Code Explanation

```python
import time
import board
import busio
from adafruit_crickit import crickit
import adafruit_us100

# ----- Ultrasonic Setup -----
uart = busio.UART(board.TX, board.RX, baudrate=9600)
us100 = adafruit_us100.US100(uart)

# ----- Servo Setup -----
servo = crickit.servo_1
servo.angle = 90  # Neutral

# ----- Alias for seesaw object -----
ss = crickit.seesaw

# IR sensor pins
right = crickit.SIGNAL1
left = crickit.SIGNAL8
ss.pin_mode(right, ss.INPUT_PULLUP)  # Right sensor
ss.pin_mode(left, ss.INPUT_PULLUP)  # Left sensor

# Motors
left_motor = crickit.dc_motor_1
right_motor = crickit.dc_motor_2

# ----- Movement functions -----
def stop():
    left_motor.throttle = 0
    right_motor.throttle = 0
    print("STOP")

```

```
def forward(speed=0.5):
    left_motor.throttle = speed
    right_motor.throttle = speed
    print("FORWARD")

def turn_left(speed=0.5):
    left_motor.throttle = 0
    right_motor.throttle = speed
    print("TURN LEFT")

def turn_right(speed=0.5):
    left_motor.throttle = speed
    right_motor.throttle = 0
    print("TURN RIGHT")

# Action dictionary

last_state = "None"

# ----- Obstacle push function -----
def push_obstacle():
    print("PUSHING obstacle")
    servo.angle = 0      # Push forward
    time.sleep(0.5)
    servo.angle = 180    # Push backward
    time.sleep(0.5)
    servo.angle = 90     # Neutral

# ----- Main Loop -----
while True:
 #  Check ultrasonic distance
    try:
        distance = us100.distance
        if distance is not None and distance < 15:
```

```
            print("--obstacle detected", distance)
            state="stop"
            if state != last_state:
                last_state="stop"
            stop()
            time.sleep(0.5)
            push_obstacle()
            time.sleep(0.5)  # Pause before resuming
            continue  # Skip rest of loop to avoid immediate movement
    except RuntimeError:
        print("Ultrasonic read error")

    #Read IR sensors
    right_sensor = ss.digital_read(right)  # Right IR
    left_sensor = ss.digital_read(left)   # Left IR

    # Determine state
    if not left_sensor and not right_sensor:
        state = "forward"
    elif left_sensor and not right_sensor:
        state = "turn_right"
    elif not left_sensor and right_sensor:
        state = "turn_left"
    else:
        state = "stop"

    # Execute only if state changed
    if state != last_state:
        if state == "forward":
            forward(0.3)
        elif state == "turn_right":
            turn_right(0.3)
        elif state == "turn_left":
            turn_left(0.3)
        elif state == "stop":
            stop()# both on white
    last_state = state
    time.sleep(0.01)
```

```
import time
import board
import busio
from adafruit_crickit import crickit
import adafruit_us100
```

- time → delay functions.
- board & busio → used to handle board pins and UART (serial communication).
- adafruit_crickit → control motors, servo, and sensors.
- adafruit_us100 → library for Ultrasonic sensor (US-100).

```
uart = busio.UART(board.TX, board.RX, baudrate=9600)
us100 = adafruit_us100.US100(uart)
```

- Connects the US-100 ultrasonic sensor via UART.
- Baud rate = 9600 bps (communication speed).
- us100.distance will give distance in centimeters.

```
servo = crickit.servo_1
servo.angle = 90  # Neutral
```

- A servo is connected to **Servo port 1**.
- Starts in the **90° neutral position**.

```
ss = crickit.seesaw
ss.pin_mode(crickit.SIGNAL1, ss.INPUT_PULLUP)  # Right sensor
ss.pin_mode(crickit.SIGNAL2, ss.INPUT_PULLUP)  # Left sensor
```

- seesaw chip controls signals.
- SIGNAL1 = right IR sensor
- SIGNAL2 = left IR sensor
- INPUT_PULLUP = input mode with internal pull-up resistor.

```
left_motor = crickit.dc_motor_1
```

```
right_motor = crickit.dc_motor_2
```

- Defines left and right DC motors.

```
def stop():
    left_motor.throttle = 0
    right_motor.throttle = 0
    print("STOP")
def forward(speed=0.5):
    left_motor.throttle = speed
    right_motor.throttle = speed
    print("FORWARD")
def turn_left(speed=0.5):
    left_motor.throttle = 0
    right_motor.throttle = speed
    print("TURN LEFT")
def turn_right(speed=0.5):
    left_motor.throttle = speed
    right_motor.throttle = 0
    print("TURN RIGHT")
```

- Functions to control robot movement.
- Speed can be adjusted (default = 0.5).

```
def push_obstacle():
    print("PUSHING obstacle")
    servo.angle = 0      # Push forward
    time.sleep(0.5)
    servo.angle = 180    # Push backward
    time.sleep(0.5)
    servo.angle = 90     # Neutral
```

- Uses servo as a pusher/arm.
- Moves from 90° → 0° → 180° → back to 90°.
- Simulates pushing an obstacle out of the way.

```
while True:
    # Check ultrasonic distance
    try:
        distance = us100.distance
        if distance is not None and distance < 15:
            print("--obstacle detected", distance)
            state="stop"
            if state != last_state:
                last_state="stop"
            stop()
            push_obstacle()
            time.sleep(0.5)
            continue
    except RuntimeError:
        print("Ultrasonic read error")
```

- Reads ultrasonic sensor.
- If an object is closer than 10.5 cm →
- Robot stops,
- pushes obstacle using servo,
- waits 0.5s,
- then continues line following.
- If error occurs, prints "Ultrasonic read error".

```
 right_sensor = ss.digital_read(crickit.SIGNAL1)  # Right IR
 left_sensor = ss.digital_read(crickit.SIGNAL2)   # Left IR
```

- Reads IR sensor values:
- Black line = False (0)
- White surface = True (1)

```
    if not left_sensor and not right_sensor:
        state = "forward"
    elif left_sensor and not right_sensor:
        state = "turn_right"
    elif not left_sensor and right_sensor:
        state = "turn_left"
    else:
        state = "stop"
```

- Both black → go forward
- Left white, Right black → turn right
-

```
    if state != last_state:
```

- Avoids repeating the same command when robot keeps doing the same motion.

```
        if state == "forward":
            forward(0.3)
        elif state == "turn_right":
            turn_right(0.3)
        elif state == "turn_left":
            turn_left(0.3)
        elif state == "stop":
            stop()# both on white
```

- Executes the appropriate motion depending on state:
- `forward(0.3)` → moves straight

- `turn_right(0.3)` → rotates right
- `turn_left(0.3)` → rotates left
- `stop()` → stops both motors

```
    last_state = state
    time.sleep(0.01)
```

- Saves the current state for comparison in the next loop.
- Short pause for smooth execution.