Mr. Path("Your task is to build a Line Following Robot.”)

Code

```python
import time
from adafruit_crickit import crickit

ss = crickit.seesaw

print("Line Follower with DC Motors using pin_mode + digital_read!")

# Set up IR sensor pins
ss.pin_mode(crickit.SIGNAL1, ss.INPUT_PULLUP)  # Left sensor
ss.pin_mode(crickit.SIGNAL8, ss.INPUT_PULLUP)  # Right sensor

# Motors
left_motor = crickit.dc_motor_1
right_motor = crickit.dc_motor_2
last_state="none"
# Movement functions
def stop():
    left_motor.throttle = 0
    right_motor.throttle = 0

def forward(speed=0.3):
    left_motor.throttle = speed
    right_motor.throttle = speed

def turn_left(speed=0.3):
    left_motor.throttle = 0
    right_motor.throttle = speed

def turn_right(speed=0.3):
    left_motor.throttle = speed
    right_motor.throttle = 0

```

```python
# Main loops
while True:
    # Read digital IR sensors (True/False)
    right_sensor = ss.digital_read(crickit.SIGNAL8)    # Left IR
    left_sensor = ss.digital_read(crickit.SIGNAL1)  # Right IR

    # Logic: adjust depending on whether True means black or white
    # 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"

    # Only run action if state changed
    if state != last_state:
        if not left_sensor and not right_sensor:
            forward(0.3)
        elif left_sensor and not right_sensor:
            turn_right(0.3)
        elif not left_sensor and right_sensor:
            turn_left(0.3)
        else:
            stop()# both on white
    last_state = state

    time.sleep(0.01)


```

Code Expalnation

import time
from adafruit_crickit import crickit
ss = crickit.seesaw

- time → for adding small delays.
- crickit → controls motors and sensors on the Adafruit Crickit board.
- ss is a shortcut for crickit.seesaw, which helps read/write sensor pins.

print("Line Follower with DC Motors using pin_mode + digital_read!")

- Displays a message when the program starts.

```
ss.pin_mode(crickit.SIGNAL1, ss.INPUT_PULLUP)
ss.pin_mode(crickit.SIGNAL8, ss.INPUT_PULLUP)
```

- SIGNAL1 = Left IR sensor
- SIGNAL8 = Right IR sensor
- INPUT_PULLUP means the input pin is held “HIGH” by default, so when the sensor detects a line, it goes “LOW.”

```
left_motor = crickit.dc_motor_1
right_motor = crickit.dc_motor_2
```

- Assigns the motors to the Crickit motor ports.

```
def stop():
left_motor.throttle = 0
right_motor.throttle = 0

def forward(speed=0.3):
left_motor.throttle = speed
right_motor.throttle = speed

def turn_left(speed=0.3):
left_motor.throttle = 0
right_motor.throttle =

speed def turn_right(speed=0.3):
left_motor.throttle = speed
right_motor.throttle = 0
```

- These control how the robot moves:

```
while True:
right_sensor = ss.digital_read(crickit.SIGNAL8)
left_sensor = ss.digital_read(crickit.SIGNAL1)
```

- Runs continuously — reads sensor data and controls motors.
- Reads the left and right sensors (True or False).

```
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"
```

Interprets sensor data:

- Both detect black (False) → Go straight.
- Left white, right black → Turn right.
- Right white, left black → Turn left.
- Both white → Stop (off the line).

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

This avoids repeating the same motor commands unnecessarily — improves smoothness.

```
        if not left_sensor and not right_sensor:
        forward(0.3)
        elif left_sensor and not right_sensor:
         turn_right(0.3)
        elif not left_sensor and right_sensor:
        turn_left(0.3)
        else:
        stop()
last_state = state
time.sleep(0.01)
```

- Saves the last movement state.
- Adds a short delay (10 ms) to prevent rapid switching.