

Code Explanation

```
1 import time
2 import board
3 import busio
4 from adafruit_crickit import crickit
5 import adafruit_us100
6
7 # ----- Ultrasonic Setup -----
8 uart = busio.UART(board.TX, board.RX, baudrate=9600)
9 us100 = adafruit_us100.US100(uart)
10
11 # ----- Servo Setup -----
12 servo = crickit.servo_1
13 servo.angle = 90 # Neutral
14
15 # ----- Alias for seesaw object -----
16 ss = crickit.seesaw
17
18 # IR sensor pins
19 right = crickit.SIGNAL1
20 left = crickit.SIGNAL8
21 ss.pin_mode(right, ss.INPUT_PULLUP) # Right sensor
22 ss.pin_mode(left, ss.INPUT_PULLUP) # Left sensor
23
24 # Motors
25 left_motor = crickit.dc_motor_1
26 right_motor = crickit.dc_motor_2
27
28 # ----- Movement functions -----
29 def stop():
30     left_motor.throttle = 0
31     right_motor.throttle = 0
32     print("STOP")
33
```

```

34 def forward(speed=0.5):
35     left_motor.throttle = speed
36     right_motor.throttle = speed
37     print("FORWARD")
38
39 def turn_left(speed=0.5):
40     left_motor.throttle = 0
41     right_motor.throttle = speed
42     print("TURN LEFT")
43
44 def turn_right(speed=0.5):
45     left_motor.throttle = speed
46     right_motor.throttle = 0
47     print("TURN RIGHT")
48
49 # Action dictionary
50
51 last_state = "None"
52
53 # ----- Obstacle push function -----
54 def push_obstacle():
55     print("PUSHING obstacle")
56     servo.angle = 0      # Push forward
57     time.sleep(0.5)
58     servo.angle = 180    # Push backward
59     time.sleep(0.5)
60     servo.angle = 90     # Neutral
61
62 # ----- Main Loop -----
63 while True:
64     # Check ultrasonic distance
65     try:
66         distance = us100.distance
67         if distance is not None and distance < 15:

```

```

68         print("--obstacle detected", distance)
69         state="stop"
70         if state != last_state:
71             last_state="stop"
72         stop()
73         time.sleep(0.5)
74         push_obstacle()
75         time.sleep(0.5) # Pause before resuming
76         continue # Skip rest of loop to avoid immediate movement
77     except RuntimeError:
78         print("Ultrasonic read error")
79
80     #Read IR sensors
81     right_sensor = ss.digital_read(right) # Right IR
82     left_sensor = ss.digital_read(left) # Left IR
83
84     # Determine state
85     if not left_sensor and not right_sensor:
86         state = "forward"
87     elif left_sensor and not right_sensor:
88         state = "turn_right"
89     elif not left_sensor and right_sensor:
90         state = "turn_left"
91     else:
92         state = "stop"
93
94     # Execute only if state changed
95     if state != last_state:
96         if state == "forward":
97             forward(0.3)
98         elif state == "turn_right":
99             turn_right(0.3)
100        elif state == "turn_left":
101            turn_left(0.3)
102        elif state == "stop":
103            stop() # both on white
104    last_state = state
105    time.sleep(0.01)
106

```

```
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 $90^\circ \rightarrow 0^\circ \rightarrow 180^\circ \rightarrow$ 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 \rightarrow
- 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.