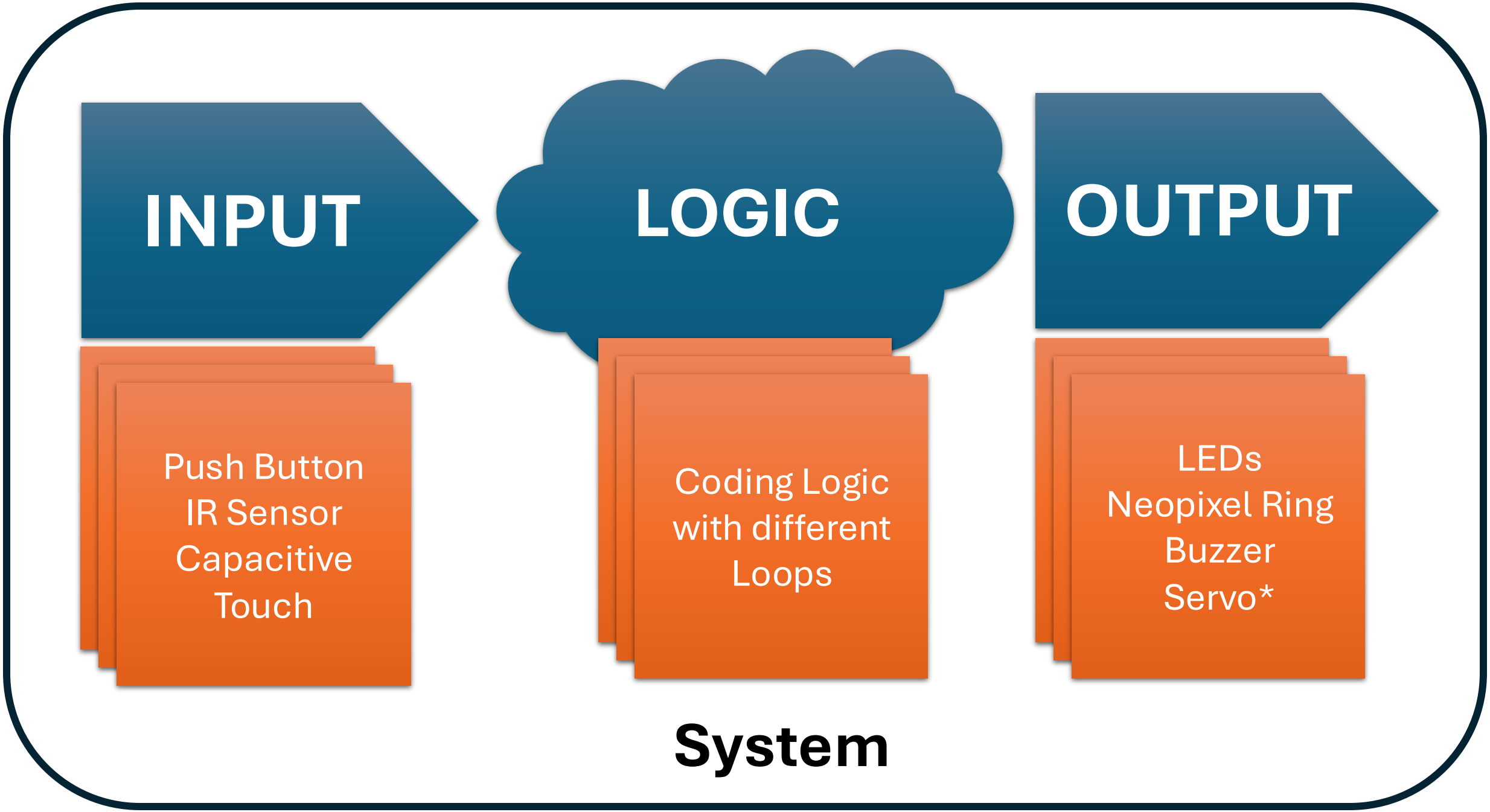
Open Design & **Technology**

Week 6

Empowering Designers to Embrace Technology


INPUT
LOGIC
OUTPUT
Push Button
IR Sensor
Capacitive
Touch
Coding Logic
with different
Loops
LEDs
Neopixel Ring
Buzzer
Servo*
System

2 Push Buttons + 2 "If" Loops

1st Button Pressed :
Print("only Button 1 pressed")

2nd Button Pressed :
Print("only Button 2 pressed")

Multiple Ifs
Vs
Elif

Counters
using
Push Button

Activity 1: Increment Counter Using Push Button

- **Objective**
  To understand how a variable can be used as a counter and incremented based on an external event.
- **Description**
  In this program, a counter variable is initialized to zero. Each time Push Button is pressed, the counter value is increased by one.
  Also, the updated counter value is printed on the serial console.

Activity 2: Increment and Decrement Counter Using Two Push Buttons

Description:

In this program, two push buttons are used to control a single counter variable:

- **Push Button 1** increments the counter value by 1
- **Push Button 2** decrements the counter value by 1
- Each button press updates the counter and prints the current value on the serial console.

Activity 3: Increment, Decrement, and Reset Counter Using Two Push Buttons

Description:
This program extends the previous activity by introducing a **reset condition**:

- **Push Button 1 pressed** → Counter increments
- **Push Button 2 pressed** → Counter decrements
- **Both buttons pressed together** → Counter resets to zero

NeoPixel

with counters

NeoPixel Control Using Push Buttons (Counter-Based Output)

Objective :

To understand how a counter value can be mapped to a physical output by controlling NeoPixel LEDs using push buttons.

In this activity, NeoPixel ring is used as a visual representation of a counter value.

Functional Behavior :

Push Button 1 pressed

- The counter value is incremented by 1
- One additional NeoPixel LED turns **ON** (next LED in sequence)
- LEDs light up **one by one** from the first LED onwards

After every button action, the current counter value is printed on the serial console to clearly observe how the counter controls the LEDs.

In this activity, NeoPixel ring is used as a visual representation of a counter value.

Functional Behavior :

Push Button 1 pressed

- The counter value is incremented by 1
- One additional NeoPixel LED turns **ON** (next LED in sequence) **with a different color**
- LEDs light up **one by one** from the first LED onwards

After every button action, the current counter value is printed on the serial console to clearly observe how the counter controls the LEDs.

In this activity, NeoPixel ring is used as a visual representation of a counter value.

Functional Behavior :

Push Button 1 pressed

- The counter value is incremented by 1
- One additional NeoPixel LED turns **ON** (next LED in sequence)
- LEDs light up **one by one** from the first LED onwards

Push Button 2 pressed

- The counter value is decremented by 1
- One NeoPixel LED turns **OFF** (in reverse order)

After every button action, the current counter value is printed on the serial console to clearly observe how the counter controls the LEDs.

In this activity, NeoPixel ring is used as a visual representation of a counter value.

Functional Behavior :

Push Button 1 pressed

- The counter value is incremented by 1
- One additional NeoPixel LED turns **ON** (next LED in sequence)
- LEDs light up **one by one** from the first LED onwards

Push Button 2 pressed

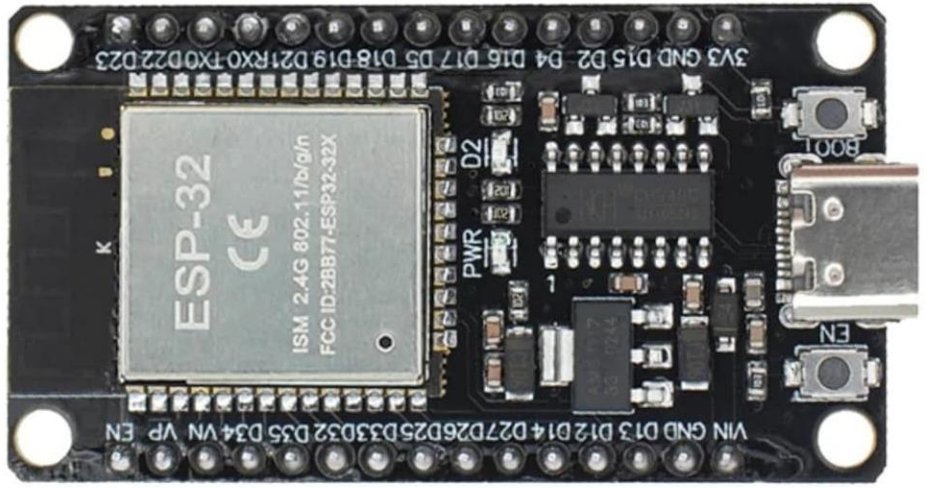
- The counter value is decremented by 1
- One NeoPixel LED turns **OFF** (in reverse order)

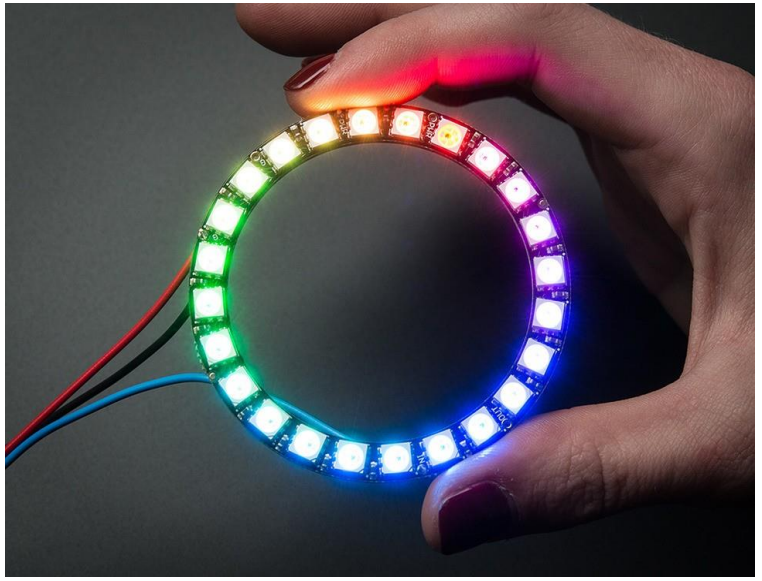
Both buttons pressed together

- The counter is reset to zero
- All NeoPixel LEDs are turned **OFF**

After every button action, the current counter value is printed on the serial console to clearly observe how the counter controls the LEDs.

Push Button1		Push Button2		
Status	**Logic**	**Status**	**Logic**	**Output**
Not Pressed	1	Not Pressed	1	Pattern 1
Pressed	0	Not Pressed	1	Pattern 2
Not Pressed	1	Pressed	0	Pattern 3
Pressed	0	Pressed	0	Pattern 4

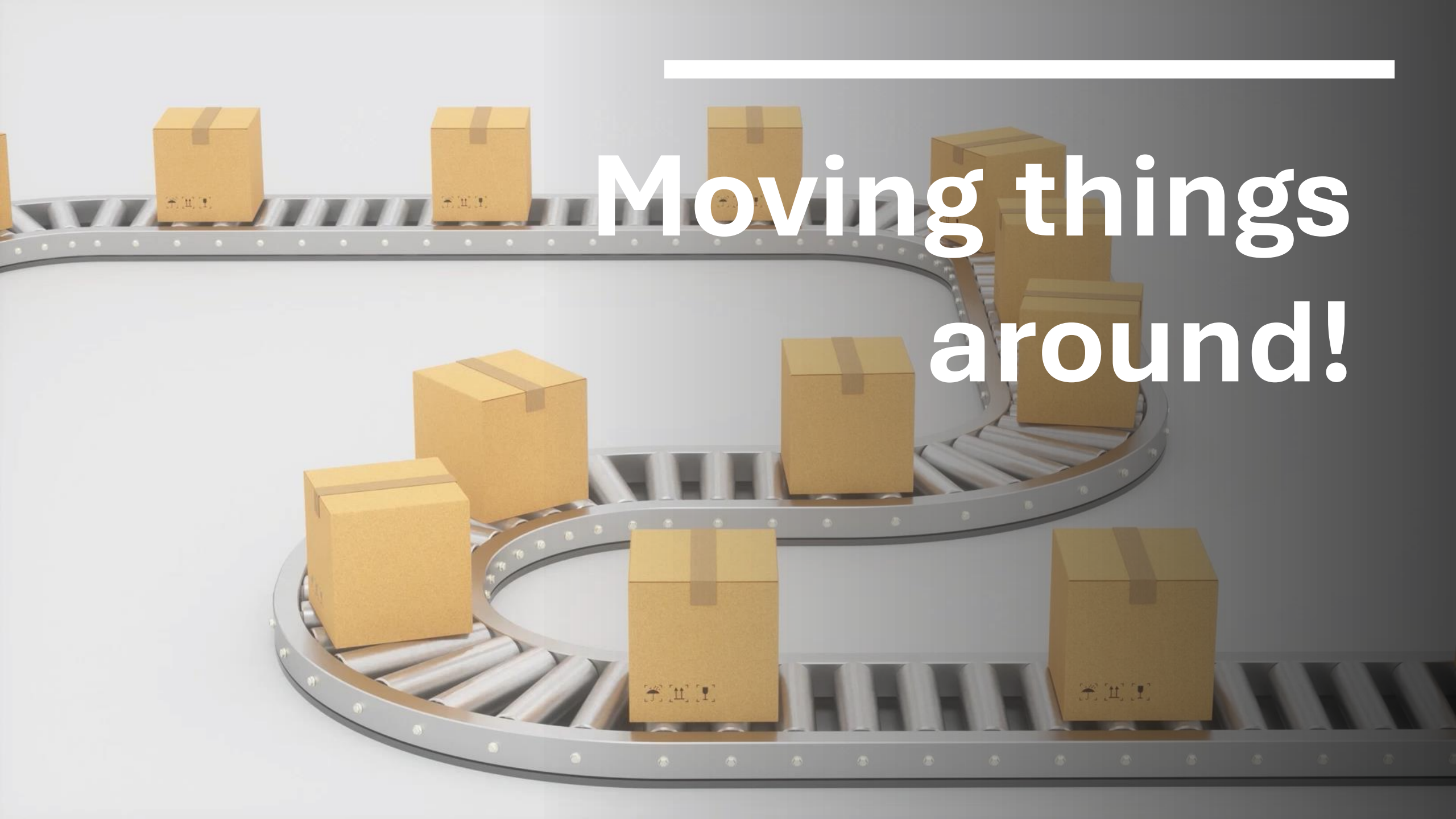




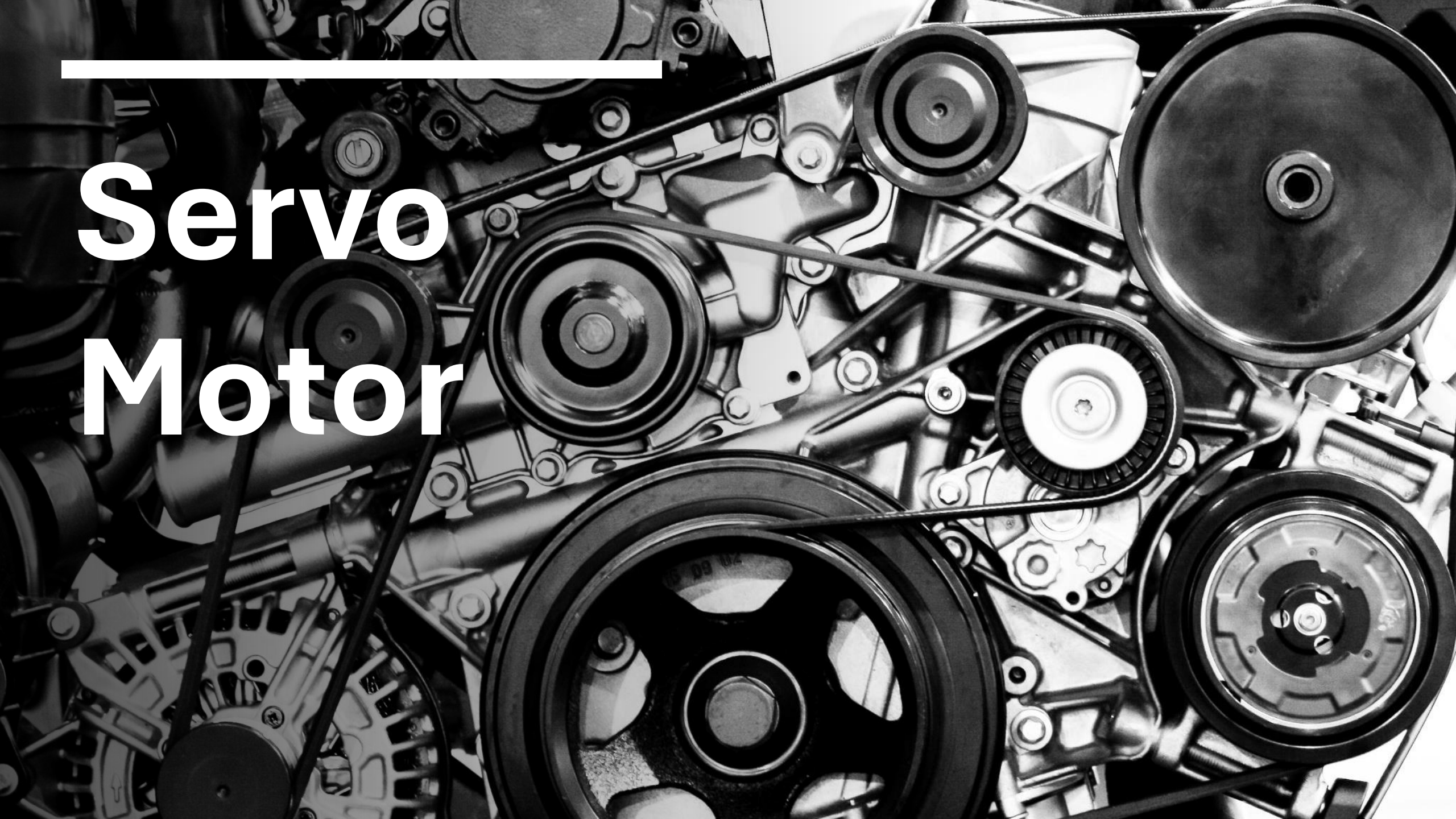

List

"For" Loop in a **"List"**


"For" Loop
Flow
Limitation


Moving things
around!

Servo Motor

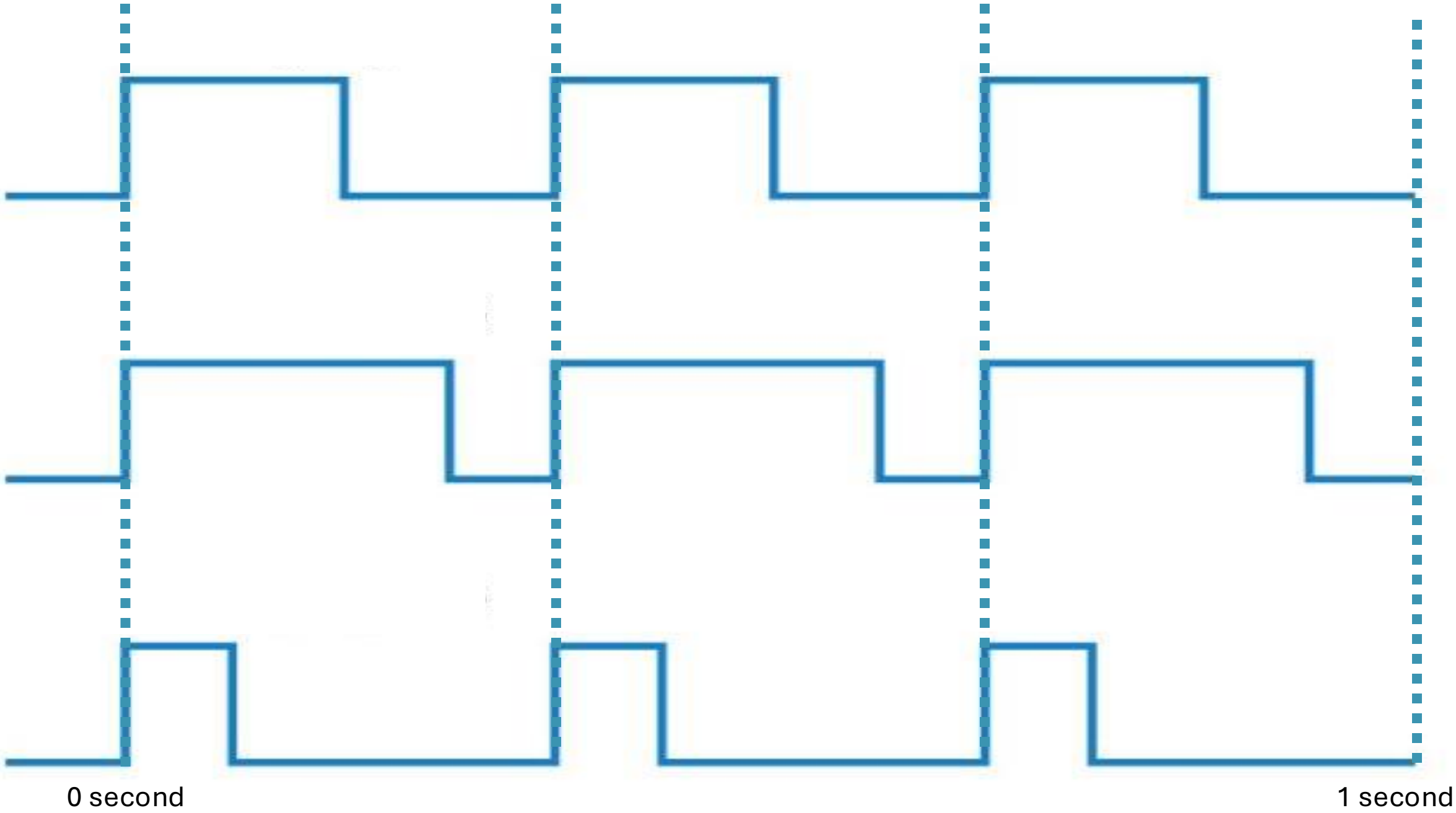


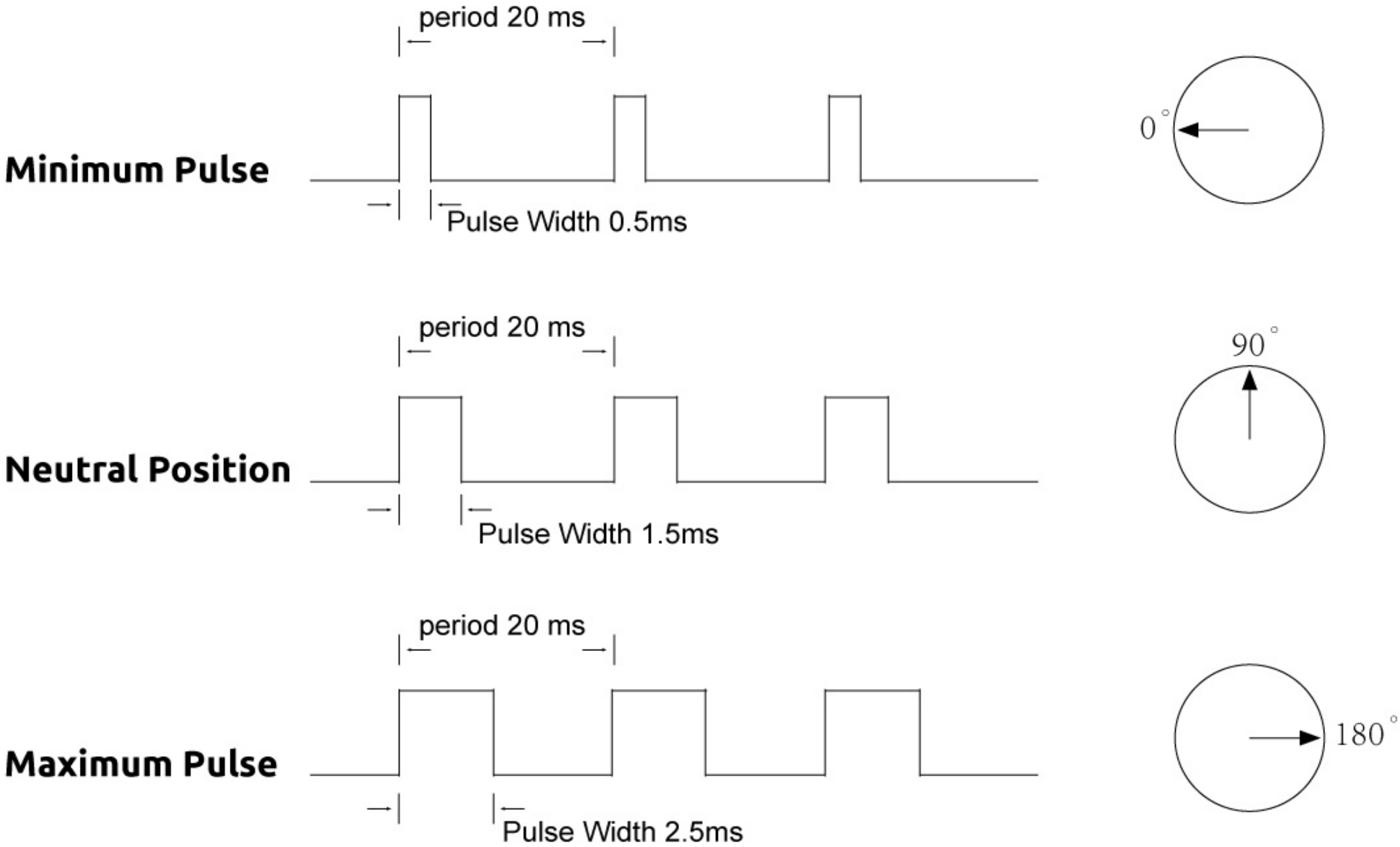
Pules Width Modulation

Duty Cycle

Frequency

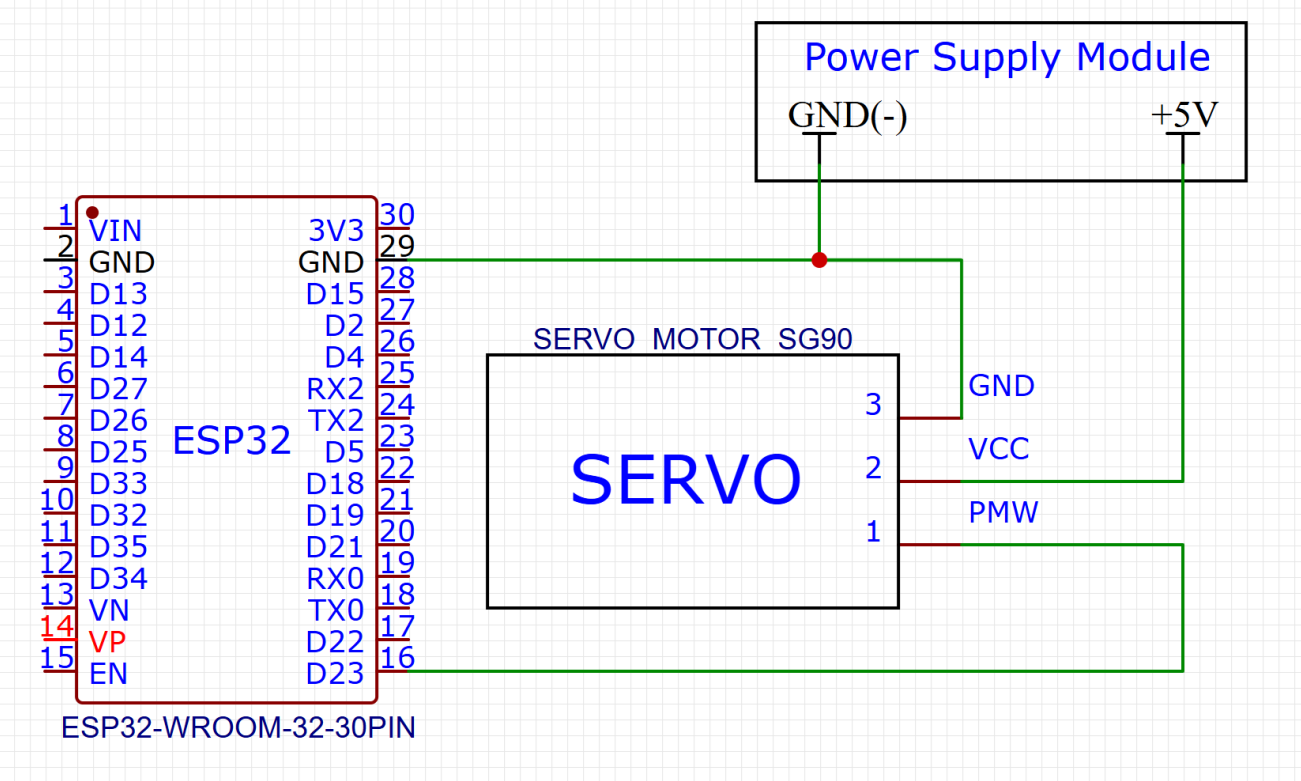
Time Period


0 second
1 second


period 20 ms
Minimum Pulse
Pulse Width 0.5ms
0°
period 20 ms
Neutral Position
Pulse Width 1.5ms
90°
period 20 ms
Maximum Pulse
Pulse Width 2.5ms
180°

The Duty Cycle of the PWM signal corresponds to the position of the servo motor shaft

Servo	ESP32	Power Module
+5v(Red)		+5v
GND(Brown)	GND	GND (-)
PWM(Orange)	GPIO	



DutyCycle – PWM in MicroPython (ESP32)

DutyCycle	Value
100%	1023
75%	768
50%	512
25%	256
0%	0

PWM resolution of ESP32 = 10 bit (2^10 = 1024)

PWM for servo motor

Frequency = **50 Hz**
Time period = **20ms**

Pulse Width to be given to servo	Duty Cycle	Value for Micro Python Code	Position (in degree)
0.5ms	2.5%	~35	~0
1.5ms	7.5%	~77	~90
2.5ms	12.5%	~120	~180

Learning Outcomes Week 6

- ✓Counters
- ✓Push Buttons
- ✓List
- ✓Servo Motors