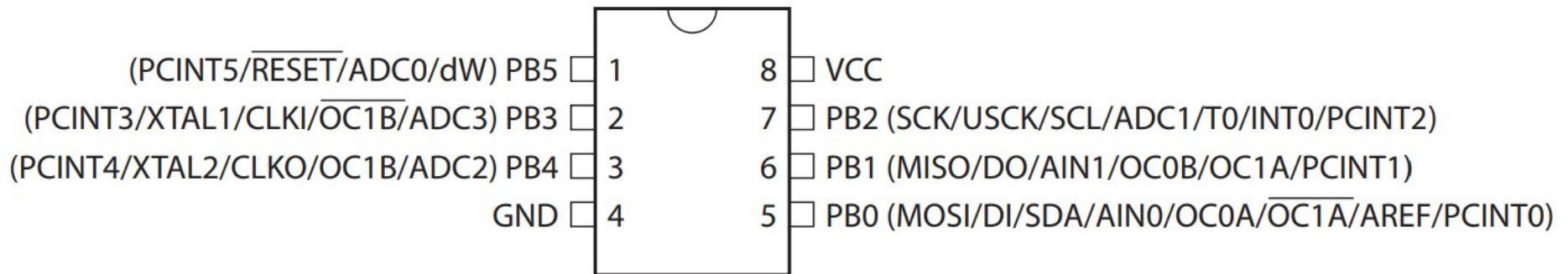
ตำแหน่งขาของ ATtiny85 (PinOut)

ขาหมายเลข 1 ตรงกับ **PB5** ปรกติจะถูกใช้เป็นขา **/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

PDIP/SOIC/TSSOP



NOTE: TSSOP only for ATtiny45/V

http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-2586-AVR-8-bit-Microcontroller-ATtiny25-ATtiny45-ATtiny85_Datasheet.pdf

ตัวอย่างที่ 1: LED Blink (Circuit View)

การต่อวงจร **LED** ที่ขา **PB0** และ
เขียนโค้ด เพื่อให้ **LED** กระพริบ

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

5.00 V
13.5 mA

ATTINY

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

ตัวอย่างที่ 1: LED Blink (Component List)

รายการอุปกรณ์สำหรับวงจรในตัวอย่างที่ 1
(ไม่รวมบอร์ดและสายไฟ)



TIN
KER
CAD

attiny85_led_blink

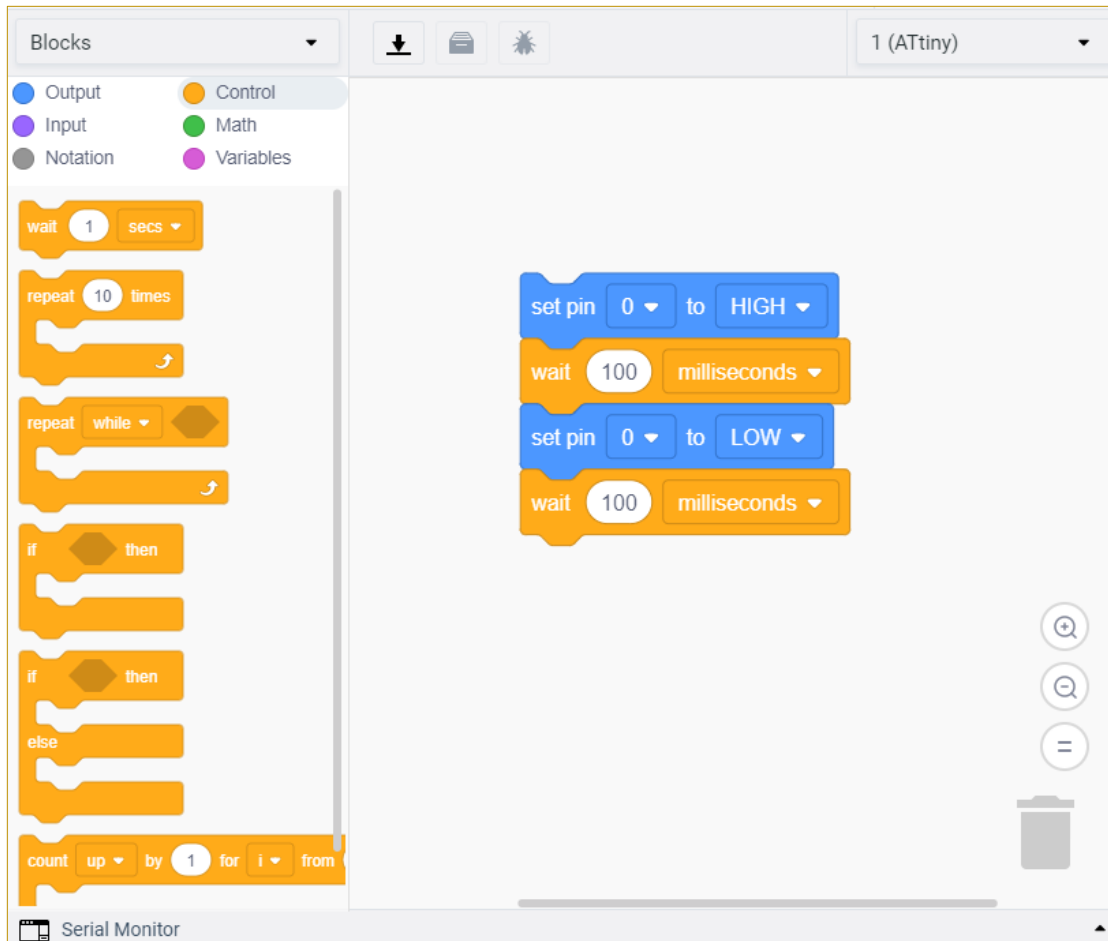
All changes saved

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**)



The screenshot shows the Arduino IDE's block-based coding environment. The workspace contains a sequence of four blocks: a blue 'set pin 0 to HIGH' block, an orange 'wait 100 milliseconds' block, a blue 'set pin 0 to LOW' block, and another orange 'wait 100 milliseconds' block. The left sidebar shows a palette of block categories including Output, Input, Notation, Control, Math, and Variables. The top right corner indicates the target board is '1 (ATtiny)'. At the bottom, there is a 'Serial Monitor' tab.

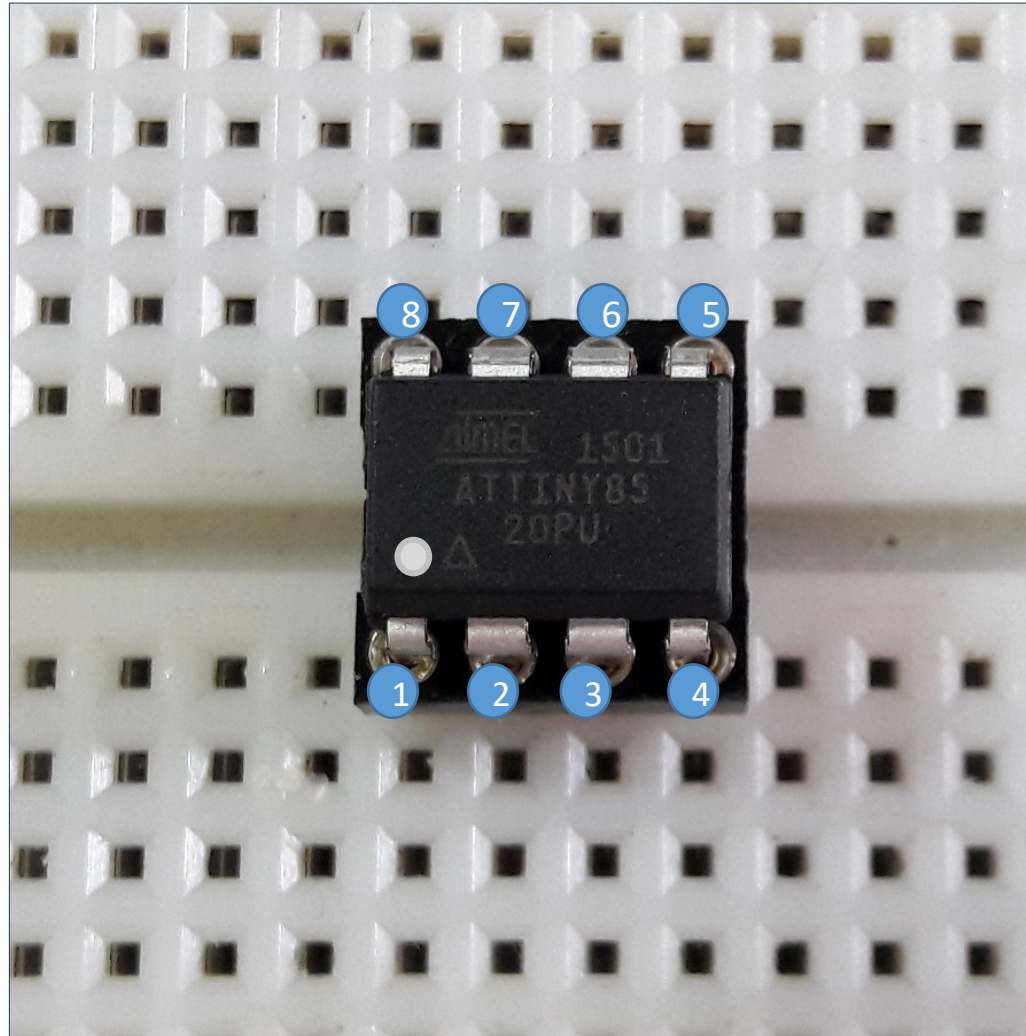
มุมมองการเขียนโค้ดด้วย
ภาษาคอมไพเลอร์ **C/C++**

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


รายการอุปกรณ์ (สำหรับตัวอย่าง **LED Blink**)

- เบริดบอร์ด
- 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)



**USB ISP
Programmer**

**IDC Cable
(2x5 Pins)**

**อุปกรณ์สำหรับแปลง
คอนเนกเตอร์ 2x5
ให้เป็น 2x3 Pins
(ICSP Header)**

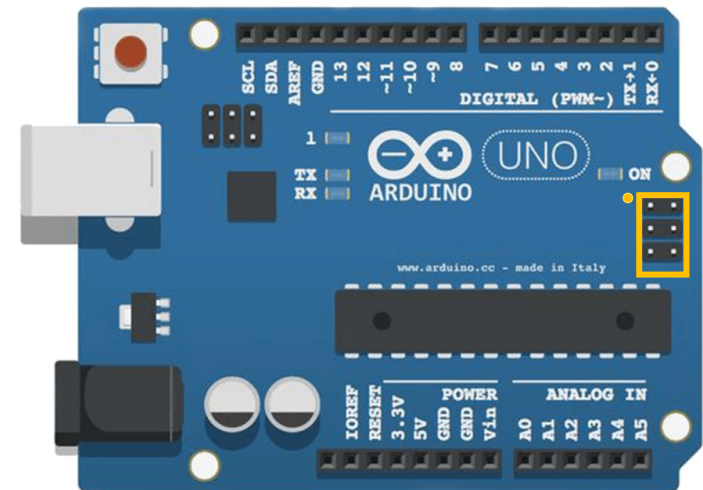
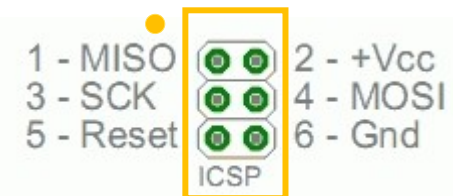
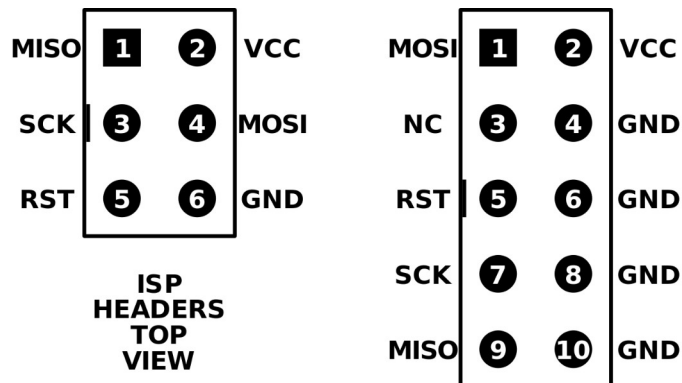
**ถ้าไม่ใช้อุปกรณ์ในลักษณะนี้ ก็สามารถใช้ Arduino Uno
ทำหน้าที่เป็น Arduino ISP Programmer ได้**
<https://www.arduino.cc/en/tutorial/arduinoISP>

**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.”

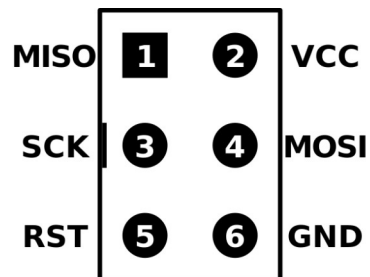
https://en.wikipedia.org/wiki/In-system_programming



6-pin and 10-pin AVR ISP headers

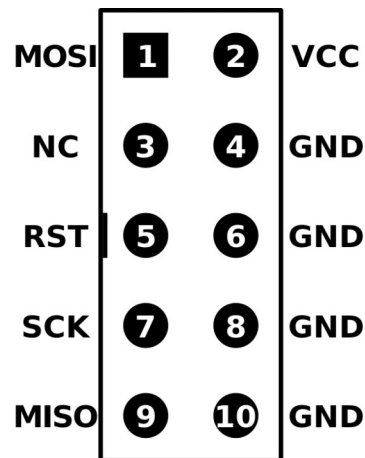
ICSP-to-ATtiny85 Wiring

**6-pin AVR
ISP header**

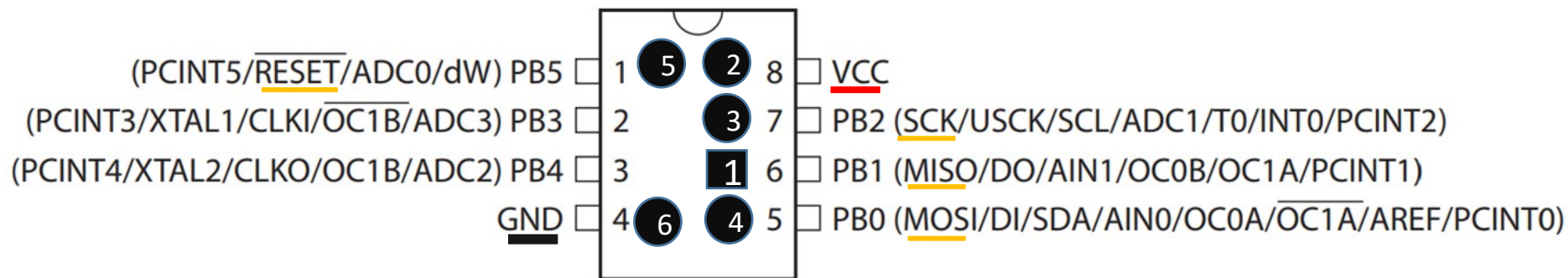


ISP HEADERS
TOP VIEW

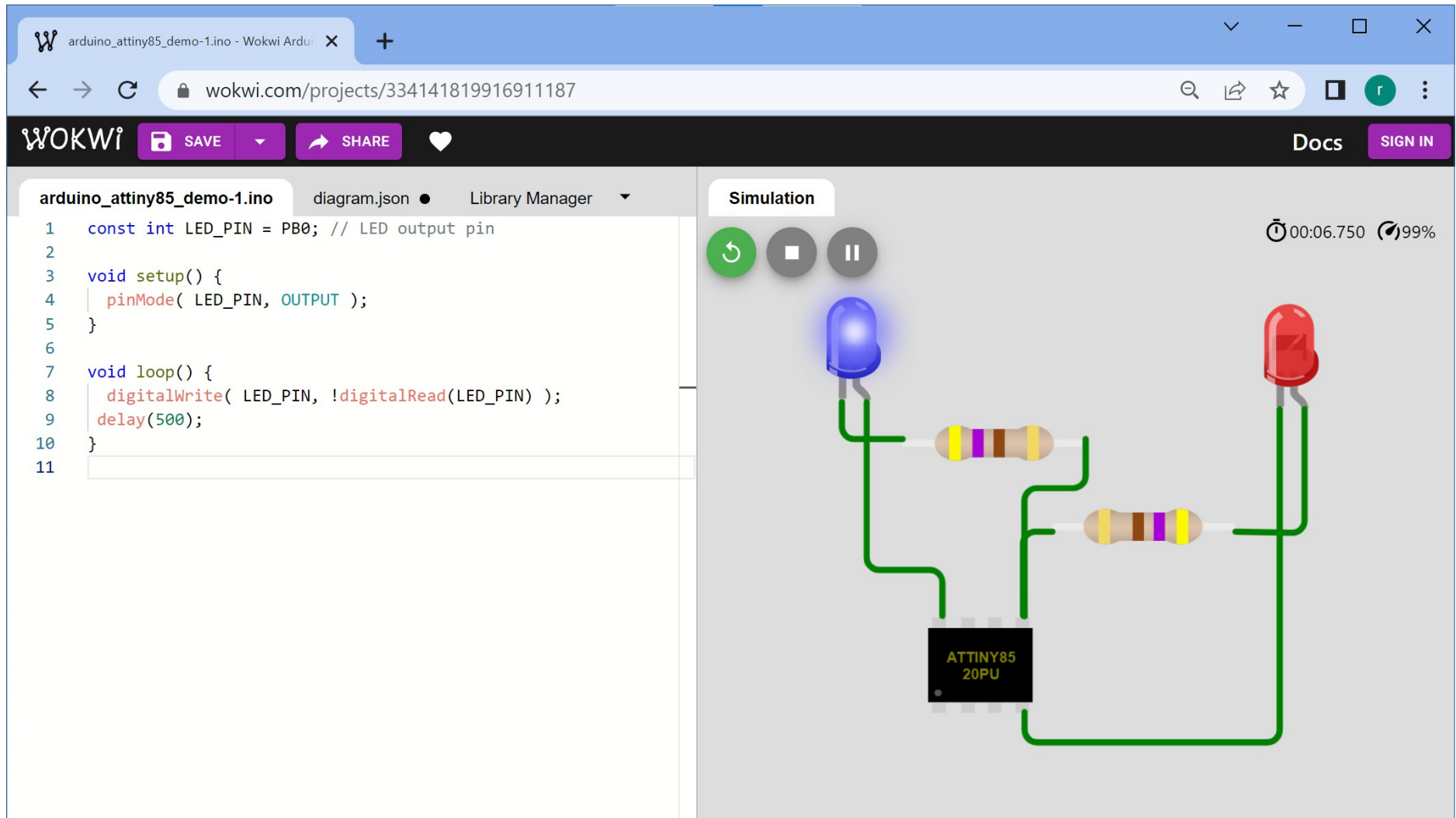
**10-pin AVR
ISP header**



PDIP/SOIC/TSSOP



การใช้ซอฟต์แวร์ WokWi Simulator

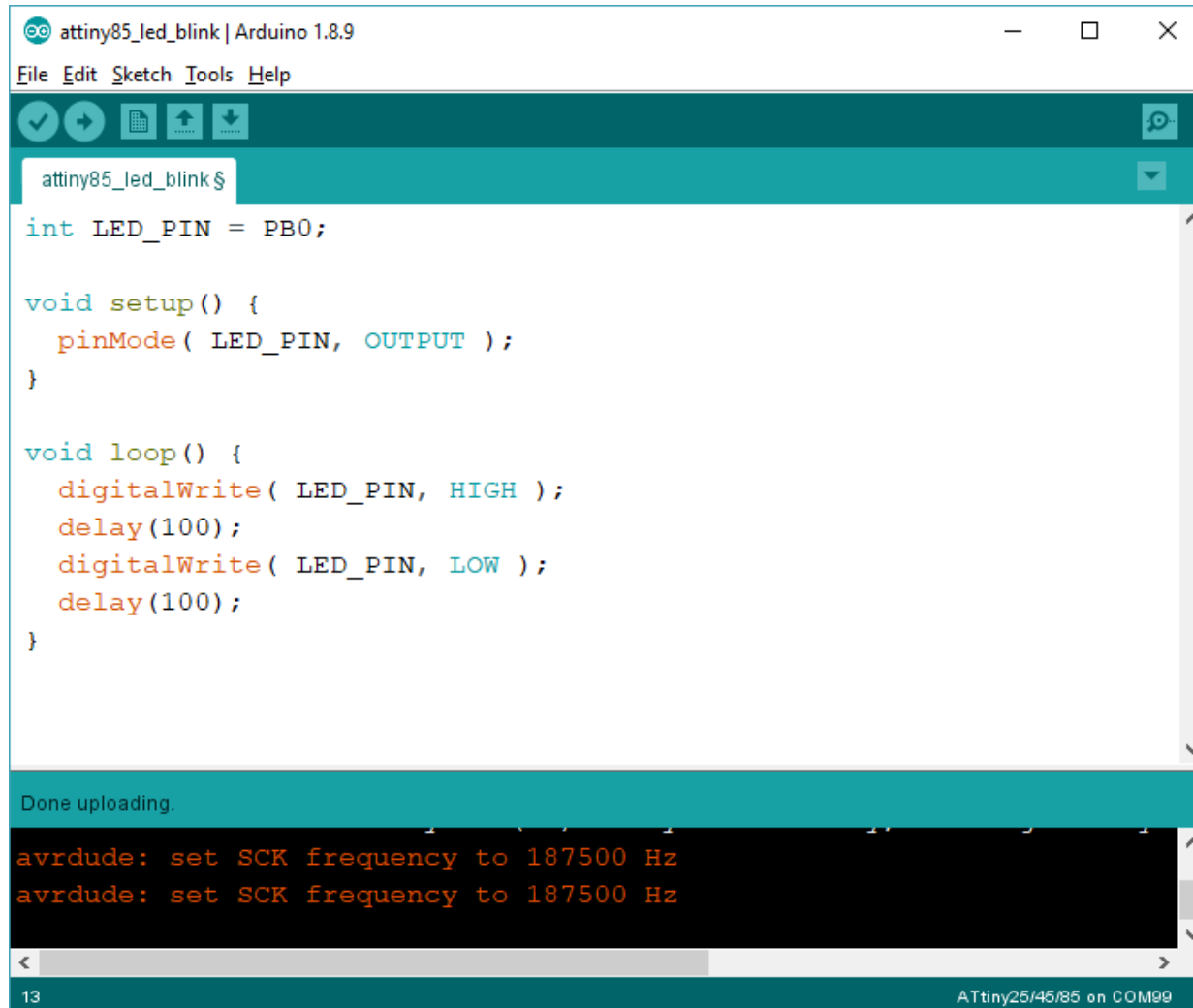


The screenshot displays the WokWi Simulator interface. The top navigation bar includes the WokWi logo, a 'SAVE' button, a 'SHARE' button, a heart icon, 'Docs', and a 'SIGN IN' button. The browser address bar shows the URL 'wokwi.com/projects/334141819916911187'. The main interface is split into two panels. The left panel is a code editor for 'arduino_attiny85_demo-1.ino', containing the following code:

```
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
```

The right panel is the 'Simulation' window, which shows a circuit diagram. It features an 'ATTINY85 20PU' microcontroller at the bottom. Two LEDs are connected to the microcontroller: a blue LED on the left and a red LED on the right. Each LED is connected to a resistor. The simulation is running, as indicated by the 'Simulation' title, a play button, a stop button, a pause button, a timer showing '00:06.750', and a refresh icon with '99%'.

การคอมไพล์โค้ดและอัปโหลดลงชิป ATTiny85



```
attiny85_led_blink | Arduino 1.8.9
File Edit Sketch Tools Help
attiny85_led_blink $
int LED_PIN = PB0;

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

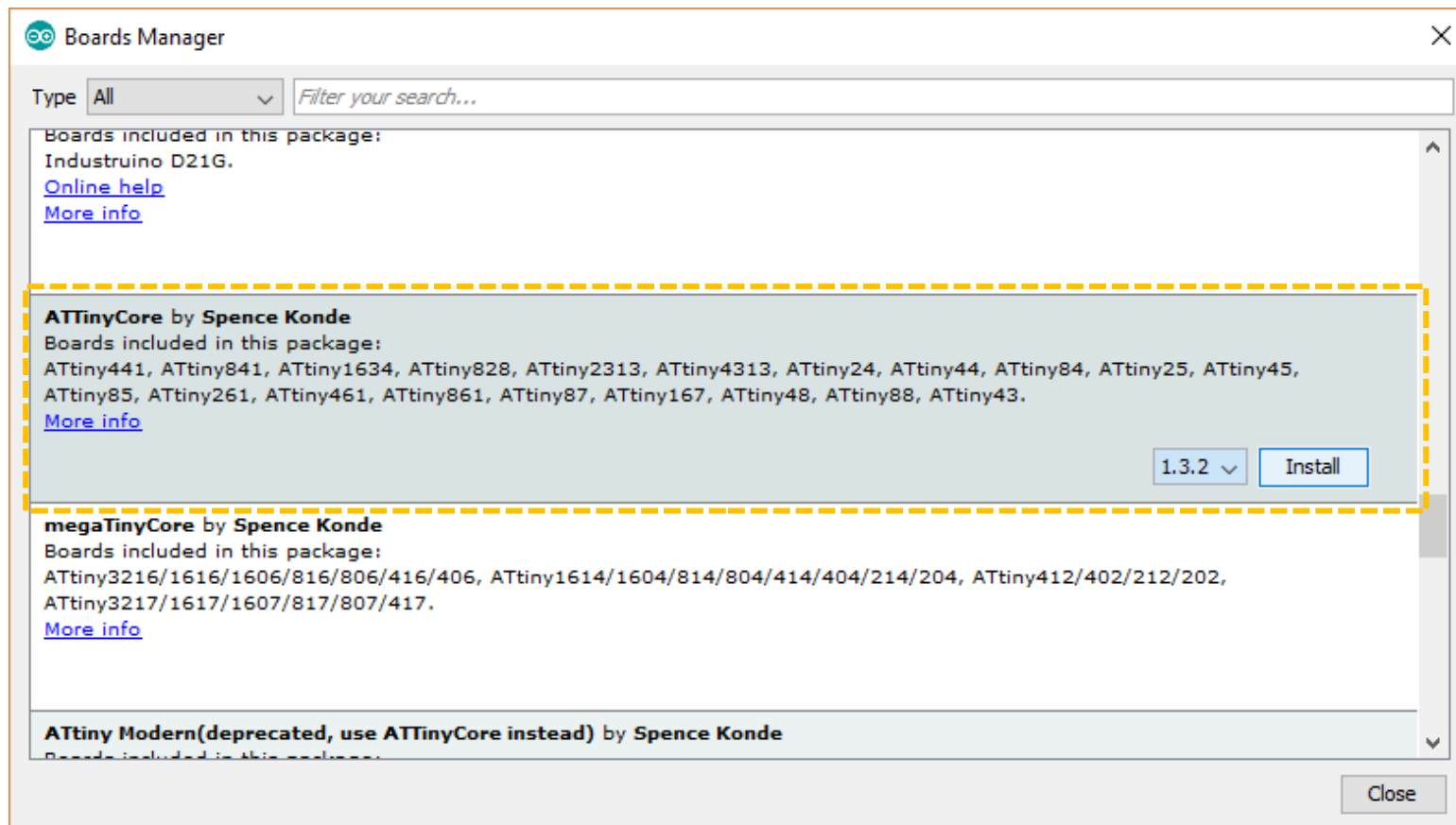
void loop() {
  digitalWrite( LED_PIN, HIGH );
  delay(100);
  digitalWrite( LED_PIN, LOW );
  delay(100);
}

Done uploading.
avrduide: set SCK frequency to 187500 Hz
avrduide: set SCK frequency to 187500 Hz
13 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 (ถ้ามี **URL** อื่น ๆ ให้ใช้สัญลักษณ์ , เป็นตัวแบ่ง) จากนั้นทำคำสั่งจากเมนู **Tools > Boards Manager** เพื่อดาวน์โหลดและติดตั้ง **ATTinyCore** ตามรูปตัวอย่าง



<https://github.com/SpenceKonde/ATTinyCore>

การเลือกใช้ ATTiny85 ใน Arduino IDE

The screenshot shows the Arduino IDE interface with the 'Tools' menu open. The 'Board' option is selected, and the list of boards is expanded. The board 'ATTiny25/45/85' is highlighted. The configuration details for this board are shown in the right-hand pane.

Tools Menu:

- Auto Format (Ctrl+T)
- Archive Sketch
- Fix Encoding & Reload
- Manage Libraries... (Ctrl+Shift+I)
- Serial Monitor (Ctrl+Shift+M)
- Serial Plotter (Ctrl+Shift+L)
- WiFi101 / WiFININA Firmware Updater
- nRF5 Flash SoftDevice
- Board: "ATTiny25/45/85"**
- Chip: "ATTiny85"
- Clock: "8 MHz (internal)"
- B.O.D. Level: "B.O.D. Disabled"
- Save EEPROM: "EEPROM retained"
- Timer 1 Clock: "CPU"
- LTO (1.6.11+ only): "Enabled"
- millis()/micros(): "Enabled"
- Port
- Get Board Info
- Programmer: "USBasp (ATTinyCore)"
- Burn Bootloader

Board List:

- Arduino Gemma
- Adafruit Circuit Playground
- Arduino Yún Mini
- Arduino Industrial 101
- Linino One
- Arduino Uno WiFi
- Arduino ARM (32-bits) Boards
- Arduino Due (Programming Port)
- Arduino Due (Native USB Port)
- ATTinyCore
 - ATtiny24/44/84
 - ATtiny44/84 (optiboot)
 - ATtiny25/45/85
 - ATtiny45/85 (Optiboot)
 - ATtiny48/88
 - ATtiny48/88 (optiboot)
 - ATtiny87/167 (No bootloader)
 - ATtiny167/87 (Optiboot)
 - ATtiny261/461/861
 - ATtiny461/861 (optiboot)

Configuration Details:

- Board: "ATTiny25/45/85"
- Chip: "ATTiny85"
- Clock: "8 MHz (internal)"
- B.O.D. Level: "B.O.D. Disabled"
- Save EEPROM: "EEPROM retained"
- Timer 1 Clock: "CPU"
- LTO (1.6.11+ only): "Enabled"
- millis()/micros(): "Enabled"
- Port
- Get Board Info
- Programmer: "USBasp (ATTinyCore)"
- Burn Bootloader

Code Editor:

```
attiny85_led_blink | Arduino 1.8.9
File Edit Sketch Tools Help
int LED_P
void setup
  pinMode
}
void loop
  digitalWrite
  delay(1
  digitalWrite
  delay(1
}
```

Terminal:

```
Done uploading.
avrdude: s
avrdude: s
```

Status Bar: 10 ATtiny25/45/85 on COM99

การเลือกใช้ความถี่ 8 MHz (Internal)

attiny85_led_blink | Arduino 1.8.9

File Edit Sketch **Tools** Help

- Auto Format Ctrl+T
- Archive Sketch
- Fix Encoding & Reload
- Manage Libraries... Ctrl+Shift+I
- Serial Monitor Ctrl+Shift+M
- Serial Plotter Ctrl+Shift+L
- WiFi101 / Wi-Fi NINA Firmware Updater
- nRF5 Flash SoftDevice
- Board: "ATtiny25/45/85" >
- Chip: "ATtiny85" >
- Clock: "8 MHz (internal)" >**
- B.O.D. Level: "B.O.D. Disabled" >
- Save EEPROM: "EEPROM retained" >
- Timer 1 Clock: "CPU" >
- LTO (1.6.11+ only): "Enabled" >
- millis()/micros(): "Enabled" >
- Port >
- Get Board Info
- Programmer: "USBasp (ATTinyCore)" >
- Burn Bootloader**

Done burning bo

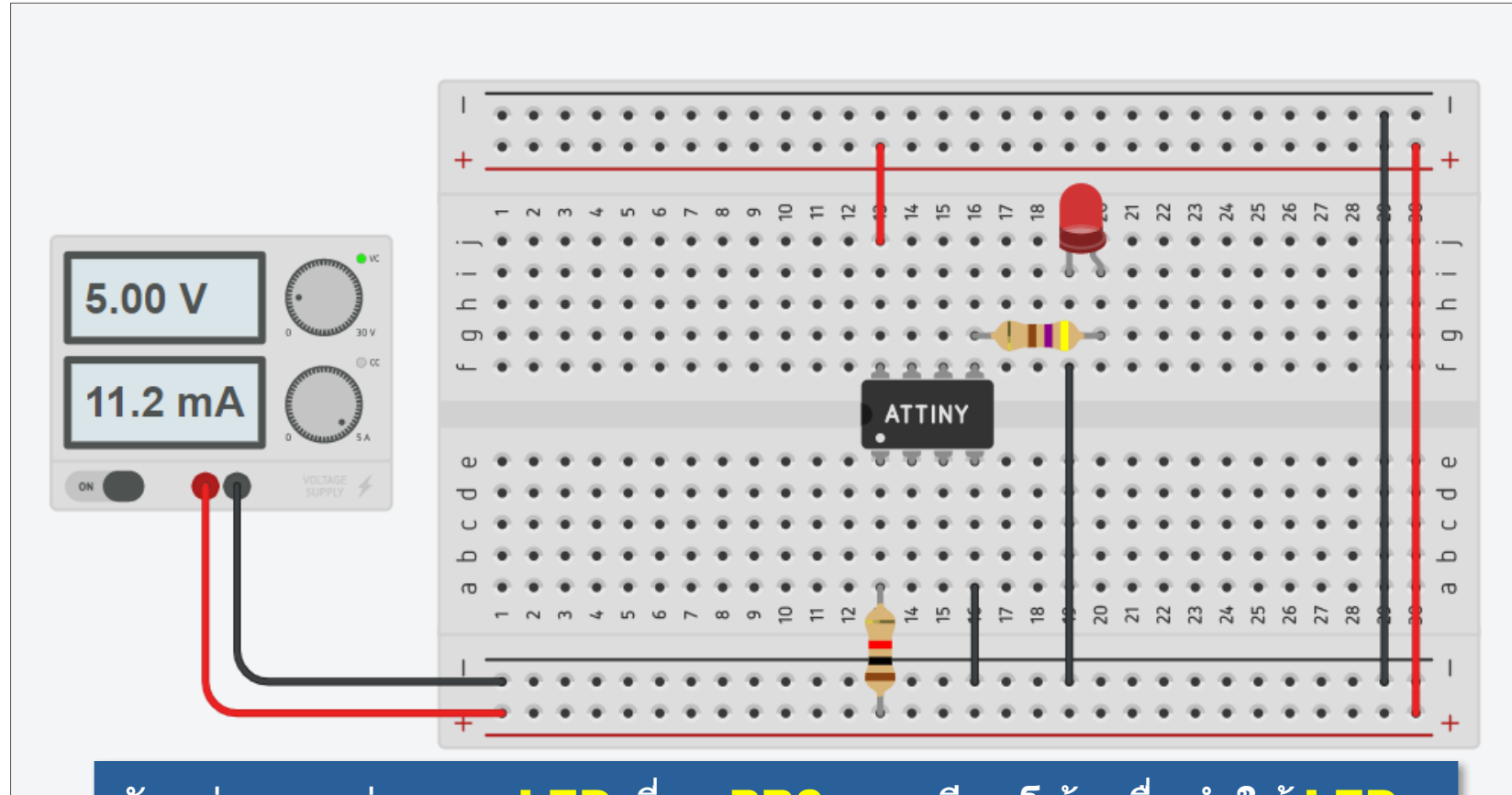
```
avrdude: s
avrdude: s
avrdude: set SCK frequency to 187500 Hz
```

1 ATtiny25/45/85 on COM99

ถ้าจะเปลี่ยนความถี่ **1MHz (Default)** เป็น **8MHz** จะต้องทำขั้นตอน **Burn Bootloader** ด้วยเพื่อเขียนค่าบิตฟิวส์ลงในชิป (**Bit Fuse Settings**)

ตัวอย่างที่ 2: PWM LED Dimming

โจทย์ฝึกหัด: จงวาดผังวงจรเพื่อต่อวงจรบนบอร์ดในตัวอย่างที่ 2



ตัวอย่างการต่อวงจร **LED** ที่ขา **PB0** และเขียนโค้ดเพื่อให้ **LED** ค่อย ๆ เพิ่มหรือลดความสว่าง
ข้อสังเกต: ต่อวงจรเหมือนตัวอย่างที่ 1 ถ้าต่อวงจรใช้งานจริง แนะนำให้ต่อตัวต้านทาน **10k** แบบ **Pull-up** ที่ขาหมายเลข **1** (ขารีเซต)

ตัวอย่างที่ 2: PWM LED Dimming

โจทย์ฝึกหัด: จงต่อวงจรทดลองและทดสอบการทำงานของโค้ดในตัวอย่างที่ 2

ใช้ตัวแปร **value** เป็นตัวนับ ที่มีค่าอยู่ในช่วง **0..255** เพื่อใช้กับคำสั่ง **analogWrite()** และกำหนดความสว่างของ วงจร **LED** ที่ขา **PB0**

ค่าของตัวแปร **value** เป็นจะเพิ่มขึ้นจนถึง **255** แล้วจะลดลงถึง **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);
}
```

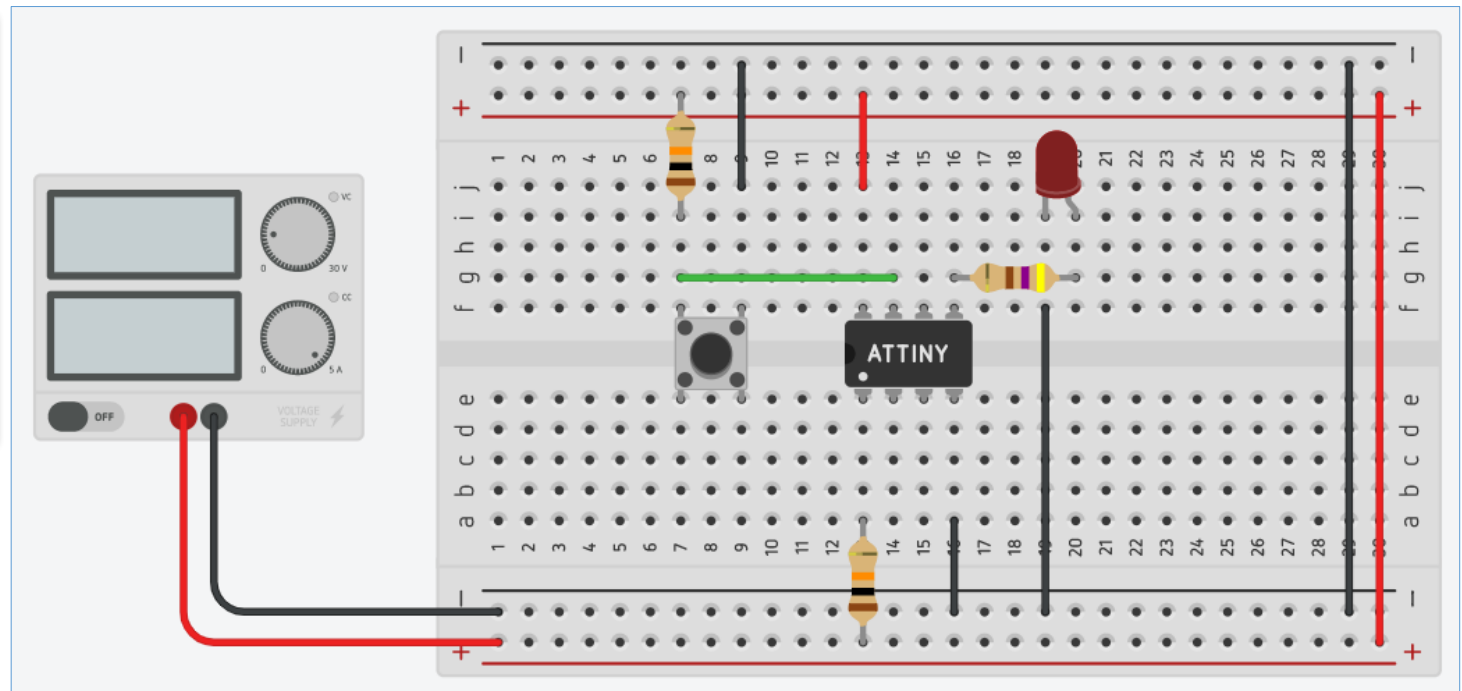
อัตราการเพิ่มขึ้นหรือลดลงสำหรับค่าของตัวแปร **value** ขึ้นอยู่กับคำสั่ง **delay(...)**

ตัวอย่างที่ 3: Push Button - LED Toggle

โจทย์ฝึกหัด: จงวาดผังวงจร (**Schematic**) สำหรับวงจรมอเตอร์บอร์ดในตัวอย่างที่ 3

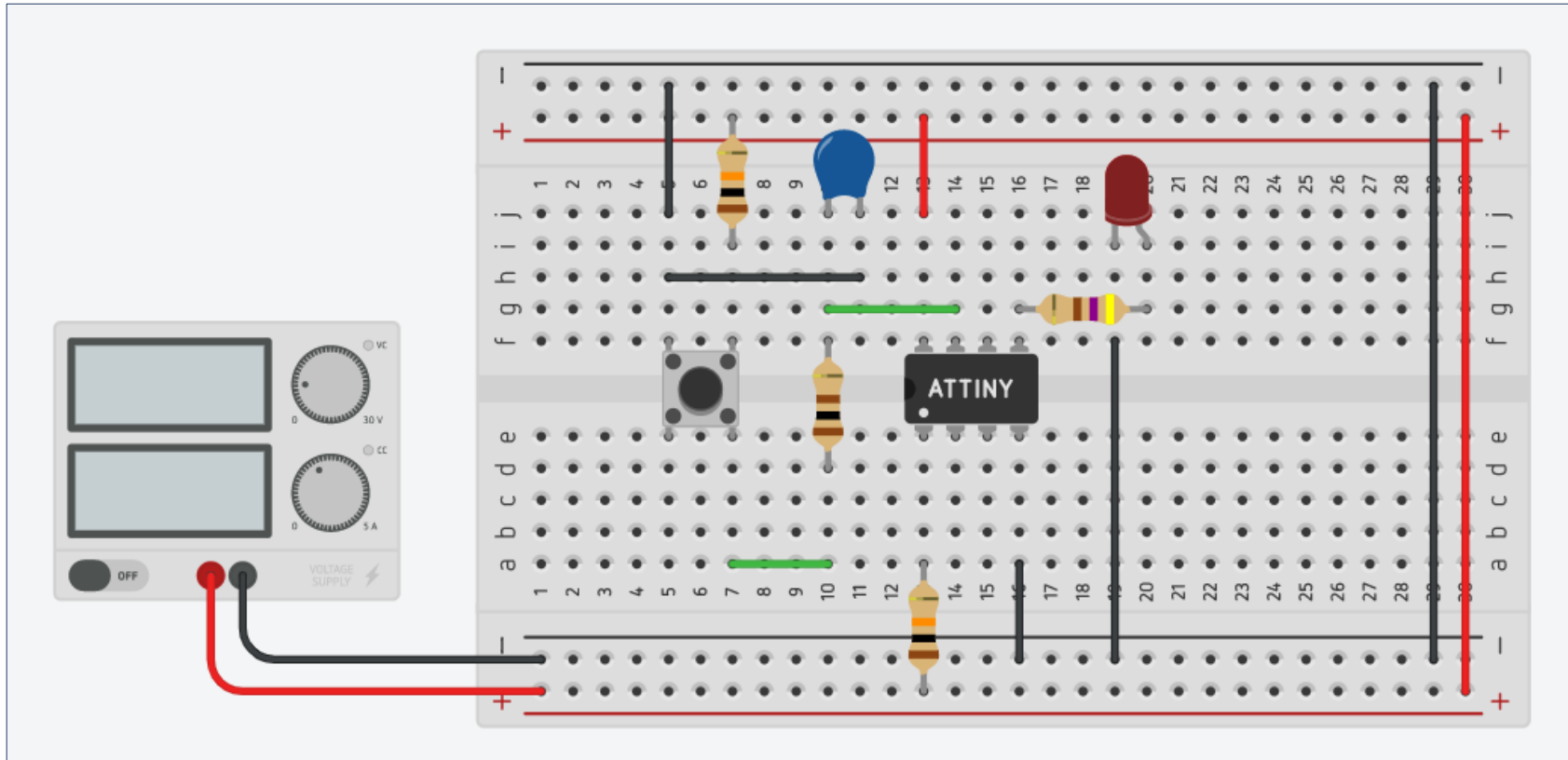
คำถาม: ถ้าทดลองต่อวงจรจริง และไม่ใส่ตัวต้านทาน **Pull-up** ที่ปุ่มกด การทำงานของวงจรมอเตอร์บอร์ด จะมีพฤติกรรมที่แตกต่างจากเดิมหรือไม่ ?

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



ตัวอย่างที่ 3: Push Button - LED

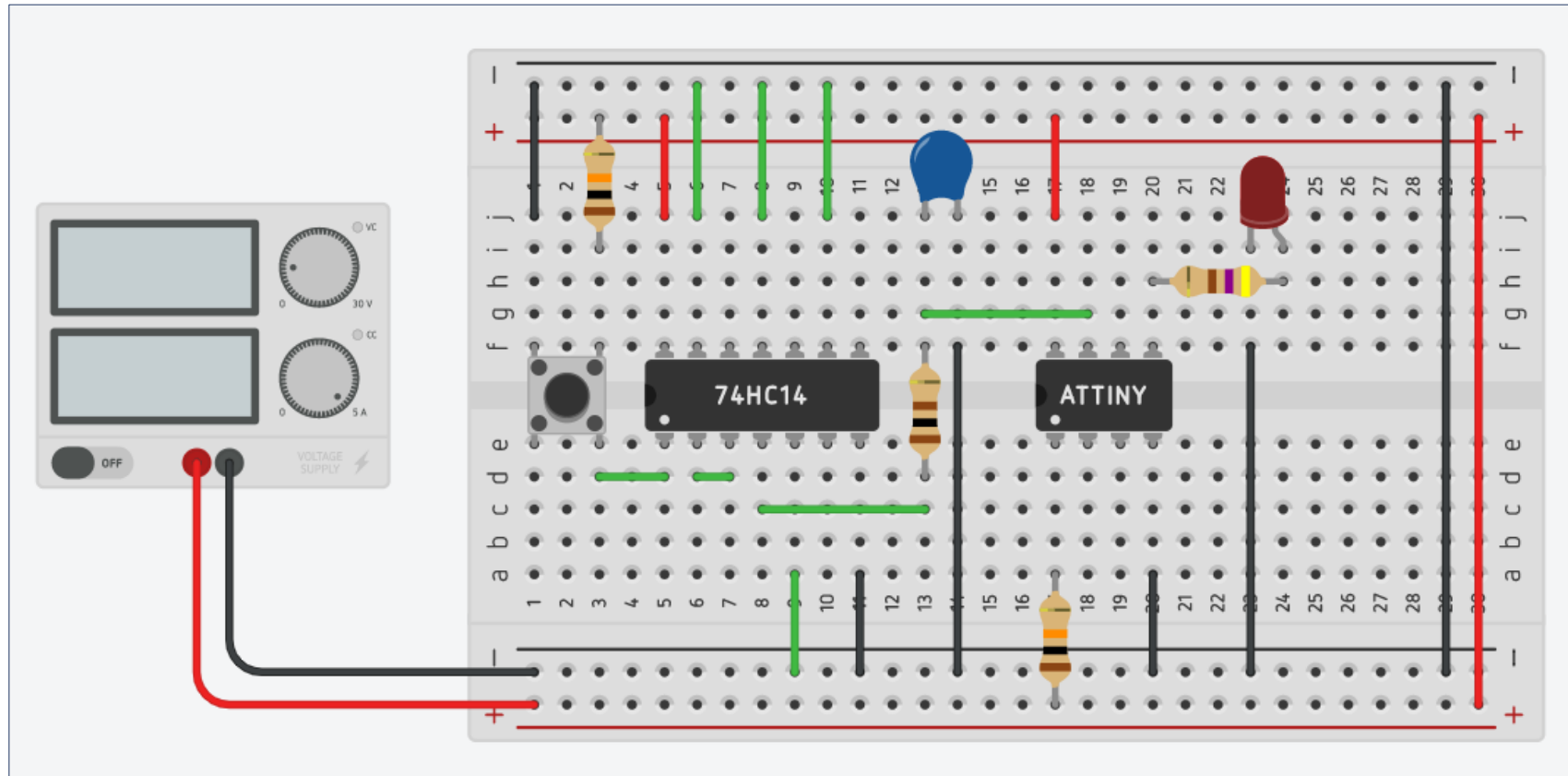
โจทย์ฝึกหัด: จงวาดผังวงจร (**Schematic**) สำหรับวงจรบนบอร์ดข้างล่างนี้



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

ตัวอย่างที่ 3: Push Button - LED

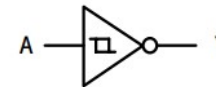
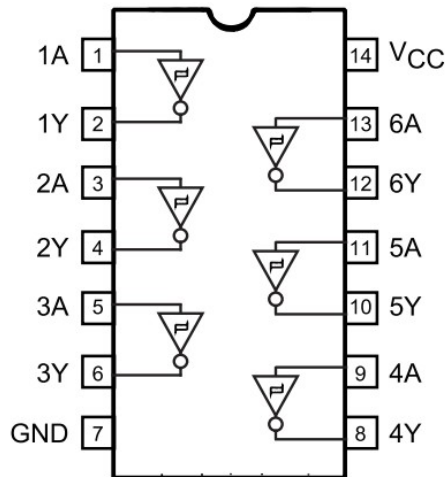
โจทย์ฝึกหัด: จงวาดผังวงจร (Schematic) สำหรับวงจรบนบอร์ดข้างล่างนี้



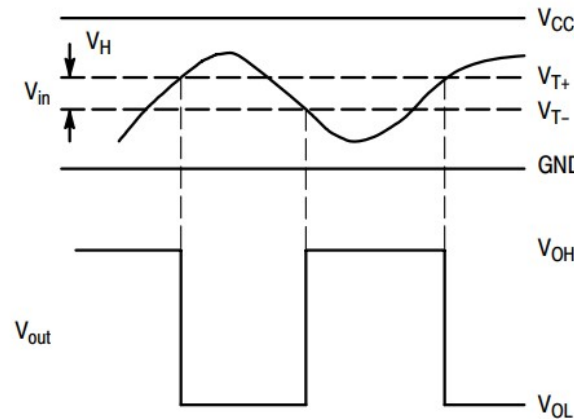
อีกตัวอย่างหนึ่งสำหรับเทคนิคในการลดปัญหาการเกิด **Switch Bounce** คือ การต่อตัวเก็บประจุ ตัวต้านทาน และไอซี **74HC14 (Schmitt-Trigger Inverters)**

เอกสารอ้างอิงสำหรับไอซี 74HC14

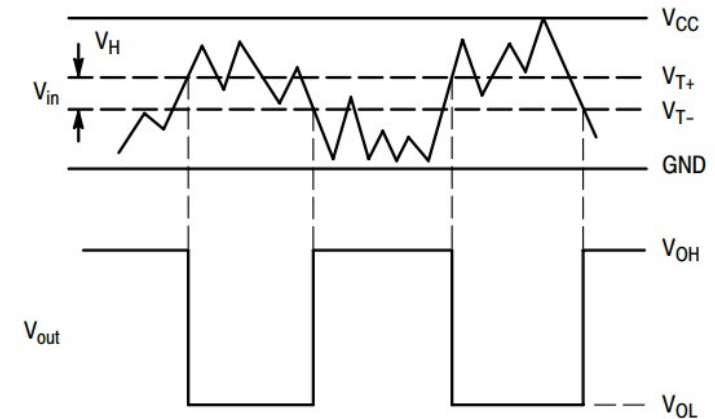
โจทย์ฝึกหัด: จงอธิบายหลักการทำงานของไอซี **74HC14** โดยศึกษาจากเอกสาร **Datasheet** ของผู้ผลิต และอธิบายความหมายของคำว่า **Hysteresis, Threshold Voltages (V_+ และ V_-)**



(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)

เขียนโค้ดเพื่อวนลูปให้อ่านค่าจากอินพุตจากวงจรปุ่มกด (ขา **PB2**) แล้วตรวจสอบสถานะ

ปุ่มกดทำงานแบบ **Active-Low** ดังนั้นในขณะที่กดปุ่ม จะได้ค่าของอินพุตเป็น 0 แต่ถ้าไม่กดปุ่ม จะได้ 1

ถ้ายังกดปุ่มค้างไว้ให้รอจนกว่าจะปล่อยปุ่ม แล้วจึงสลับสถานะของเอาต์พุต (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
  }
}
```

การวนลูปซ้ำเพื่อคอยอ่านค่าอินพุตแล้วทำขั้นตอนตามเงื่อนไข เป็นรูปแบบการทำงานที่เรียกว่า **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**

ไม่มีการวนลูปเพื่ออ่านค่าอินพุตที่ขา **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

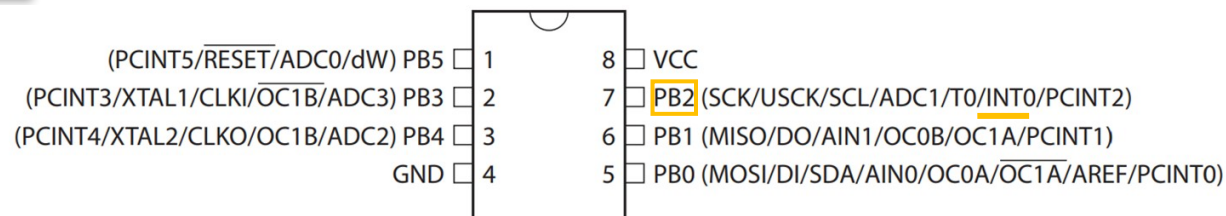
void callback() {
  state = !state; // toggle state
}

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()` จะถูกเรียกเมื่อกดปุ่ม (ลอจิกจะเปลี่ยนค่าจาก **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**

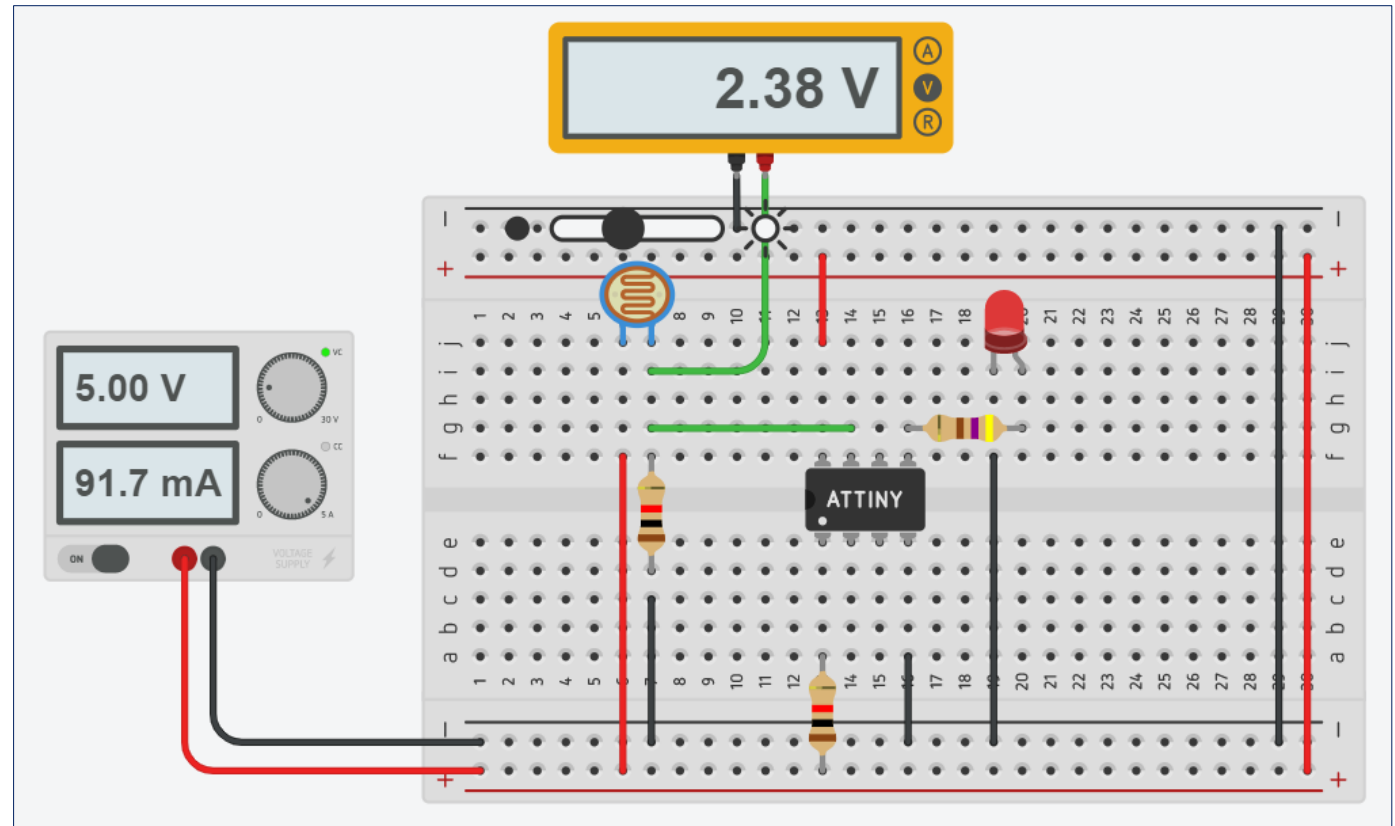
```
void loop() {
  if ( is_btn_pressed ) { // the button was pressed.
    is_btn_pressed = false; // clear flag
    blinking = !blinking; // toggle blink mode
  }
  if ( blinking ) { // LED blinking enabled
    if ( millis() - ts >= 100 ) {
      ts = millis();
      state = !state; // toggle LED state
    }
  } 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**) สำหรับวงจรบนบอร์ดในตัวอย่างที่ 4

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



ตัวอย่างที่ 4: LDR - LED

โจทย์ฝึกหัด: จงต่อวงจรทดลองและทดสอบการทำงานของโค้ดในตัวอย่างที่ 4

โค้ดตัวอย่างนี้ใช้คำสั่ง `analogRead()` เพื่ออ่านค่าอินพุตที่ขา **PB2 / A1** ซึ่งจะได้ค่าในช่วง **0..1023**

ถ้าใช้แรงดันไฟเลี้ยง **5V** หรือ **5000 mV** ก็สามารถคำนวณเพื่อแปลงค่าอินพุตให้เป็นค่าในหน่วยมิลลิโวลต์ (**mV**) ได้

อ่านค่าอินพุตจากขาแอนะล็อกได้มากขึ้น 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 );
}

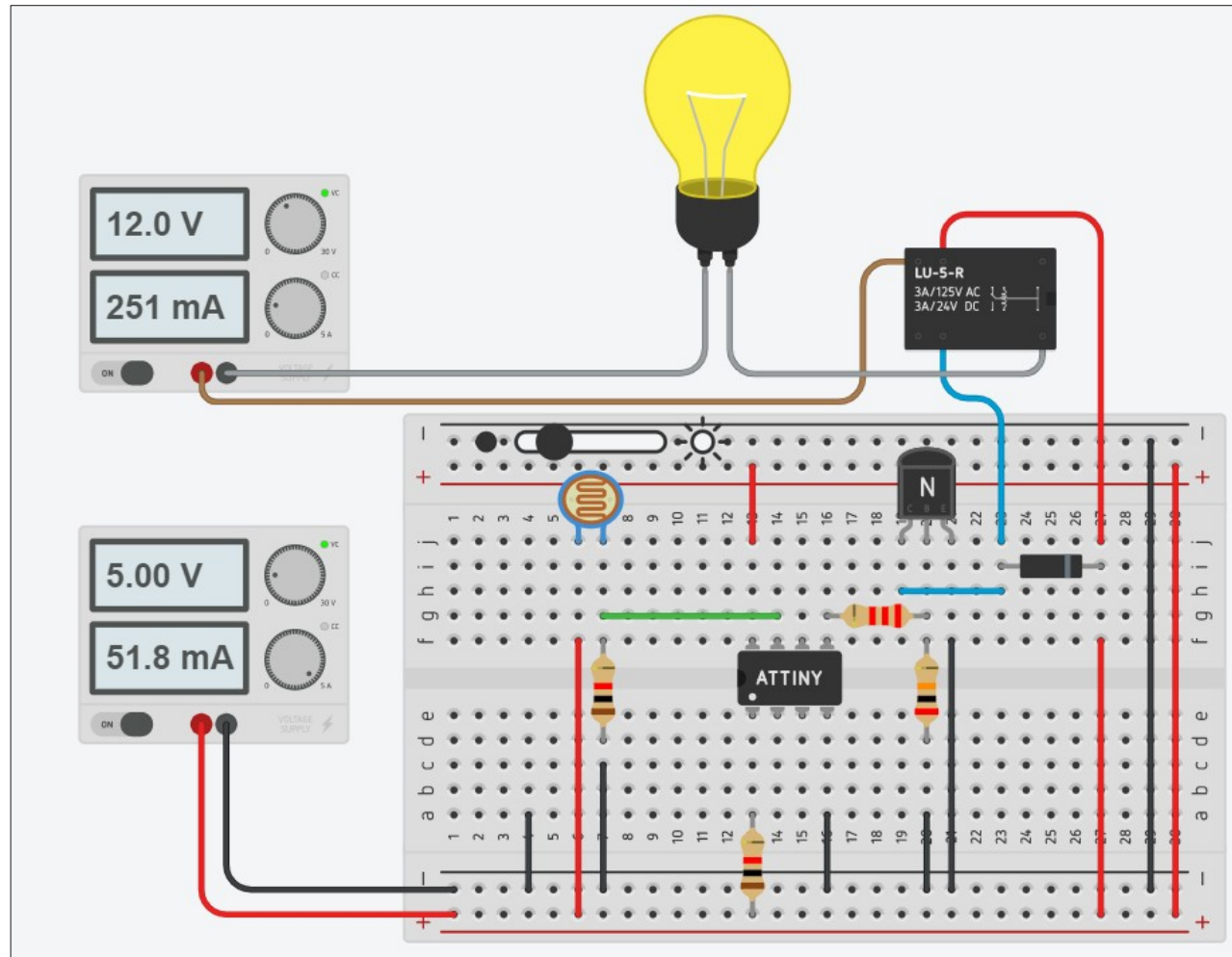
void loop() {
  uint16_t value = analogRead( AIN_PIN );
  uint16_t mV = (value * 5000UL) / 1024;
  if ( state && mV > level_high ) {
    state = false; // turn LED OFF
  }
  else if ( !state && mV < level_low ) {
    state = true; // turn LED ON
  }
  digitalWrite( LED_PIN, state);
}
```

LED จะเปลี่ยนสถานะเป็น **ON** เมื่อค่าอินพุตน้อยกว่า **level_low** และเปลี่ยนสถานะเป็น **OFF** เมื่อค่าอินพุตเพิ่มขึ้นมากกว่า **level_high**

ตัวอย่างที่ 5: LDR – Relay – Light Bulb

โจทย์ฝึกหัด: จงวาดผังวงจรสำหรับวงจรบนบอร์ดบอร์ดในตัวอย่างที่ 5

ตัวอย่างการต่อวงจรเปิด-ปิดหลอดไฟแสงสว่าง โดยอัตโนมัติ เมื่อแสงน้อย ด้วยรีเลย์ (โมเดล **LU-5-R**) ซึ่งใช้แรงดัน **5V** สำหรับคอยล์ (**Coil Voltage**) ใช้ทรานซิสเตอร์ (**NPN**) ควบคุมการทำงานของรีเลย์



ตัวอย่างข้อมูลเชิงเทคนิคสำหรับรีเลย์ LU-5-R

CONTACT RATING

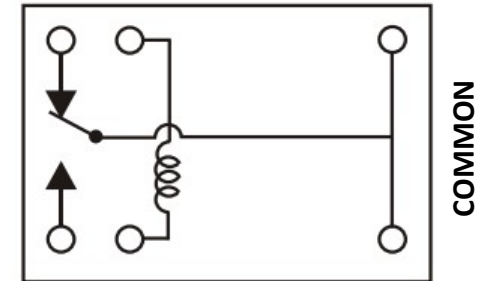
1 Form C (1PDT)	AC 120V	1A
	DC 24V	2A

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.



N.O.

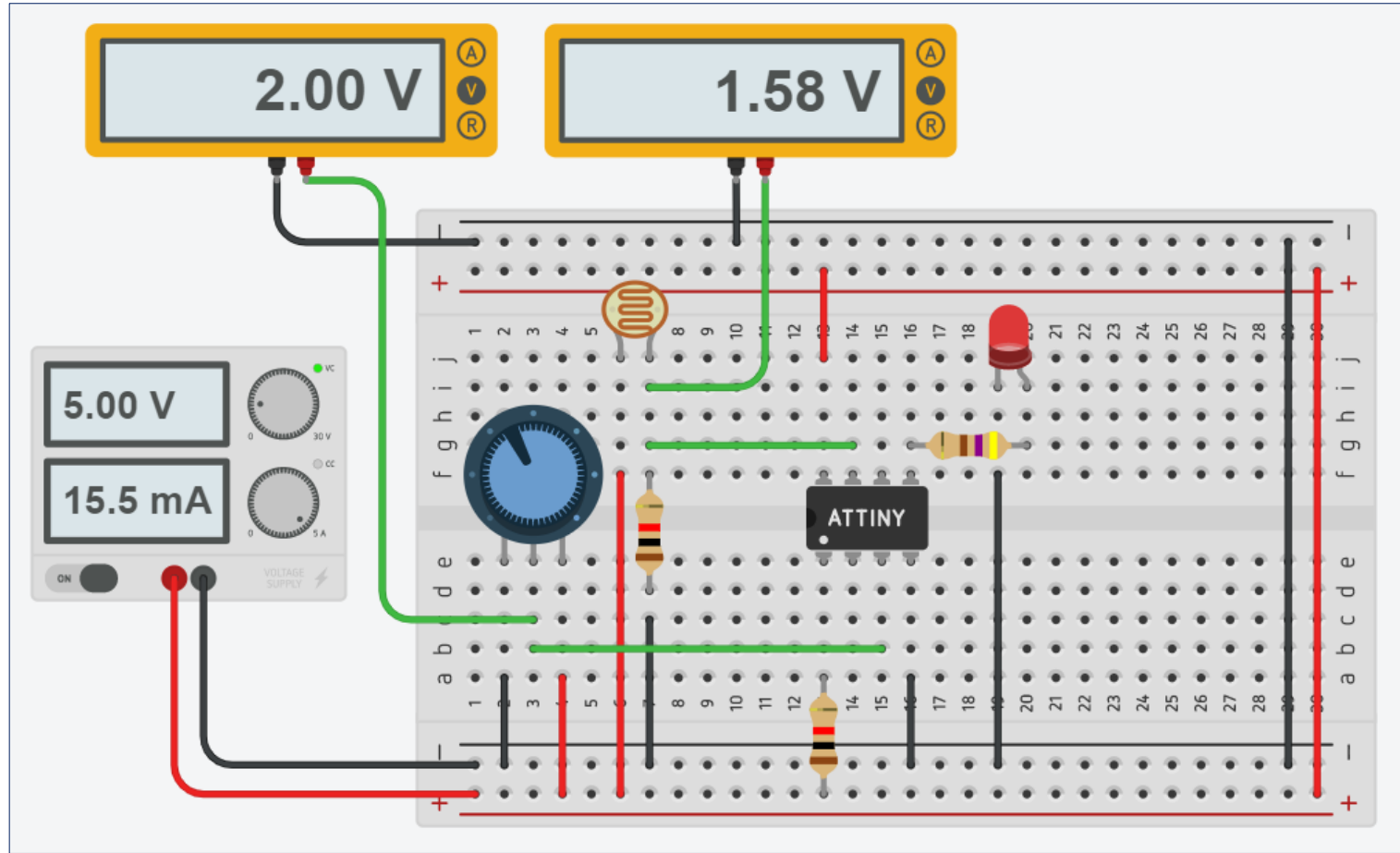
N.C. = Normally Closed
N.O. = Normally Open

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

ตัวอย่างที่ 6: Voltage Level Comparator

โจทย์ฝึกหัด: จงวาดผังวงจรสำหรับวงจรมนเบรตบอร์ดในตัวอย่างที่ 6

ตัวอย่างการอ่าน
ค่าอินพุต-แอนะ
ล็อก 2 ช่อง (A
และ B) นำมา
เปรียบเทียบกัน
แล้วกำหนด
สถานะของ
เอาต์พุต (LED)



ตัวอย่างที่ 6: Voltage Level Comparator

โจทย์ฝึกหัด: จงต่อวงจรทดลองและทดสอบการทำงานของโค้ดในตัวอย่างที่ 6

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

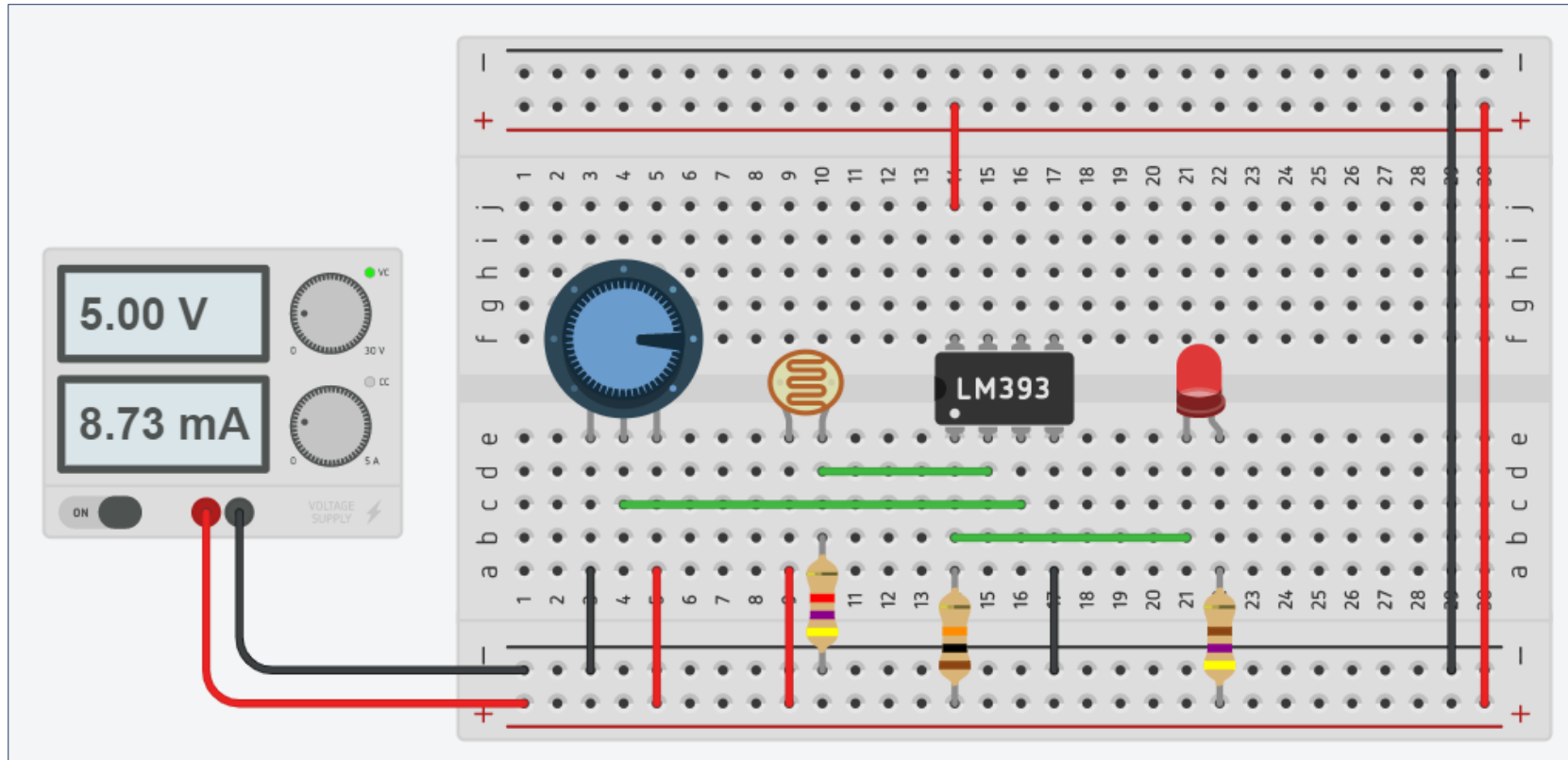
```
const int LED_PIN = PB0; // LED output pin
const int A_PIN = A1; // analog input A, use ADC1 pin
const int B_PIN = A2; // analog input B, use ADC2 pin

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: จงวาดผังวงจรสำหรับวงจรบนบอร์ดตามรูปตัวอย่างข้างล่างนี้

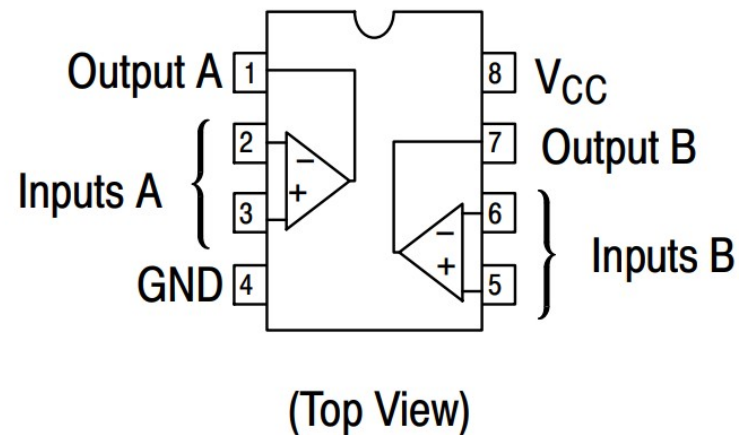


ในตัวอย่างที่ 6 เป็นการใช้อะตินาย **ATTiny85** แล้วเขียนโค้ดให้ทำหน้าที่อ่านค่าแรงดันแอนะล็อก 2 ช่อง แล้วเปรียบเทียบกัน เพื่อกำหนดสถานะเอาต์พุตให้ **LED** ตัวอย่างนี้สาธิตการใช้ไอซี **LM393** ซึ่งเป็นไอซีเปรียบเทียบแรงดัน (**Voltage Comparator**) แทนการใช้ **ATTiny85**

ตำแหน่งขาของ LM393

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

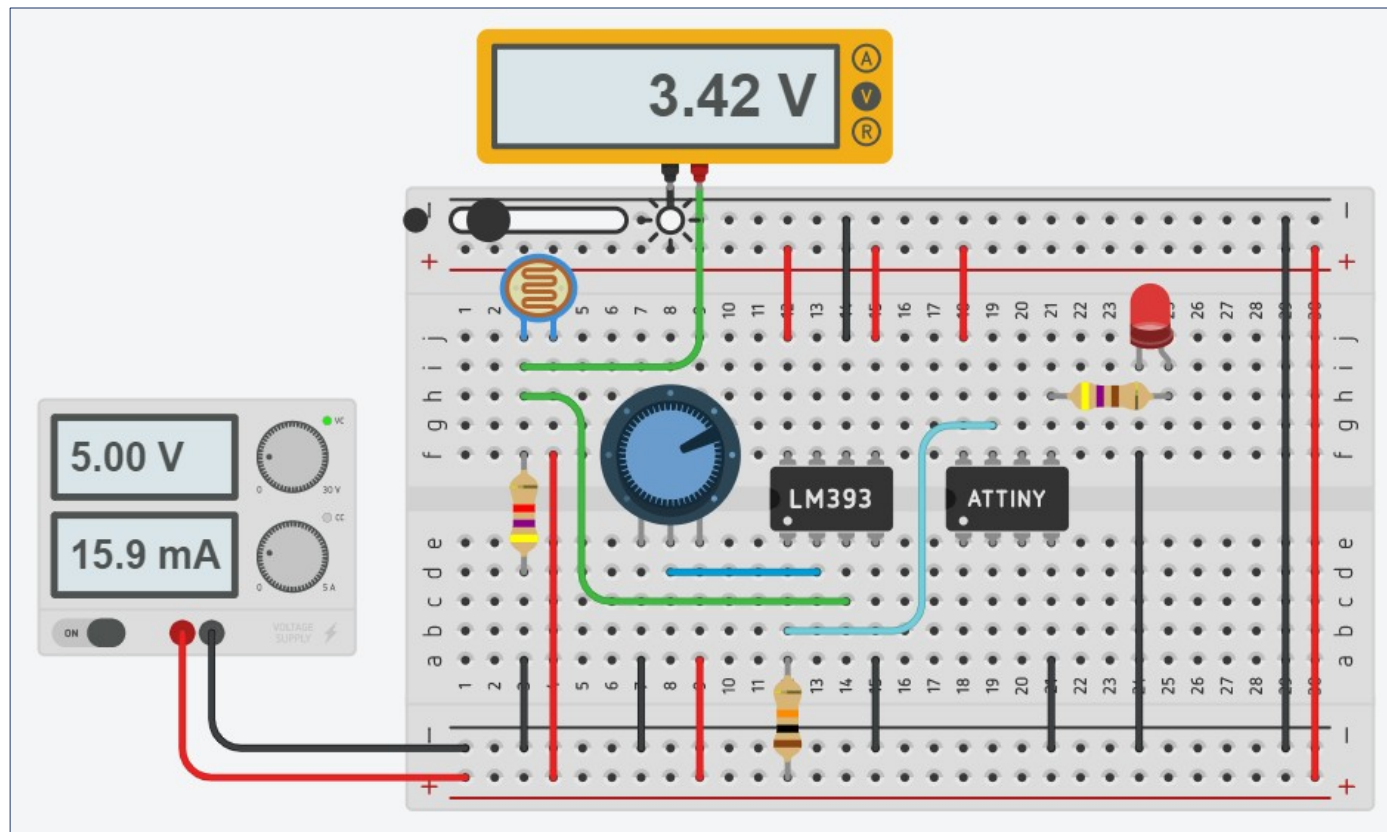
PIN CONNECTIONS



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

ตัวอย่างที่ 7: LDR – LM393 – LED

โจทย์ฝึกหัด: จงวาดผังวงจรสำหรับวงจรมอเตอร์ในตัวอย่างที่ 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);
}
```

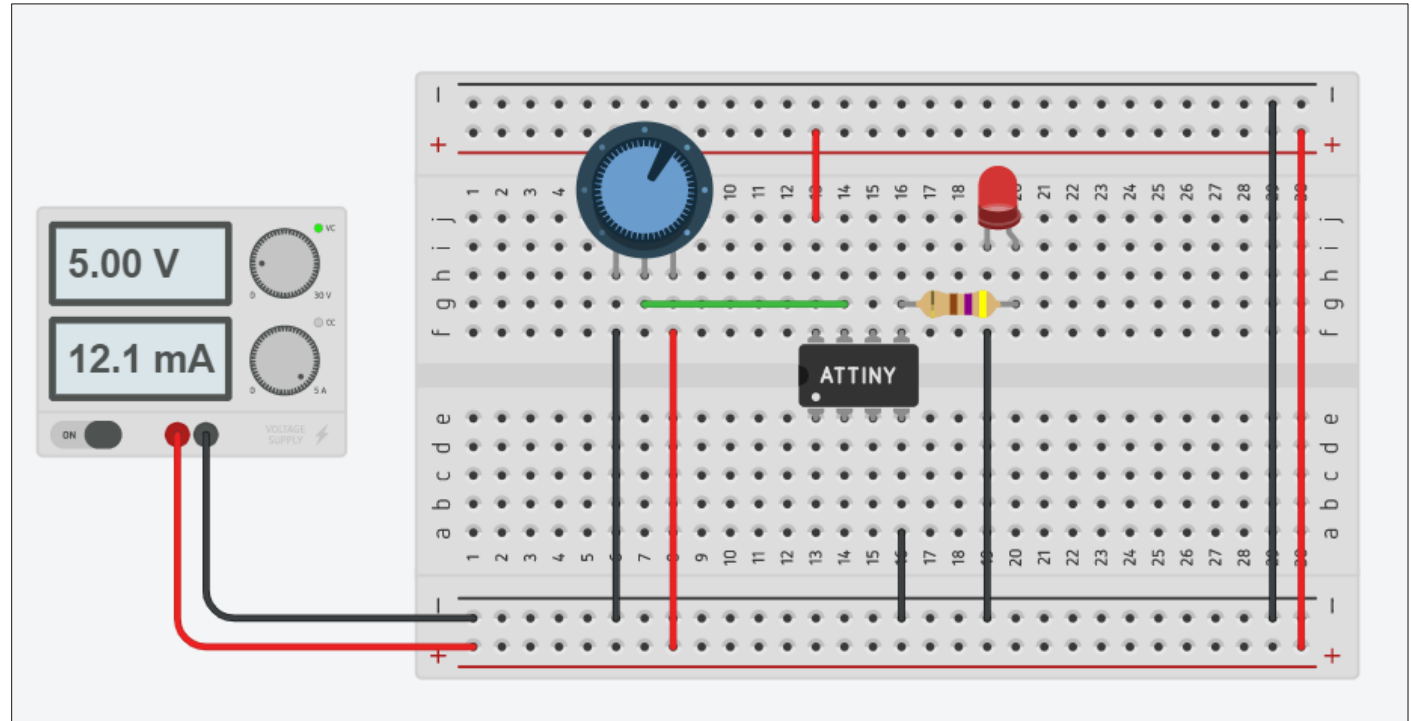
สัญญาณเอาต์พุตจาก **LM393** จะถูกใช้เป็นสัญญาณอินพุต-ดิจิทัล ที่ขา **PB2** ซึ่งตรงกับอินเทอร์รัพท์หมายเลข **INT0**

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

ตัวอย่างที่ 8: Trimpot - LED Dimming

โจทย์ฝึกหัด: จงวาดผังวงจรสำหรับวงจรบนเบรตบอร์ดในตัวอย่างที่ 8

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



Trimpot = Trimmer Potentiometer

ตัวอย่างที่ 8: Trimpot - LED Dimming

โจทย์ฝึกหัด: จงต่อวงจรทดลองและทดสอบการทำงานของโค้ดในตัวอย่างที่ 8

คำสั่ง `analogWrite()` ใช้สำหรับสร้างสัญญาณเอาต์พุตแบบ **PWM (Pulse Width Modulation)** ซึ่งเป็นสัญญาณแบบมีคาบ แต่ปรับช่วงกว้างที่เป็นลอจิก **1 (High)** เทียบกับคาบของสัญญาณ โดยคิดเป็นเปอร์เซ็นต์: **0=0%** ถึง **255=100%**

คำสั่ง `analogRead()` อ่านได้ค่าในช่วง **0..1023** ถ้านำค่าที่ได้ไปใช้กับคำสั่ง จะต้องหารด้วย **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
}

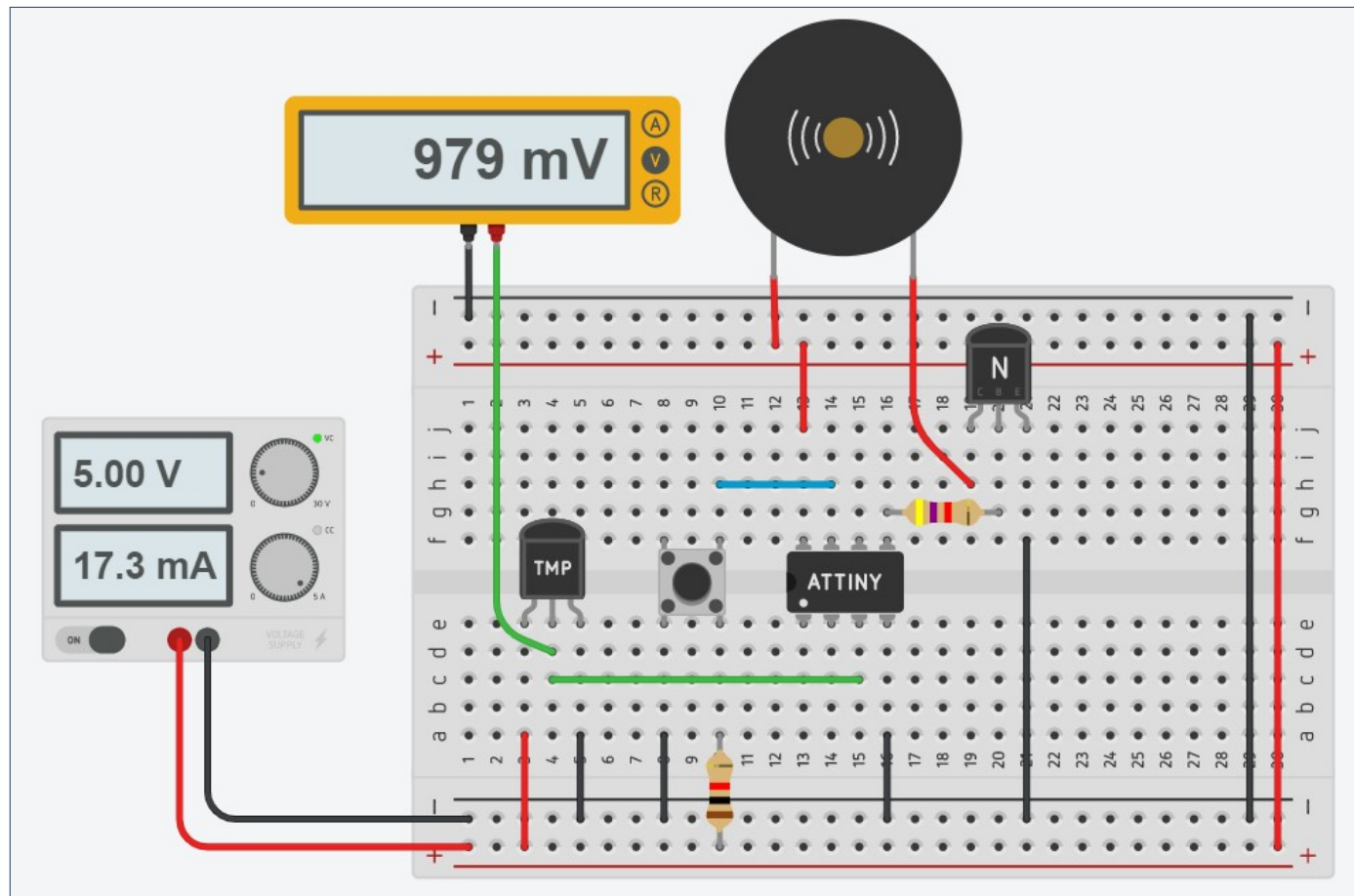
void loop() {
  int new_value = analogRead( AIN_PIN )/4;
  if ( new_value != value ) { // input value changed ?
    value = new_value; // update value
    analogWrite( LED_PIN, value ); // update PWM
    delay(100);
  }
}
```


ตัวอย่างที่ 9: TMP36 - Buzzer

โจทย์ฝึกหัด: จงวาดผังวงจรสำหรับวงจรมนเบรตบอร์ดในตัวอย่างที่ 9

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

วงจรใช้ทรานซิสเตอร์
แบบ **NPN** เปิด-ปิด
การทำงานของบัส
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 );
}
```

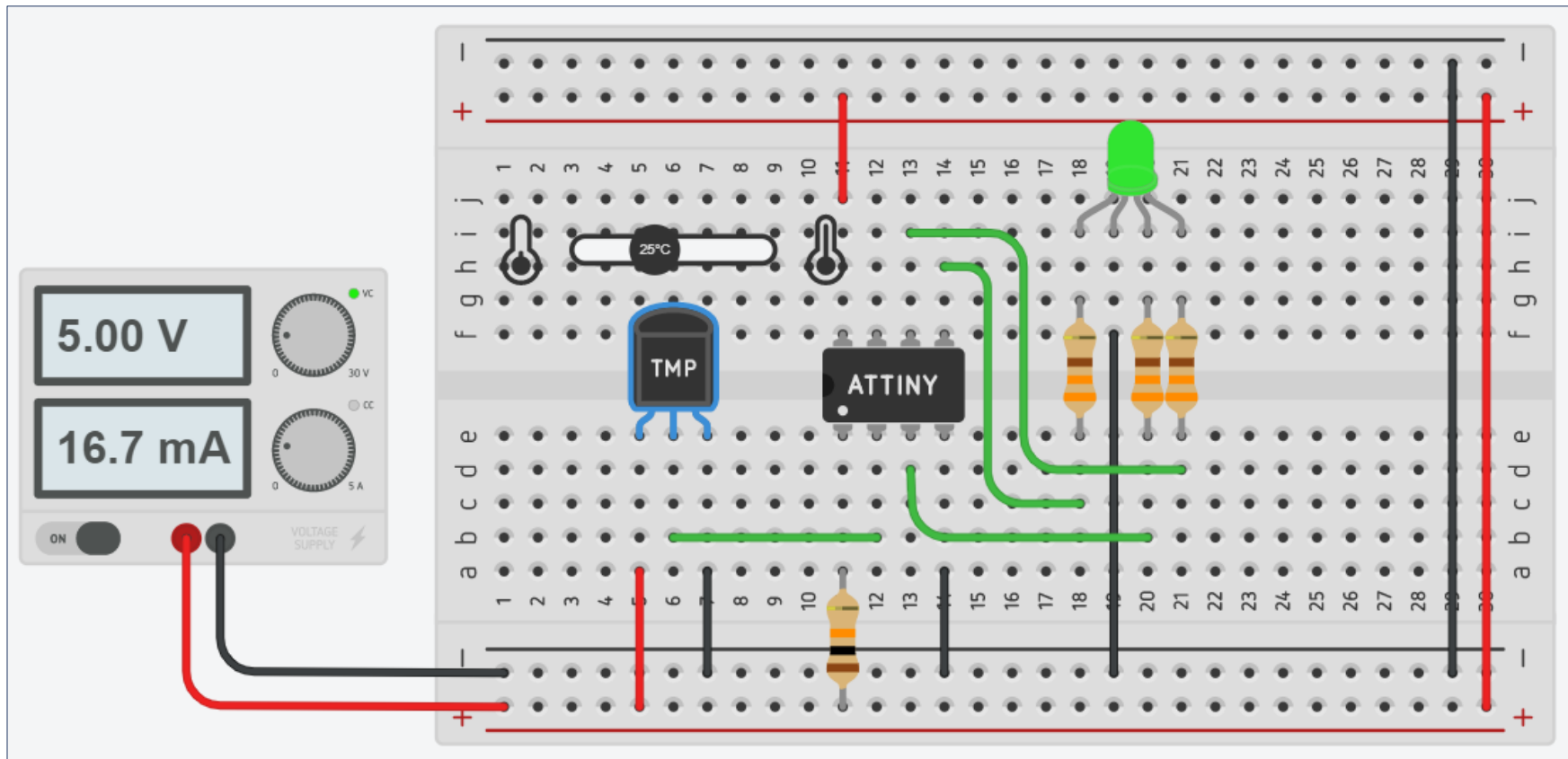
```
// TMP36: 0V @ -50°C, 1.75V @ 125°C
// Temp [deg°C] = (voltage [mV] - 500)/10

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** จะทำให้บัชเซอร์เสียง (**Sound Buzzer**) มีเสียงดัง เพื่อเป็นการแจ้งเตือน และสามารถกดปุ่มเพื่อเปิดหรือปิดการแจ้งเตือนด้วยเสียงได้

ตัวอย่างที่ 10: TMP36 – RGB LED

โจทย์ฝึกหัด: จงวาดผังวงจรสำหรับวงจรบนบอร์ดในตัวอย่างที่ 10



ในตัวอย่างนี้ สาธิตการวัดอุณหภูมิด้วยไอซี **TMP36** และใช้ **RGB LED (Common Cathode)** แสดงช่วงของค่าที่วัดได้ เช่น สีน้ำเงินสำหรับค่าที่ต่ำกว่า 20 องศา หรือ สีแดงสำหรับค่าที่สูงกว่า 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 );
    }
}
```

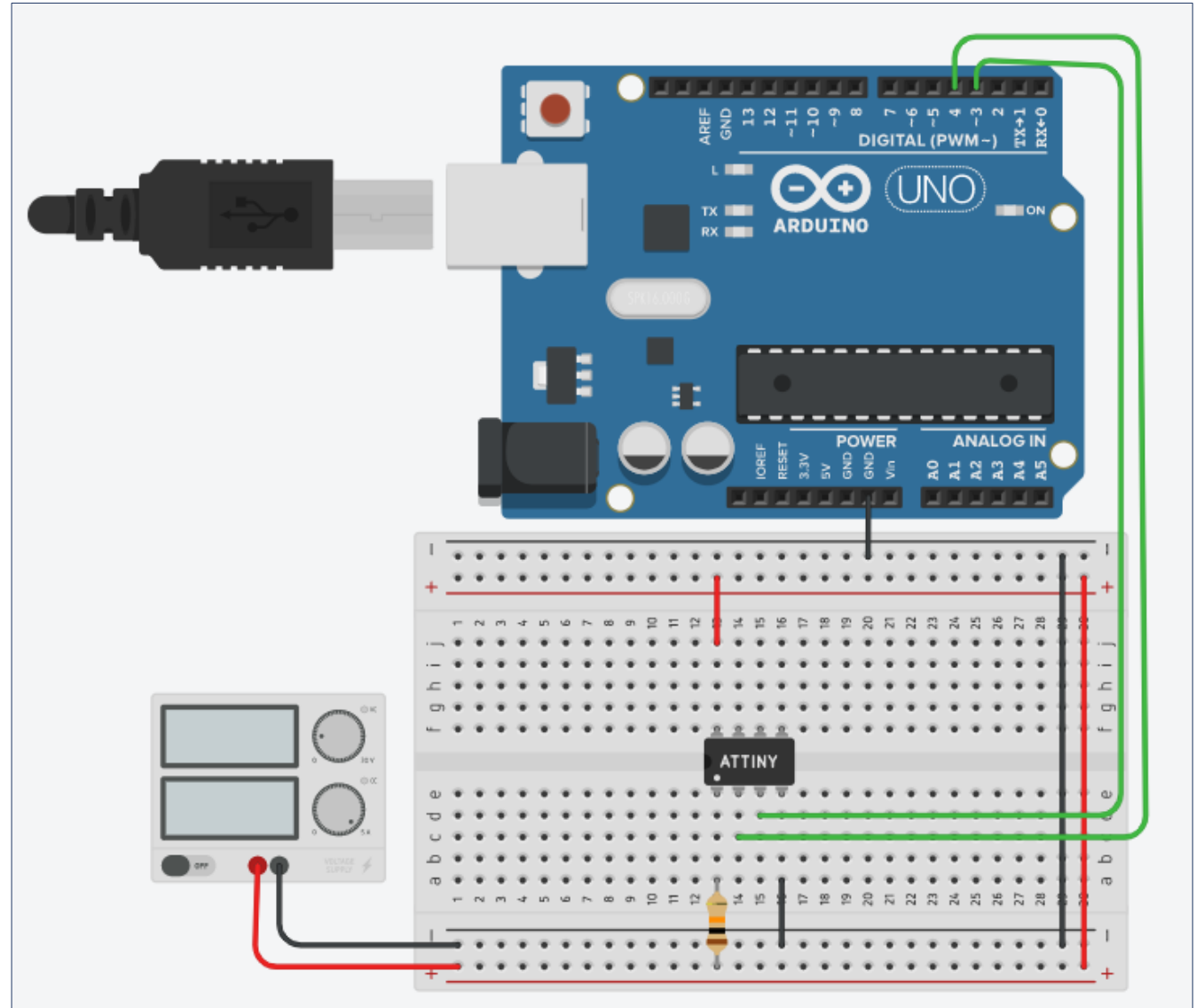
```
void loop() {
    // read analog input from TMP36 sensor
    int value = analogRead( TMP36_PIN );
    int mV = (value * 5000L) / 1024;
    int temp = (mV - 500) / 10; // value in deg.C
    memset( rgb.v, 0x00, 3 );
    if (temp < 10) { // below 10 deg.C
        rgb.s.b = 1; // blue color
    } else 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

ตัวอย่างสาธิตการ
เชื่อมต่อเพื่อสื่อสาร
ข้อมูลแบบบิตอนุกรม
(Serial) ระหว่าง
บอร์ด **Arduino
Uno** และ **ATtiny85**
บนเบอร์ดบอร์ด

ขา **3** และ **4** ใช้
สำหรับขา **Rx** และ
Tx ของ **SoftSerial**
ตามลำดับ

การต่อสายระหว่างขา
Rx และ **Tx** ของทั้ง
สองบอร์ด จะต้องไขว้
สายกัน (**Rx -> Tx**
และ **Tx <- Rx**)



ตัวอย่างที่ 11: Uno – Attiny85 Serial Link

The image shows a screenshot of an IDE (likely TinkerCAD) displaying a simulation of an Arduino Uno connected to an ATtiny85 via a breadboard. The ATtiny85 is powered by a 5.00V source and has a current of 8.00 mA. The ATtiny85 is connected to the Arduino Uno's digital pins 3 and 4, which are configured as RX and TX respectively. The code on the right uses SoftwareSerial to communicate with the ATtiny85. The Serial Monitor shows the output 'cnt: 001' through 'cnt: 008'.

Simulator time: 00:00:01.773

All changes saved

Code Stop Simulation Export Share

2 (Arduino Uno R3)

```
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 }
24
```

Serial Monitor

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 ทำหน้าที่คอยรับข้อมูลจาก **Attiny85** ด้วยวิธี **SoftSerial** (ตั้งค่า **baud 4800**) และเมื่อได้รับข้อความใด ๆ 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);
  }
}
```

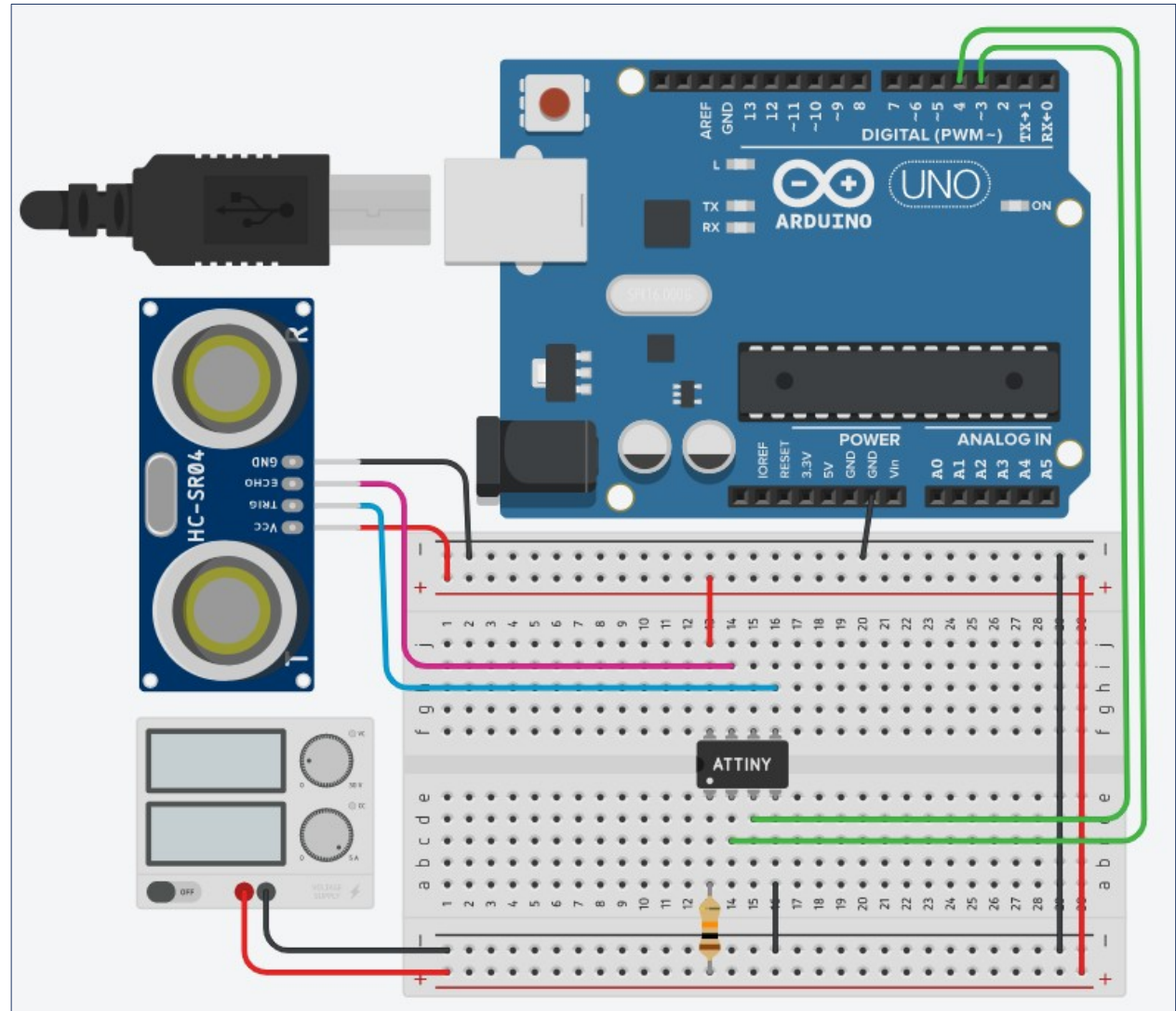
Uno

ตัวอย่างที่ 12: Ultrasonic Distance Sensor

โจทย์ฝึกหัด: ให้ศึกษาการทำงานของโมดูล **HC-SR04** จากเอกสารผู้ผลิต

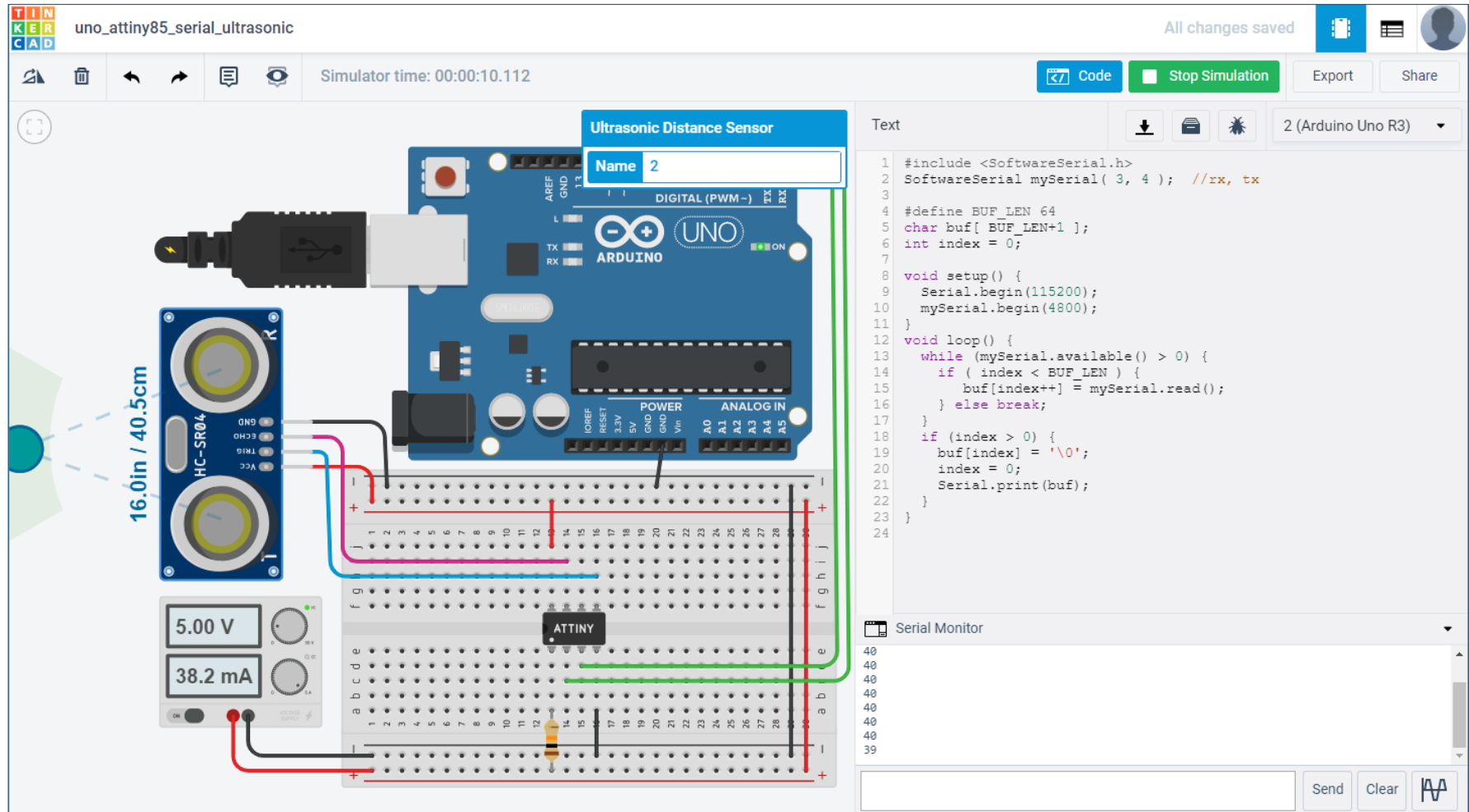
ตัวอย่างนี้สาธิตการอ่านค่าระยะห่างจากวัตถุด้วยโมดูล **HC-SR04 (Ultrasonic Distance Sensor)** ซึ่งมีขา **TRIGGER** และ **ECHO**

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



ตัวอย่างที่ 12: Ultrasonic Distance Sensor

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



The image shows a TinkerCAD simulation of an Arduino Uno R3 board connected to an Attiny85 microcontroller and an HC-SR04 ultrasonic distance sensor. The sensor is connected to the Arduino's digital pins (TX, RX, and GND). The Attiny85 is connected to the Arduino's digital pins (TX, RX, and GND). A 5.00 V power source is connected to the sensor's VCC pin. The sensor's range is indicated as 16.0in / 40.5cm. The code in the Serial Monitor shows the sensor's output being read and printed.

```
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 }
24
```

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);
  }
}
```

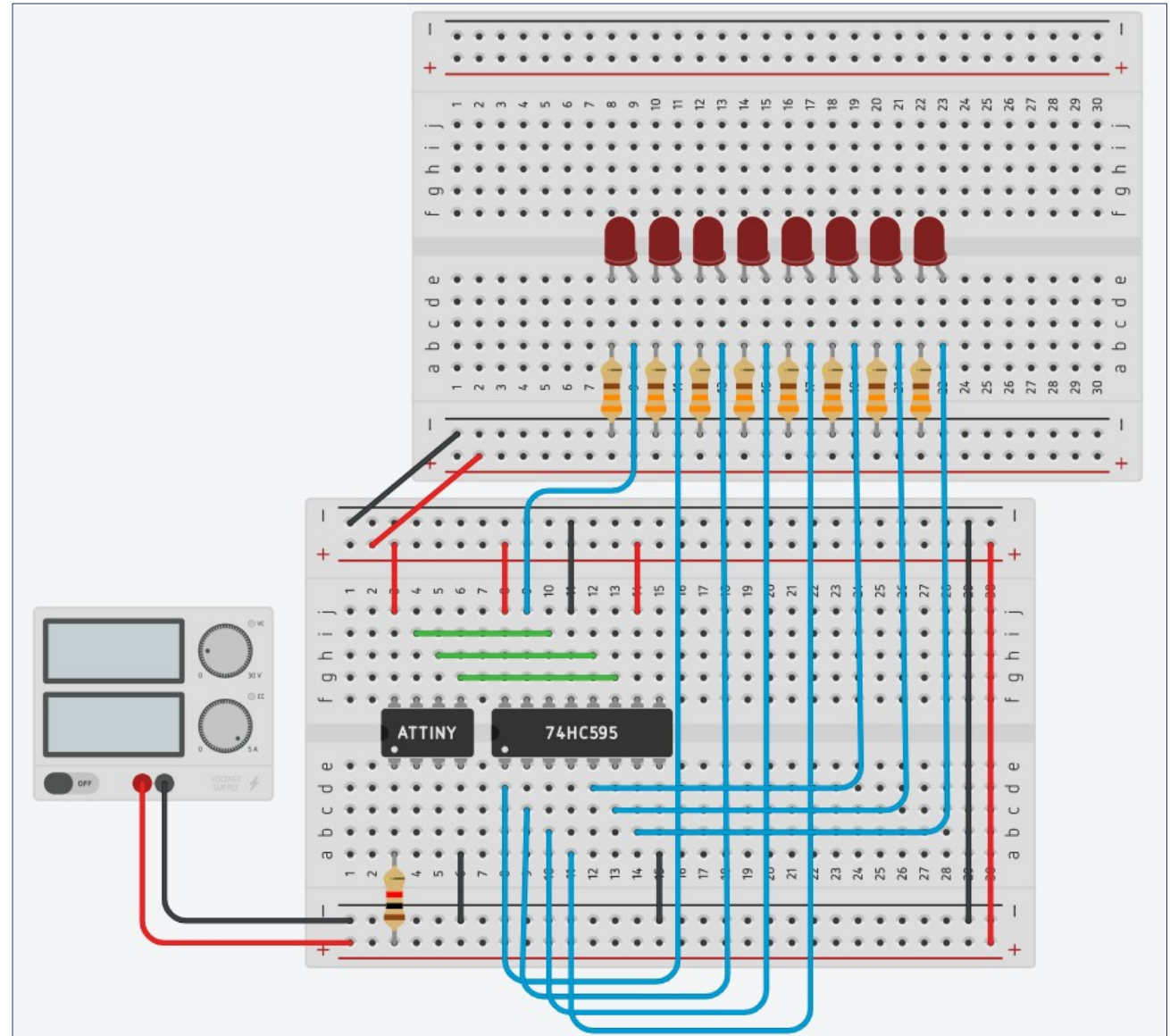
Uno

ข้อสังเกต: คำสั่ง `pulseIn()` ของ **Arduino API** ใช้สำหรับวัดความกว้างของสัญญาณพัลส์

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** โดยใช้
สัญญาณ

- 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** สำหรับเลื่อนข้อมูลขนาดหนึ่งไบต์ที่ละบิตออกไป โดยกำหนดลำดับของการเลื่อนบิต (**Bit Order**) คือ ให้บิต **MSB** ออกไปก่อน (**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

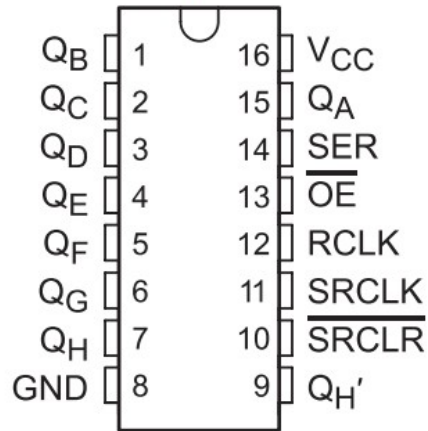
byte data = 0x01; // data byte

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 );
}

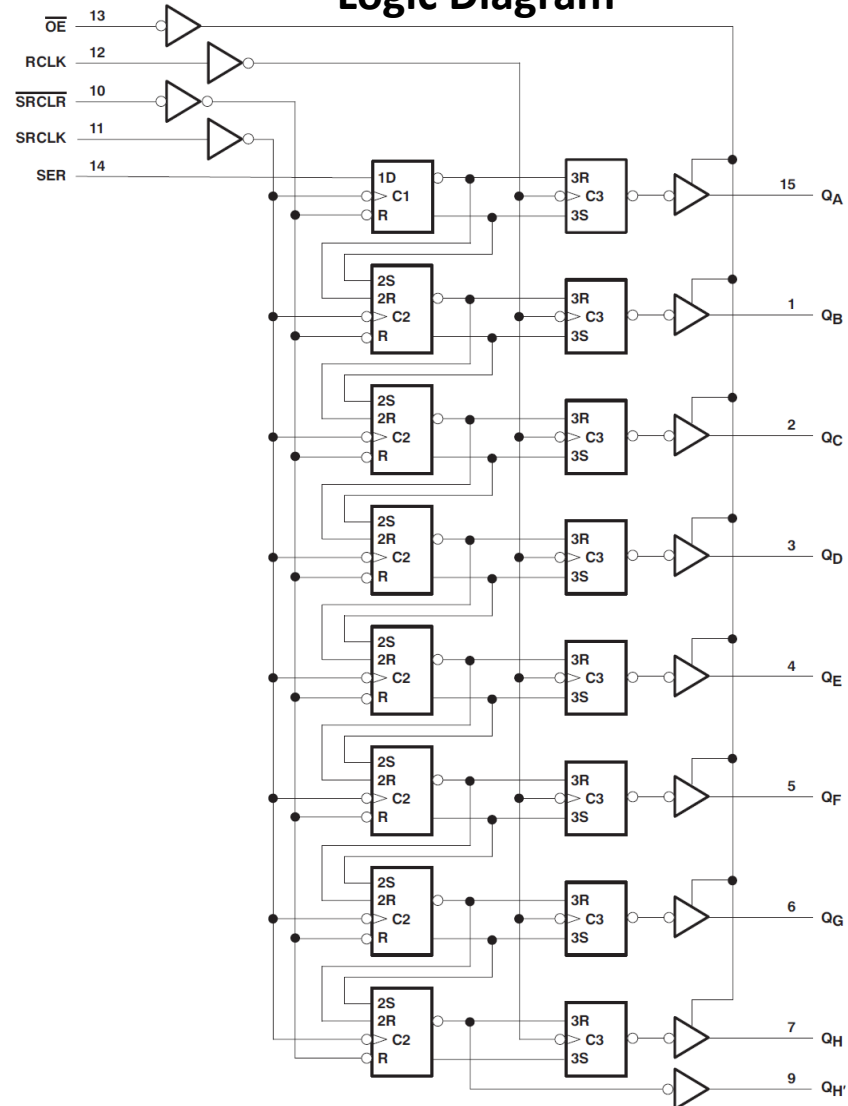
void loop() {
  shiftOut( DS_PIN, SH_CP_PIN, MSBFIRST, data );
  digitalWrite( ST_CP_PIN, HIGH );
  digitalWrite( ST_CP_PIN, LOW );
  // rotate-shift-left by 1-bit position
  data = (data << 1) | (data >> 7);
  delay(200);
}
```

เอกสารอ้างอิงเกี่ยวกับ 74HC595

Pin Assignments

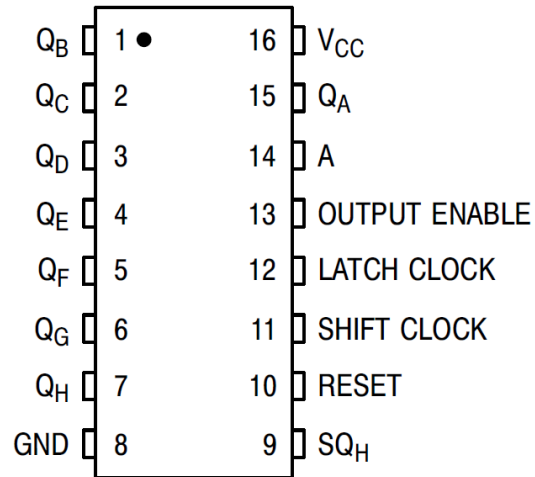


Logic Diagram



เอกสารอ้างอิงเกี่ยวกับ 74HC595

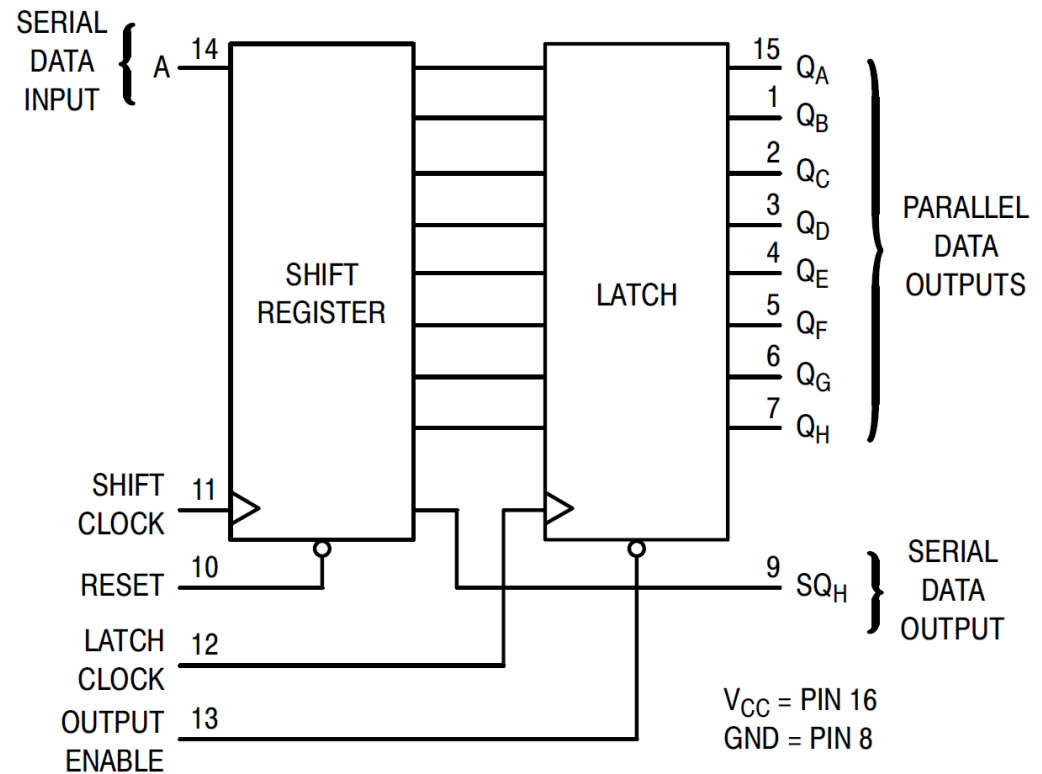
PIN ASSIGNMENT



SHIFT CLOCK = Shift Register Clock

LATCH CLOCK = Storage Register Clock

LOGIC DIAGRAM



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)

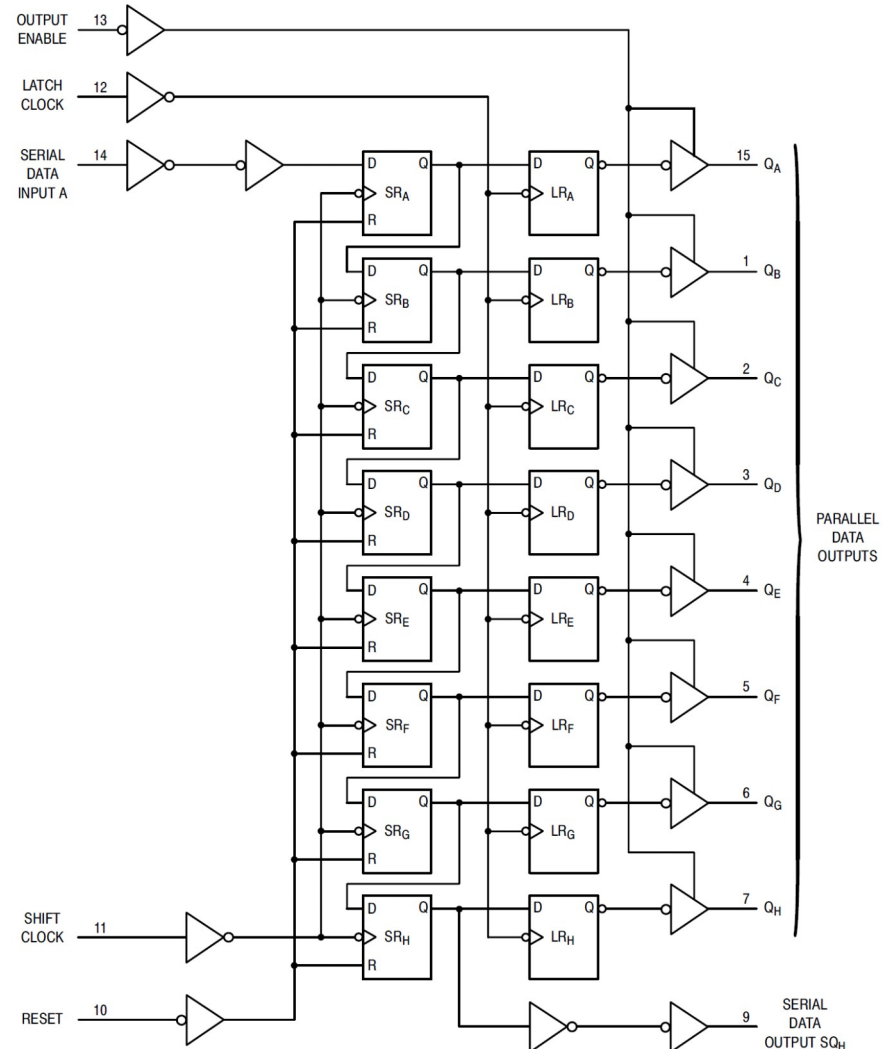
Noninverted, 3-state, latch outputs.

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.

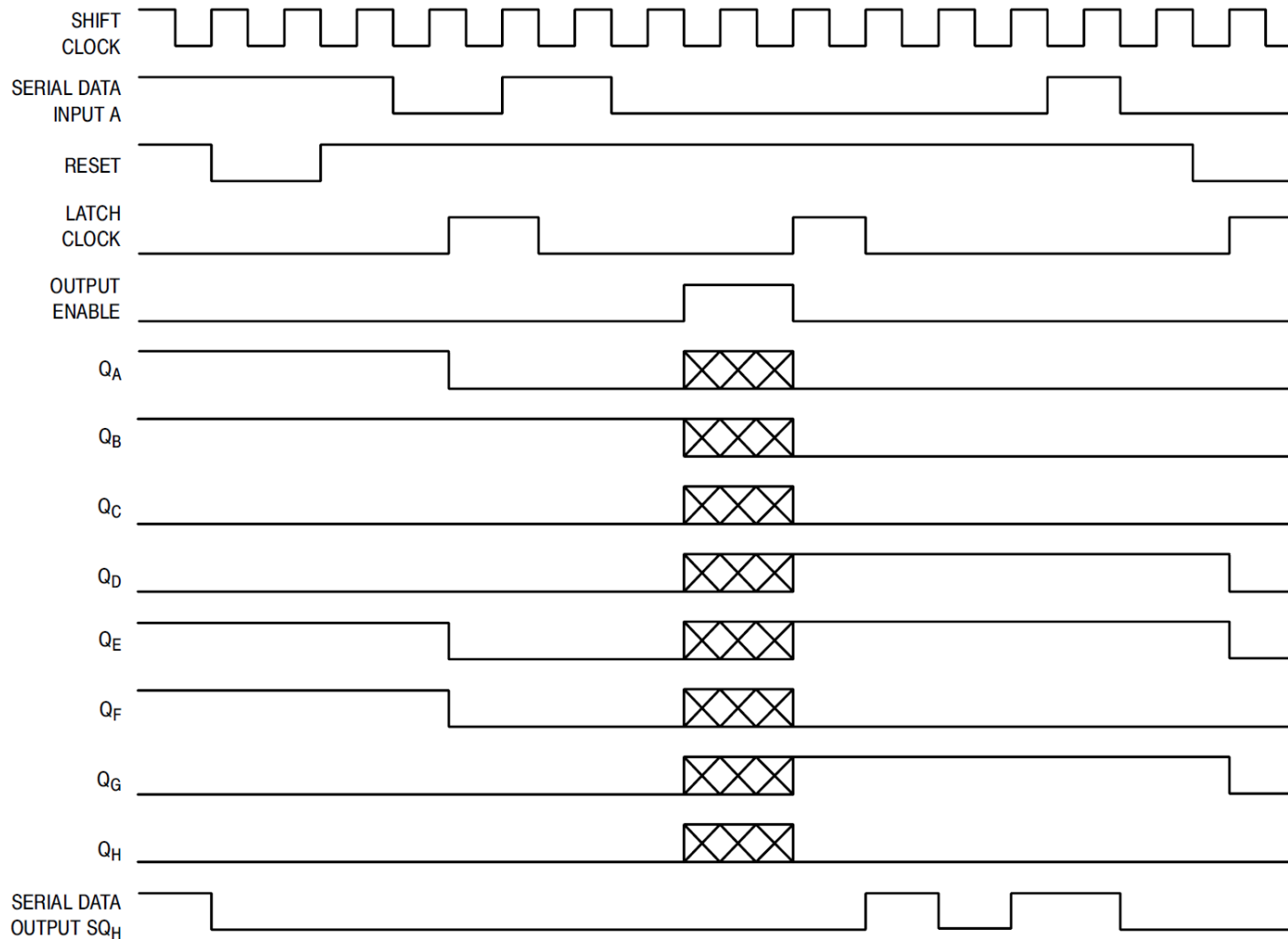
เอกสารอ้างอิงเกี่ยวกับ 74HC595

Logic Diagram



เอกสารอ้างอิงเกี่ยวกับ 74HC595

Timing Diagram



NOTE: **XXXX** 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