

Embedded Systems Programming on STM32 MCU

การโปรแกรมระบบสมองกลฝังตัวบน


ไมโครคอนโทรลเลอร์ STM32

ครั้งที่ 4 : การมอดูเลตเชิงความกว้างพัลส์

 Pulse Width Modulation (PWM) การมอดูเลตเชิงความกว้างพัลส์

 ความหมายและลักษณะของ PWM

 การสร้างสัญญาณ PWM


 วิธีทางฮาร์ดแวร์ (และที่รองรับโดย STM32)

 วิธีทางซอฟต์แวร์

 การประยุกต์ใช้งาน PWM

 ควบคุมความสว่าง LED

 ควบคุมความเร็ว DC motor

 ควบคุมแกนของเซอร์โว



Asst.Prof. Thanwa SRIPRAMONG
PRESENTER

TODAY TOPIC IS
PWM

การมอดูเลตเชิงความกว้างพัลส์ (PWM)

🌱 เป็นการผสมสัญญาณระหว่างสัญญาณแอนะล็อก กับ สัญญาณดิจิทัลพาหะ

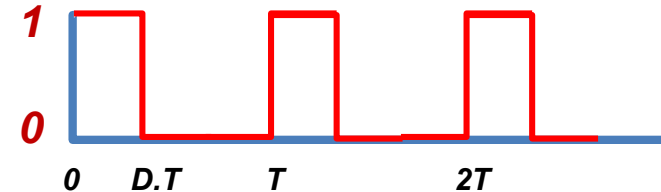
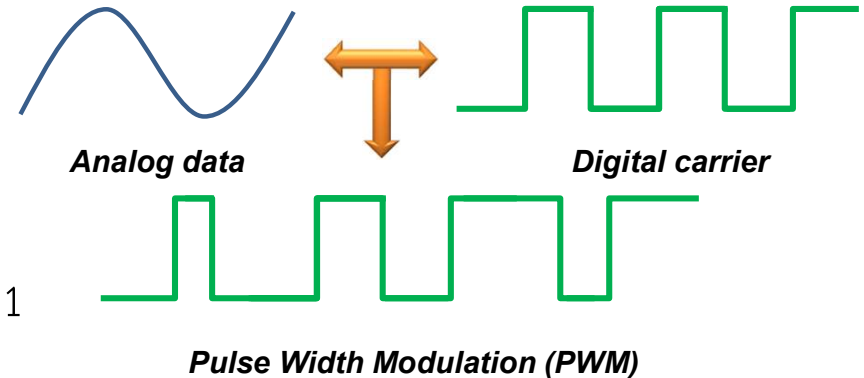
💡 เมื่อระดับสัญญาณแอนะล็อกมีค่าสูง จะทำให้สัดส่วนของ คาบเวลาลอจิก 1 มีมากขึ้นเมื่อเทียบกับลอจิก 0 และเมื่อระดับ สัญญาณแอนะล็อกมีค่าต่ำลง ทำให้สัดส่วนของคาบเวลาลอจิก 1 ลดลงเมื่อเทียบกับลอจิก 0

💡 หากพิจารณาในช่วงเวลาแคบๆ ที่คลุมคาบเวลาของพัลส์บวกและ ลบสัญญาณดิจิทัล (ครบหนึ่งรอบความถี่) ค่าระดับสัญญาณแอนะล็อกจะแปรผันตรงกับพื้นที่ใต้คาบเวลาของพัลส์บวก

🌈 สัดส่วนคาบเวลาช่วงพัลส์บวกเทียบกับช่วงพัลส์ลบเรียกว่า duty cycle

🌈 duty cycle 0% เท่ากับช่วงพัลส์ 1 มีคาบเวลาเป็น 0 แทนค่าแอนะล็อกที่มีค่าต่ำ ที่สุด (ในพิสัยที่กำหนด)

🌈 duty cycle 100% เท่ากับช่วงพัลส์ 1 มีคาบเวลาเท่ากับรอบความถี่ (ไม่มีระดับ ลอจิก 0 เลย) แทนค่าแอนะล็อกที่มีค่าสูงที่สุด



$$\bar{y} = \frac{1}{T} \int_0^T f(t) dt$$

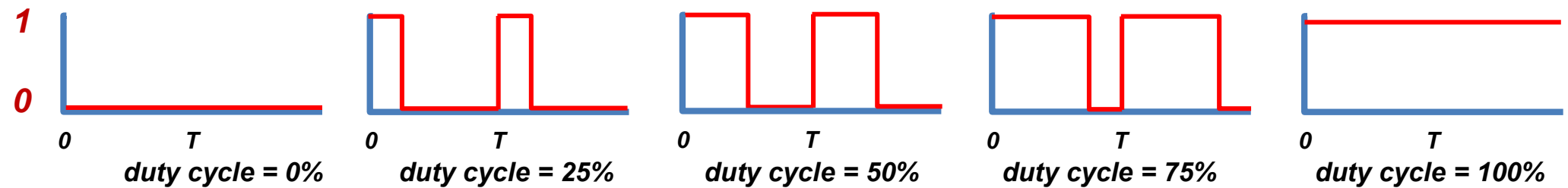
$$\bar{y} = D \cdot (1) + (1 - D)(0)$$



Asst.Prof. Thanwa SRIPRAMONG
PRESENTER

TODAY TOPIC IS
PWM

การมอดูเลตเชิงความกว้างพัลส์ (PWM)



Asst.Prof. Thanwa SRIPRAMONG

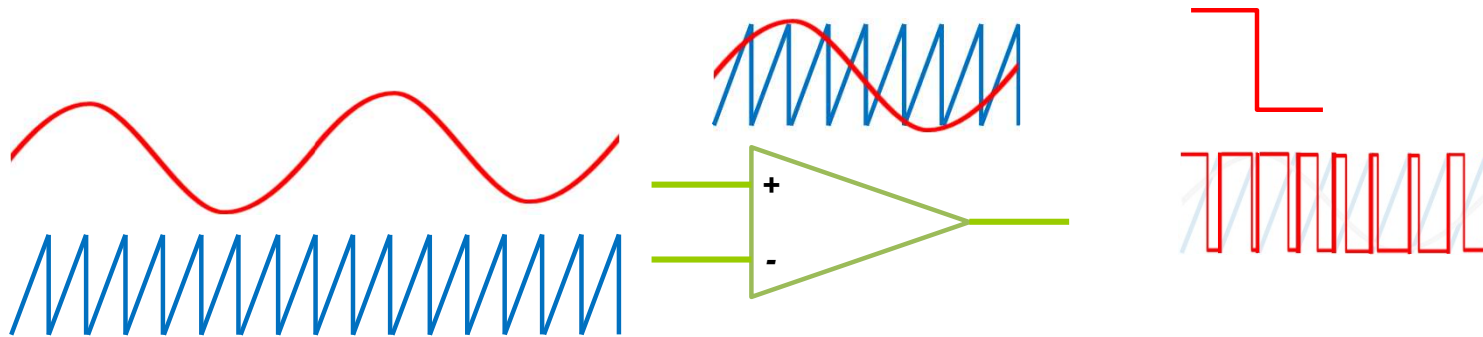
PRESENTER

TODAY TOPIC IS
PWM

หลักการสร้างสัญญาณ PWM ด้วยวิธีต่างๆ

🌱 การใช้วงจรสัญญาณ sawtooth กับวงจรเปรียบเทียบสัญญาณ (comparator)

💡 สร้างสัญญาณ sawtooth ที่มีความถี่ของสัญญาณสูงเพียงพอต่อการใช้งาน และนำสัญญาณแอนะล็อกที่ต้องการมอดูเลต มาผ่านวงจรเปรียบเทียบสัญญาณ โดยช่วงเวลาที่สัญญาณแอนะล็อกมีค่าสูงกว่าจะให้ลอจิกขาออกเป็น 1 ส่วนช่วงที่สัญญาณแอนะล็อกมีค่าต่ำกว่าจะให้ลอจิกขาออกเป็น 0



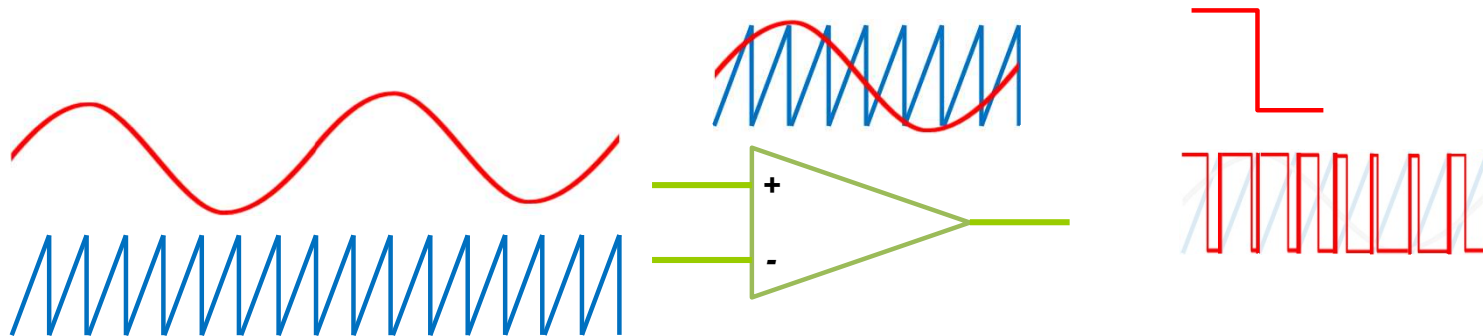
Asst.Prof. Thanwa SRIPRAMONG
PRESENTER

TODAY TOPIC IS
PWM

หลักการสร้างสัญญาณ PWM ด้วยวิธีต่างๆ

🌿 การใช้วงจรสัญญาณ sawtooth กับวงจรเปรียบเทียบสัญญาณ (comparator)

- 💡 ในเชิงซอฟต์แวร์ ใช้วงจรนับขึ้น (แทนวงจรสัญญาณ sawtooth) ร่วมกับค่า duty cycle (มีค่าในช่วงของค่าที่วงจรนับขึ้นทำได้) และใช้การเปรียบเทียบค่าทั้งสองเพื่อกำหนดค่าผลลัพธ์ว่าจะเป็น 0 หรือ 1 ตัวอย่างเช่น วงจรนับขึ้นมีขนาด 16 บิต นับเลขจำนวนเต็มได้ระหว่าง 0-65535 ค่า duty cycle ที่ต้องการจึงมีค่าระหว่าง 0 (0% duty cycle) และ 65535 (100% duty cycle) ค่าที่พอร์ตขาออกของ PWM เป็น 1 เมื่อค่าจากวงจรนับขึ้นน้อยกว่าค่า duty cycle หรือเป็น 0 เมื่อเท่ากับหรือมากกว่า
- 💡 กลไกที่กล่าวมาข้างต้น มักถูกสร้างเป็นวงจรสำเร็จรูปภายใน MCU ในปัจจุบัน โดยเป็นส่วนหนึ่งของ Timer-counter



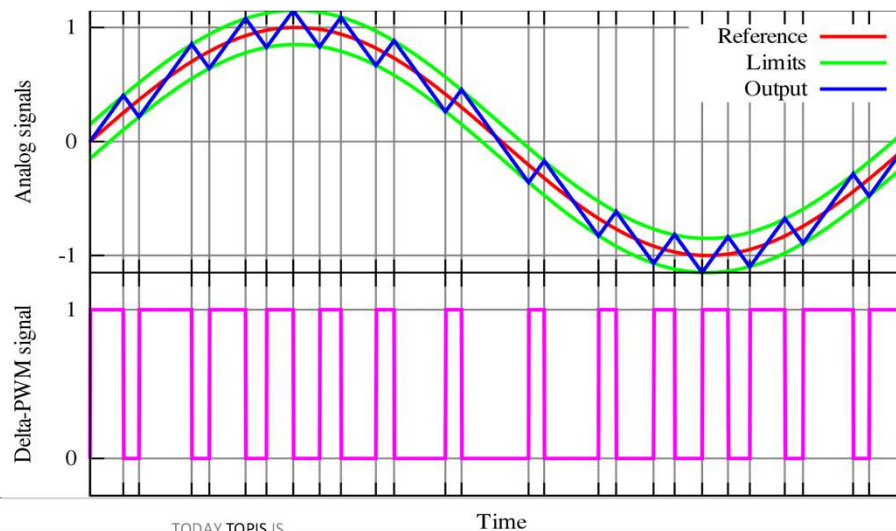
Asst.Prof. Thanwa SRIPRAMONG
PRESENTER

TODAY TOPIC IS
PWM

หลักการสร้างสัญญาณ PWM ด้วยวิธีต่างๆ

🌱 การใช้หลักการของ Delta modulation

🔵 สร้างสัญญาณแอนะล็อกอีกสองสัญญาณ โดยมีค่า offset \pm ของระดับสัญญาณจากค่าเดิม (เป็นสัญญาณช่วงบนและล่าง) และใช้ช่วงจรรีบขึ้น/ลง ที่มีอัตราความเร็วในการนับขึ้น/ลง ที่คงที่ (และแปลงออกมาเป็นสัญญาณแอนะล็อกในพิสัยครอบคลุมสัญญาณแอนะล็อกตั้งต้น) ในช่วงนับขึ้นจะให้ค่าลอจิกขาออกเป็น 1 เมื่อถึงสัญญาณช่วงบนให้นับลง ช่วงนับลงให้ค่าลอจิกเป็น 0)



Asst.Prof. Thanwa SRIPRAMONG
PRESENTER

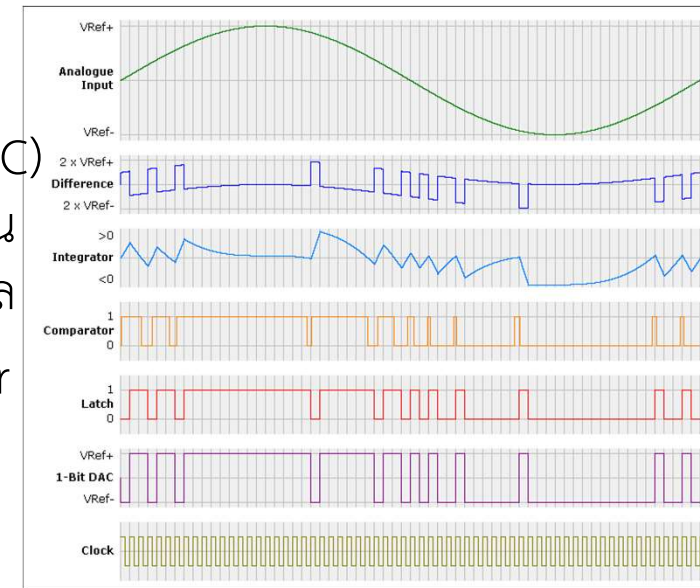
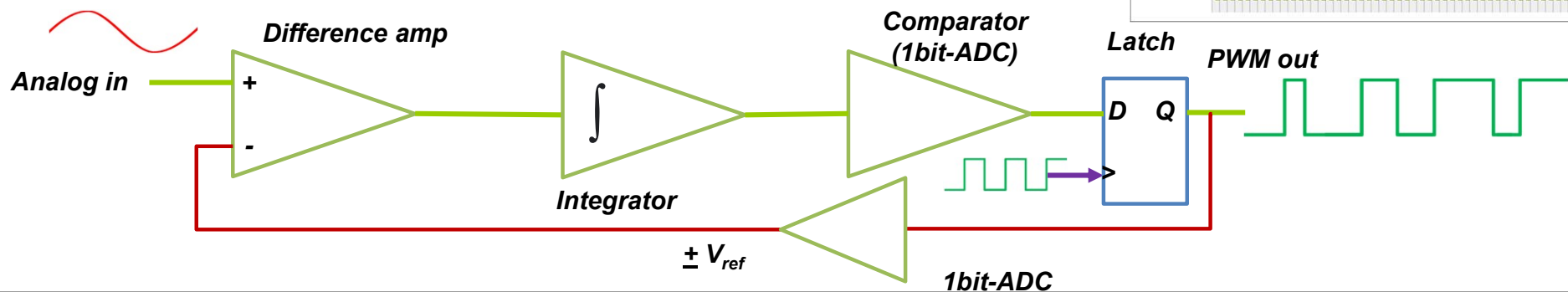
TODAY TOPIC IS
PWM

หลักการสร้างสัญญาณ PWM ด้วยวิธีต่างๆ

🌱 การใช้หลักการของ Delta-Sigma modulation ($\Delta\Sigma$)

🔵 $\Delta\Sigma$ ถูกนำมาใช้เพื่อสร้างวงจร Analog-to-digital convertor (ADC) ซึ่งในกระบวนการมีการสร้างสัญญาณ PWM เพื่อส่งต่อให้วงจรนับขึ้นที่กำหนดคาบเวลานับคงที่ แปลงคาบเวลาลอจิก 1 เป็นค่าทางดิจิทัล

🔵 ในการออกแบบจะมีการเลือกค่า RC ที่ใช้ประกอบวงจร Integrator ให้มีขนาดเหมาะสมต่อการ charge/discharge ที่ความถี่สัญญาณนาฬิกาภายในที่ใช้



Asst.Prof. Thanwa SRIPRAMONG
PRESENTER

TODAY TOPIC IS
PWM

การสร้างสัญญาณ PWM จากวงจรฐานเวลาใน STM32

- STM32 MCU รองรับการสร้างสัญญาณ PWM ภายในวงจรนับและจับเวลา
- กำหนดวงจรนับและจับเวลา ให้วนทำงานตามคาบเวลา/ความถี่ที่ต้องการ

$$F_{PWM} = \frac{F_{CLK}}{(ARR + 1) \times (PSC + 1)}$$

- ตัวอย่างเช่น ต้องการสัญญาณ PWM ที่ความถี่ 50 เฮิร์ตซ์ และค่าความละเอียด (resolution เป็น 1000) โดยฐานเวลาระบบเท่ากับ 84 เมกะเฮิร์ตซ์

ค่า $F_{CLK} = 84,000,000$

ค่า $ARR + 1 = 1000$

ค่า $F_{PWM} = 50$

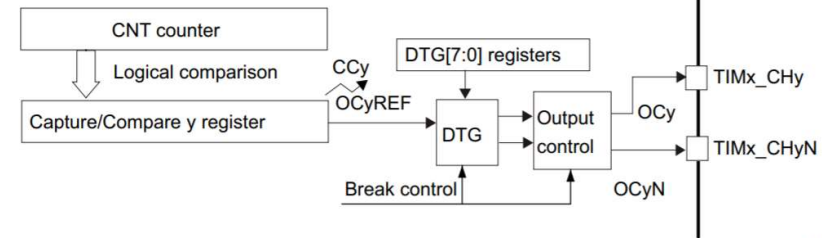
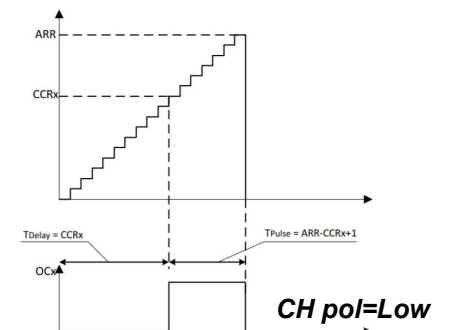
จะได้ค่า $PSC + 1 = 1678.3$ หรือประมาณ 1678

F_{pwm} : ความถี่ของสัญญาณ PWM

F_{CLK} : ความถี่สัญญาณนาฬิกาฐาน

ARR : Auto-reload register

PSC : Prescaler



MSv41561V1



Asst.Prof. Thanwa SRIPRAMONG
PRESENTER

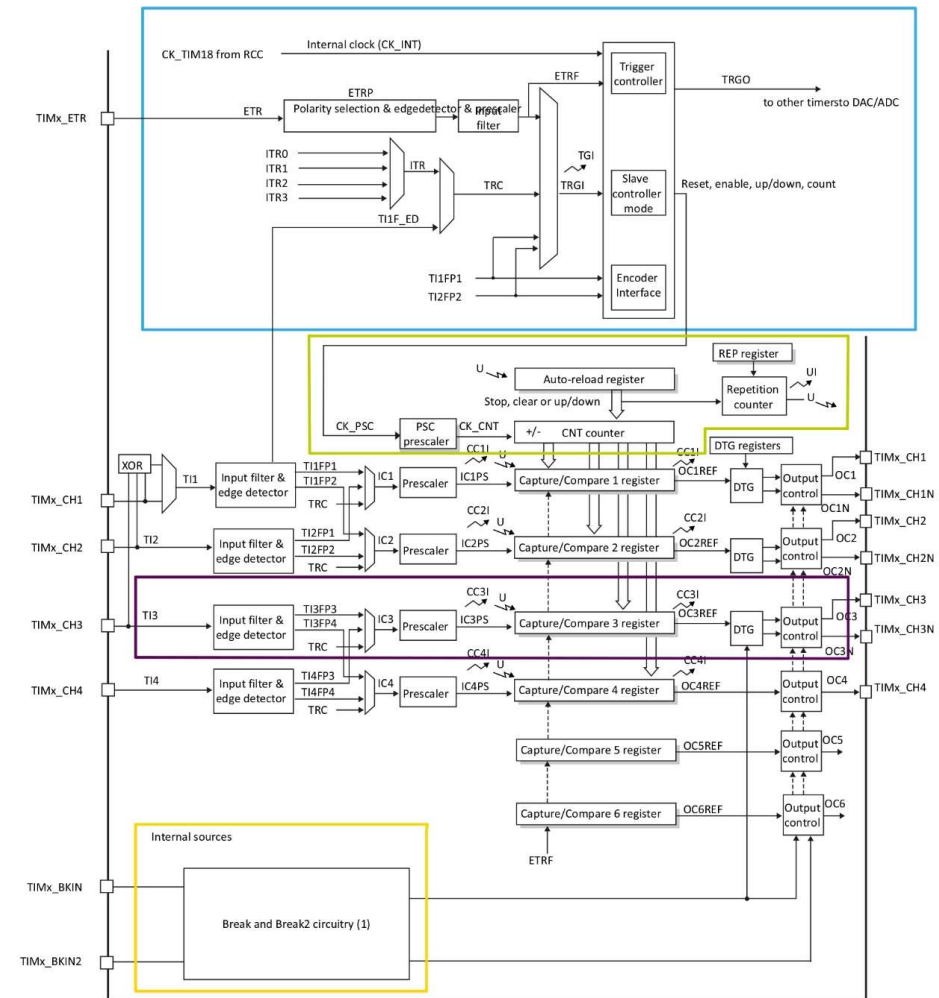
TODAY TOPIC IS
PWM

วงจร TIM ใน STM32

- 🌿 Timer/Counter แต่ละตัวใน STM32 รองรับช่องสัญญาณได้หลายช่อง
- 🌿 แต่ละช่องใช้ตัวนับขึ้น/ลง ตัวเดียวกัน แต่ตัวเรจิสเตอร์เปรียบเทียบกับสัญญาณแยกอิสระจากกัน
- 💡 สามารถควบคุมสัญญาณ PWM ที่มี duty cycle แตกต่างกันในแต่ละช่องสัญญาณได้ แต่จะมีความถี่เดียวกัน ตำแหน่งเริ่มของสัญญาณพร้อมกัน

🌈 (ควบคุม PWM ที่ CCRx)

Figure 1. TIM1 timer-peripheral block diagram



การใช้งาน PWM ควบคุมอุปกรณ์ output

- 🌿 อุปกรณ์หลายประเภทไม่สามารถตอบสนองต่อระดับแรงดันได้อย่างเชิงเส้น
 - 💧 เมื่อแรงดันต่ำกว่าค่าอ้างอิงค่าหนึ่ง อุปกรณ์จะไม่ทำงาน (หรืออาจเสี่ยงต่อความเสียหาย)
 - 💧 ตัวอย่างเช่น LED และ DC motor
 - 💧 จึงต้องอาศัยการจ่ายกระแส/แรงดัน ในรูปของ PWM ที่ค่า duty cycle (และความถี่)ที่เหมาะสม
- 🌿 อุปกรณ์บางประเภทถูกออกแบบมาเพื่อมารับสัญญาณ PWM เพื่อนำไปควบคุม
 - 💧 ตัวอย่างเช่น เซอร์โวลิวขนาดเล็ก ที่ต้องการสัญญาณ PWM ในย่าน duty cycle และในความถี่ตามที่กำหนด



Asst.Prof. Thanwa SRIPRAMONG
PRESENTER

TODAY TOPIC IS
PWM



สรุปหัวข้อ

- 🌿 การมอดูเลตเชิงความกว้างพัลส์ (Pulse Width Modulation- PWM) เป็นการมอดูเลตสัญญาณแอนะล็อกกับพาหะที่เป็นสัญญาณดิจิทัล
- 🌿 ค่า duty cycle คืออัตราส่วนระหว่างช่วงลอจิก 1 ต่อลอจิก 0 ของสัญญาณ PWM หนึ่งรอบสัญญาณ
- 🌿 มีวิธีการมากมายในการสร้างสัญญาณ PWM ทั้งด้วยวงจรทางฮาร์ดแวร์ที่เป็นแอนะล็อก และโดยใช่วงจรดิจิทัล (อันที่จริงยังสามารถเขียนซอฟต์แวร์ในการสร้างสัญญาณ PWM โดยอาศัยวงจรฐานเวลาเป็นตัวอ้างอิงได้อีกทางหนึ่ง)
- 🌿 สัญญาณ PWM ถูกนำไปใช้ในงานควบคุมอุปกรณ์ที่ไม่ตอบสนองต่อแรงดันอย่างเชิงเส้น หรือใช้กับอุปกรณ์ที่ถูกออกแบบมาโดยเฉพาะให้รับสัญญาณ PWM เพื่อนำไปควบคุม
 - 🌀 วงจร ADC แบบ Delta-sigma modulation ใช้สัญญาณ PWM ร่วมกับวงจรนับขึ้นเพื่อแปลงสัญญาณแอนะล็อกเป็นดิจิทัล



Asst.Prof. Thanwa SRIPRAMONG
PRESENTER

TODAY TOPIC IS
PWM