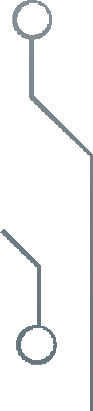
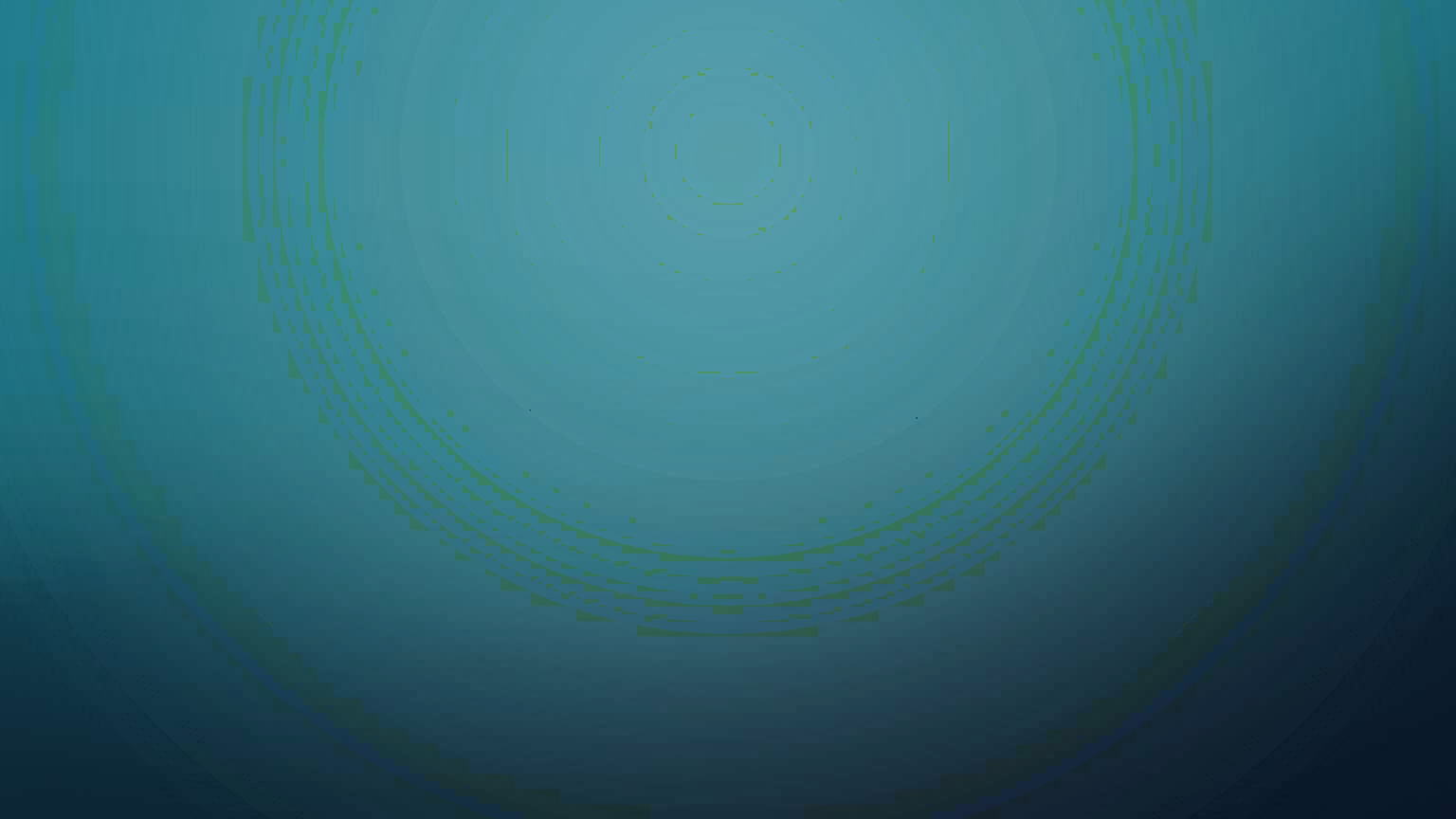


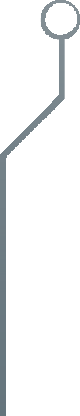
TRAFFIC MANAGEMENT SYSTEM

AGENDA:

To discuss about project objectives, sensor setup, block

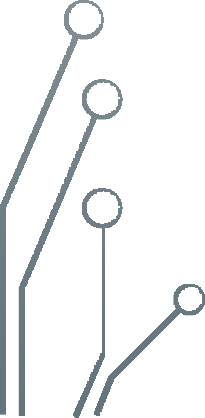
diagrams and code implementation.



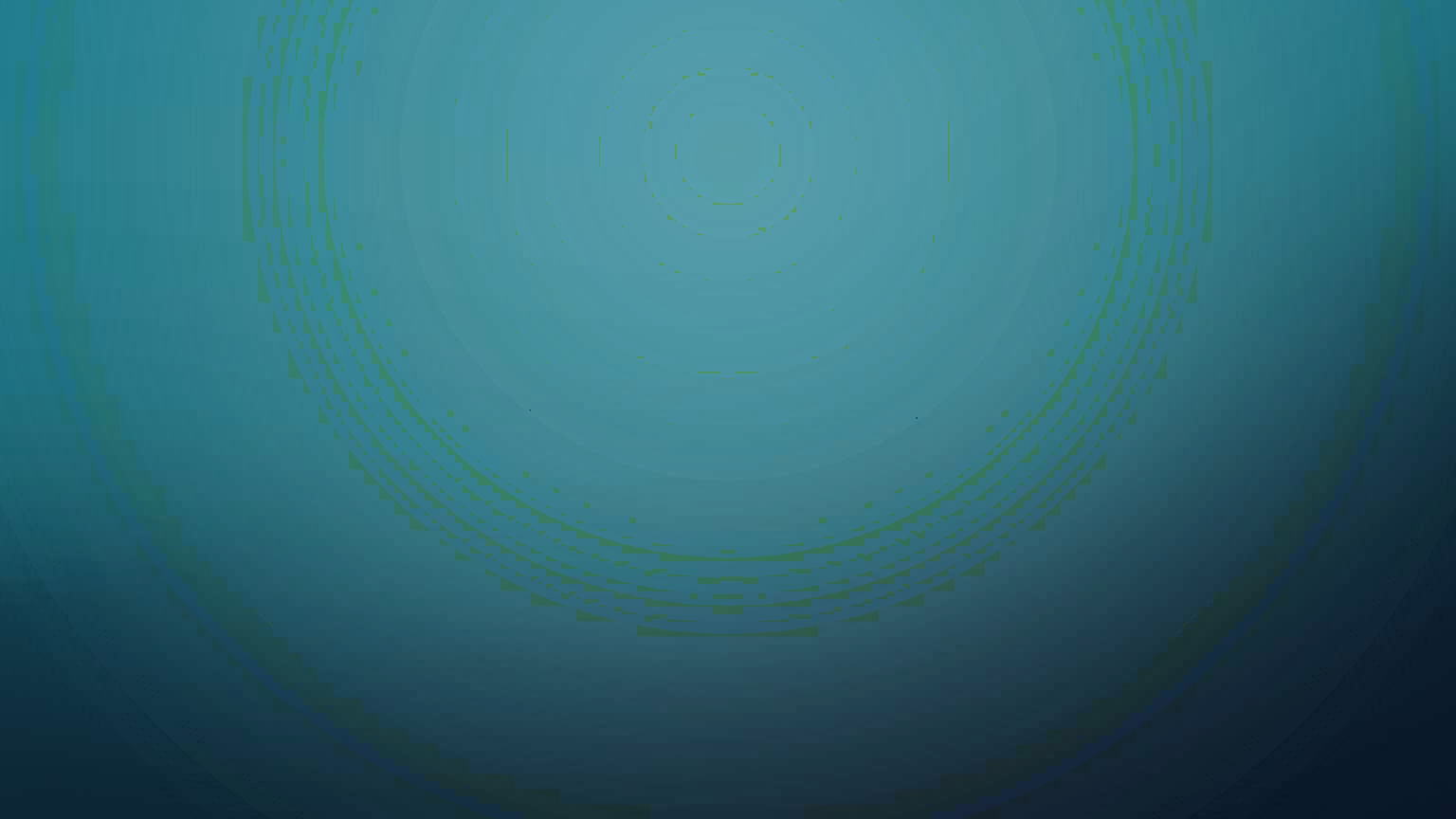


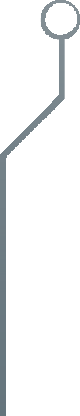
# TEAM MEMBERS:

* PRABU T (MENTOR)
* REVU BHARATH KUMAR (LEADER)
* THIYAGARAJAN D (MEMBER)
* SUDHAKAR V (MEMBER)
* MEESALA PRAVEEN (MEMBER)



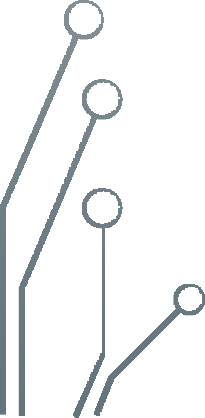
* MUSALI DIVAKAR (MEMBER)

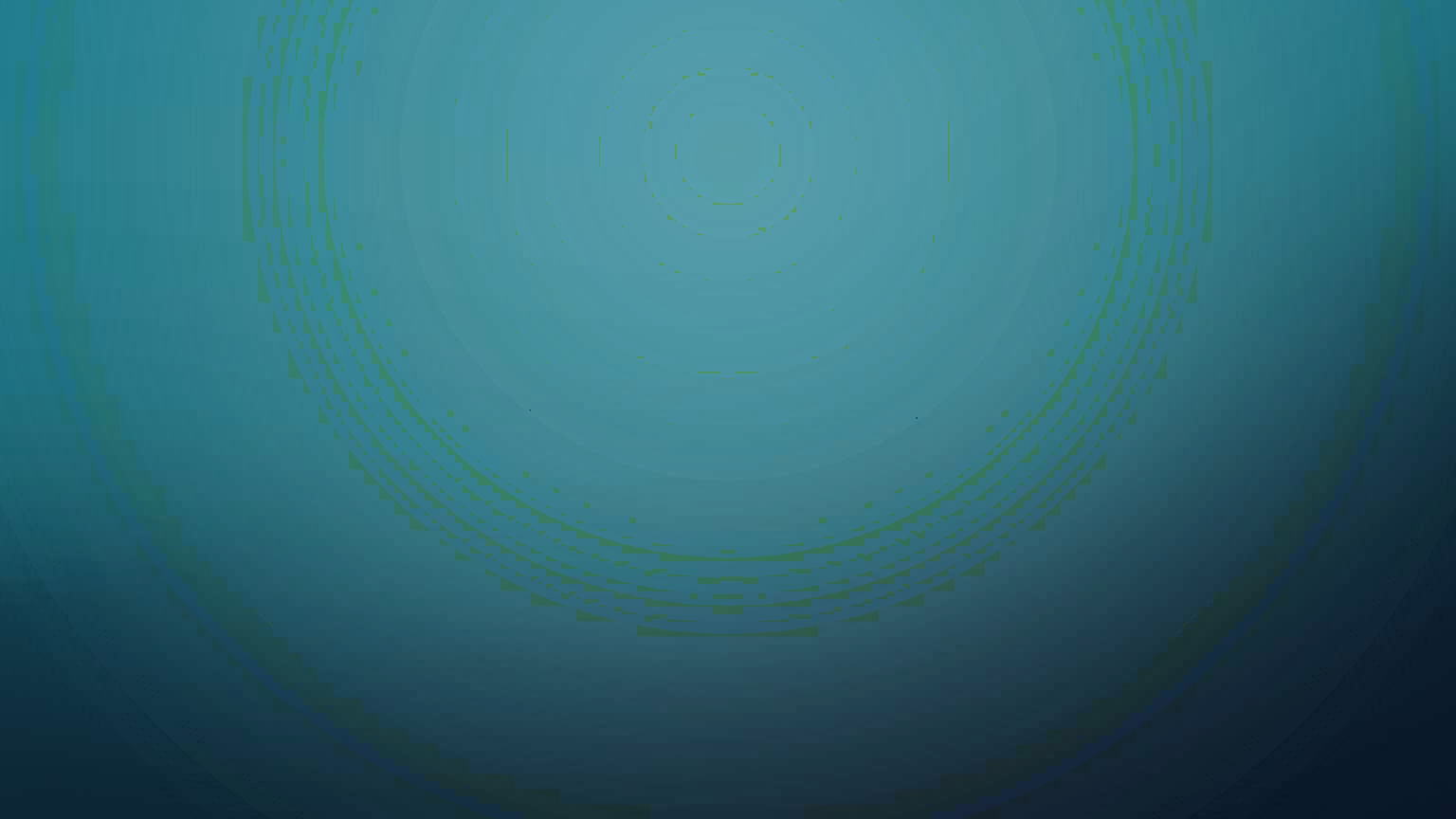


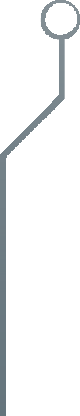


## PROJECT DEFINITION:

* The project involves using IoT devices and data analytics to monitor traffic flow and congestion in real-time, providing commuters with access to this information through a public platform or mobile apps. The objective is to help commuters make informed decisions about their routes and alleviate traffic congestion. This project includes defining objectives, designing the IoT traffic monitoring system, developing the traffic information platform, and integrating them using IoT technology and Python.



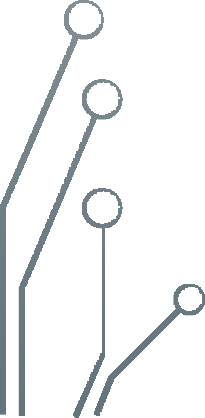




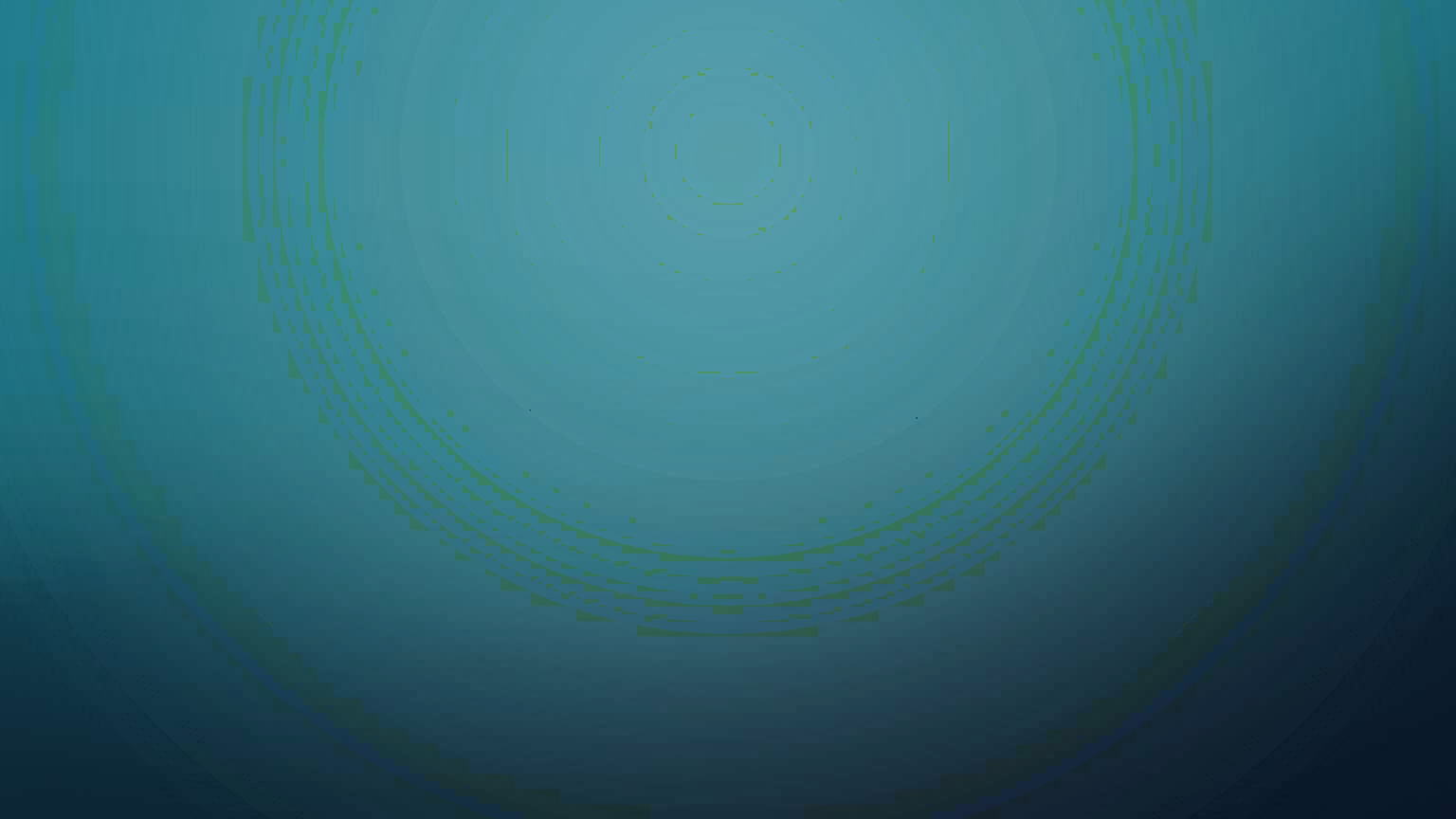
# PROJECT OBJECTIVES:

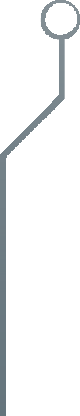
* **Real-time Traffic Monitoring:** Implement a system to continuously monitor traffic conditions in real-time.
* **Congestion Detection:** Develop algorithms to identify and flag areas of traffic congestion promptly.

 **Route Optimization:** Create a feature that suggests optimal routes for commuters based on real-time traffic data.



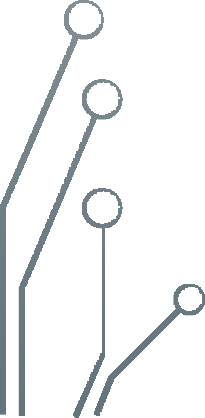
* **Improved Commuting Experience:** Enhance the overall commuting experience by providing accurate and timely traffic information to users.

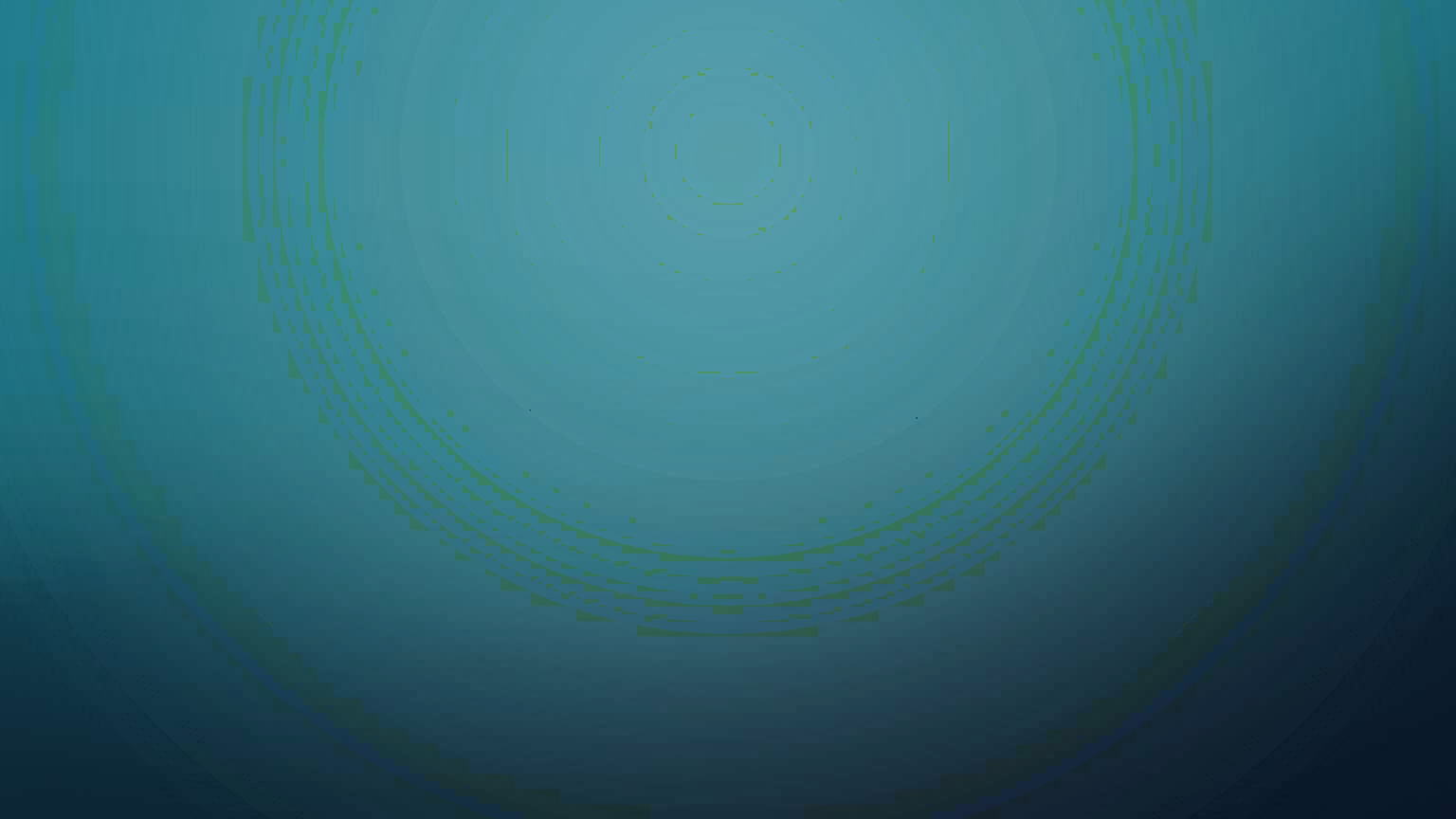


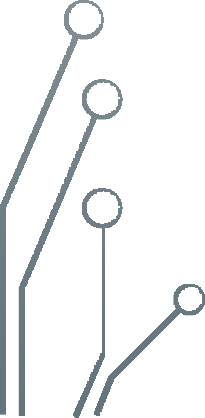
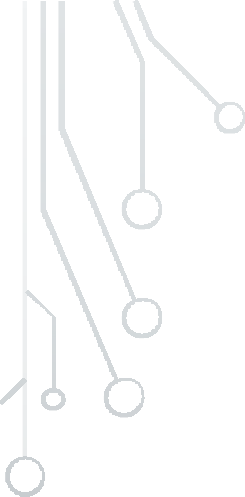


## IOT SENSOR SETUP:

* + **The Inductive Loop Detector (ILD) sensor**
  + **The ultrasonic sensors**

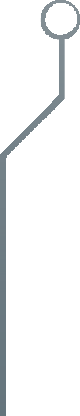






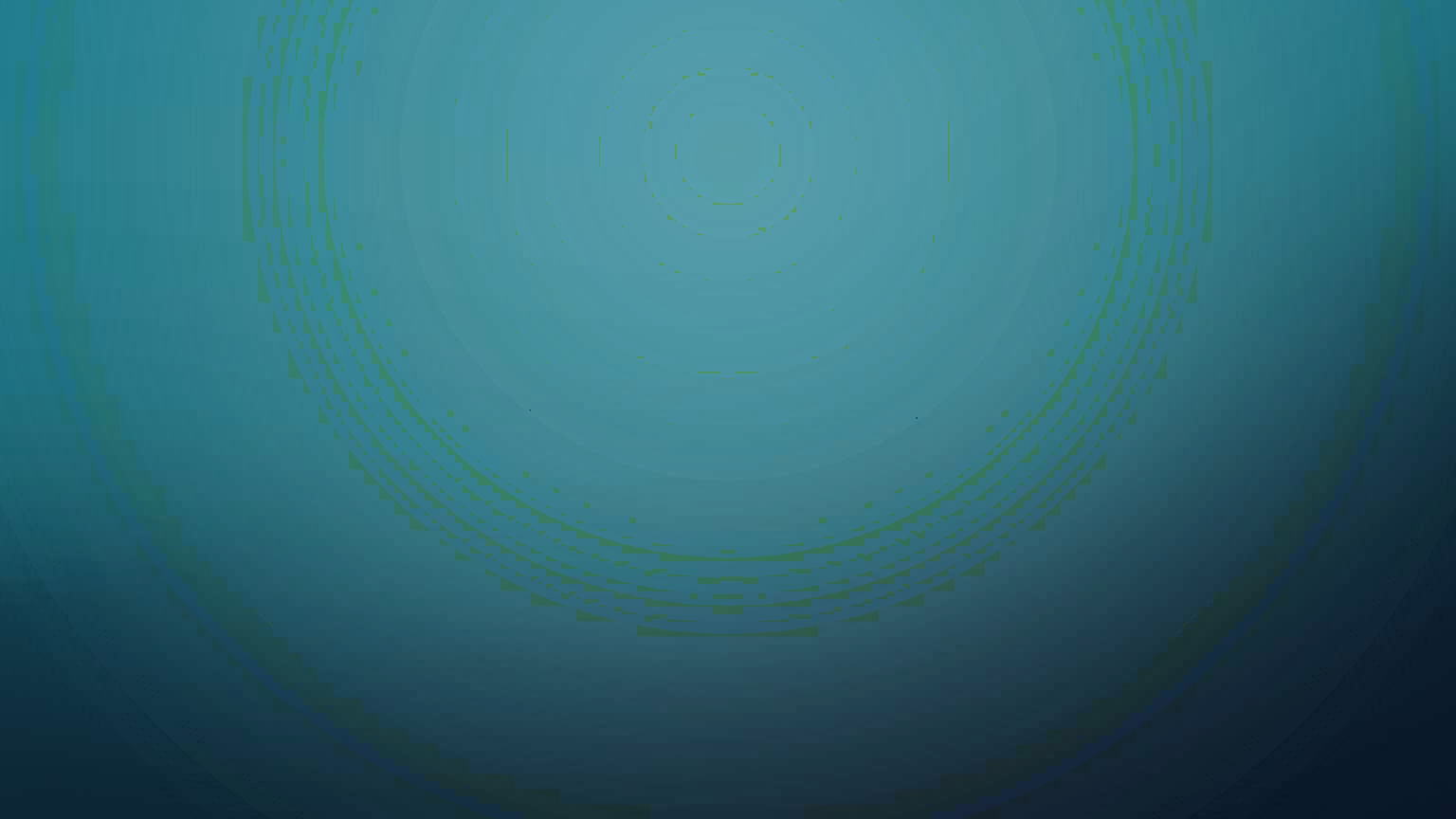


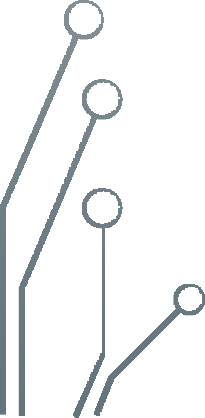
**INDUCTIVE LOOP DETECTOR (ILD) SENSOR:**



**ULTRASONIC SENSOR:**

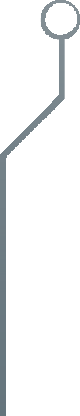






* It is one of the most common sensors in traffic management. It is used for collecting traffic flow, vehicle's occupancy,

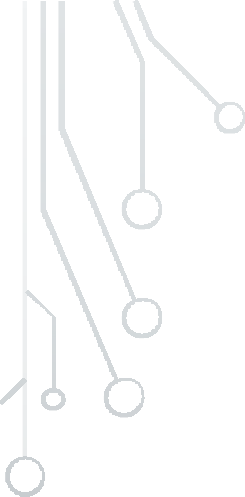
length, and speed .



* It used to detect vehicles, and the density levels of a given road are sent to an LCD, and the data sent to the server for later usage. In similar research , the authors proposed an ultrasonic sensor- based system model specifically for road intersections.

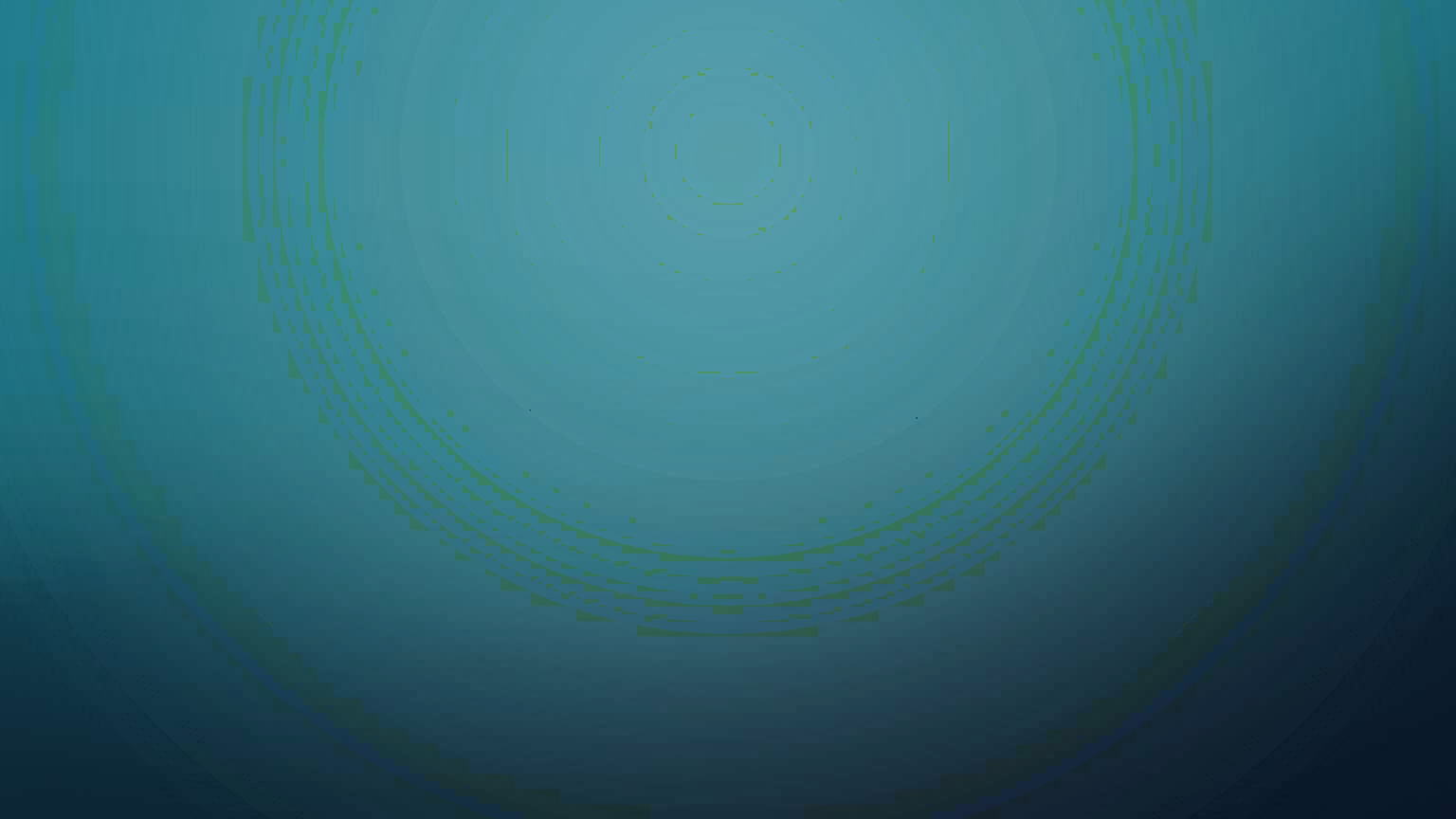
**The ultrasonic sensors**

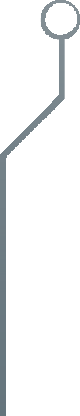
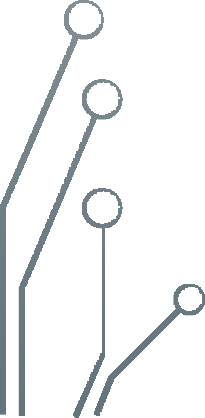
# SENSOR DESCRIPTION :

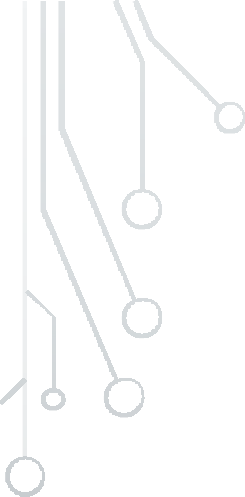


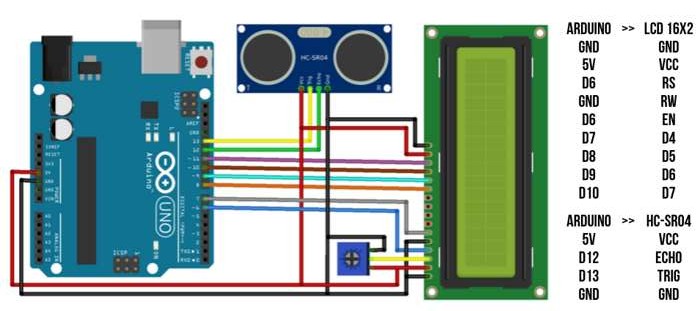
**The Inductive Loop Detector (ILD) sensor**

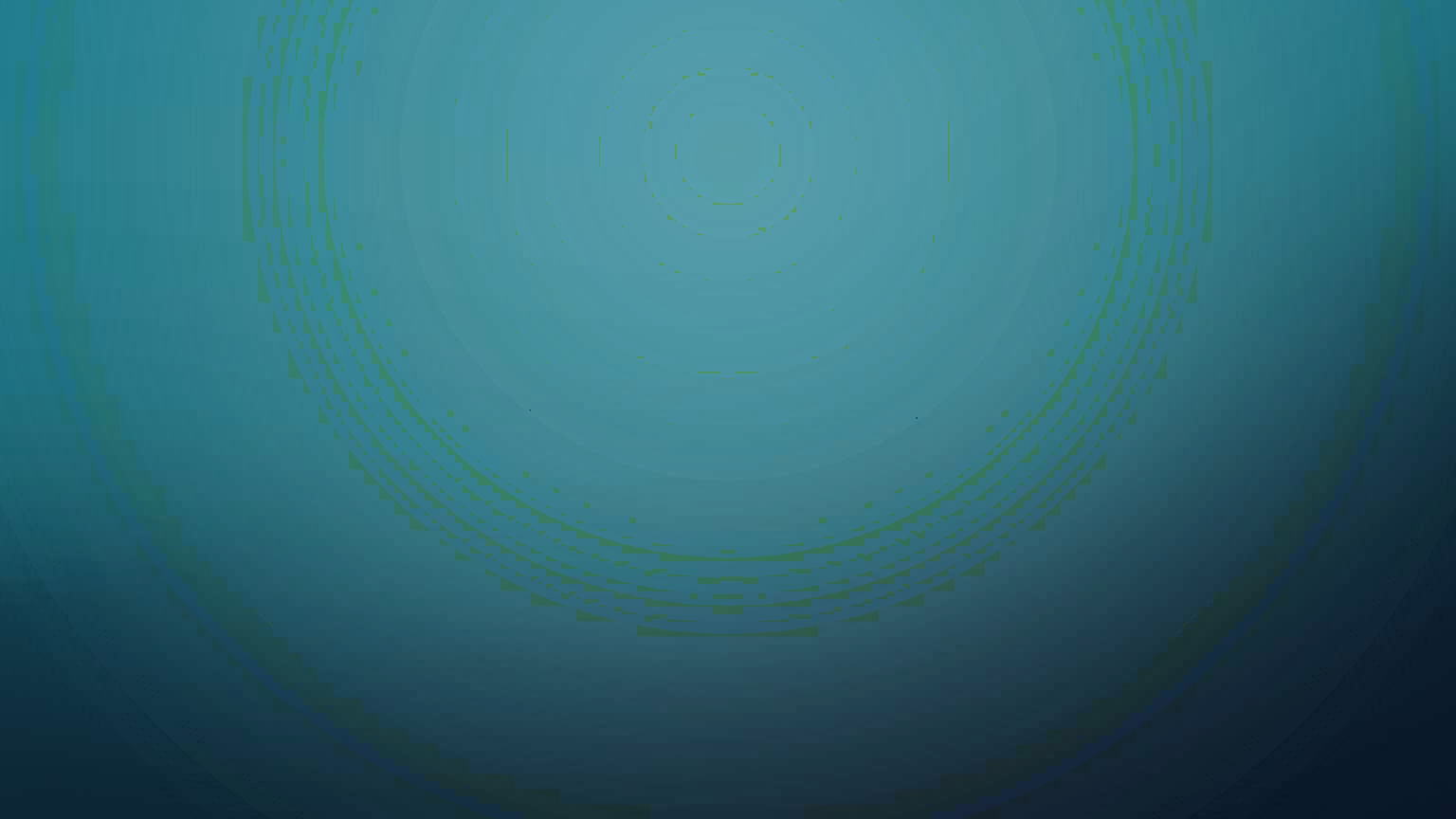


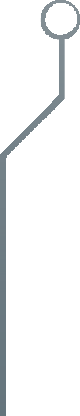




CIRCUIT DIAGRAM:





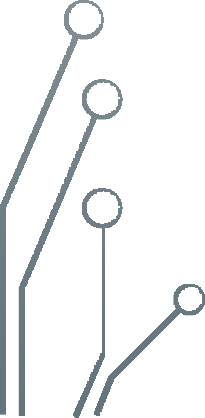


# COMPONENTS USED:

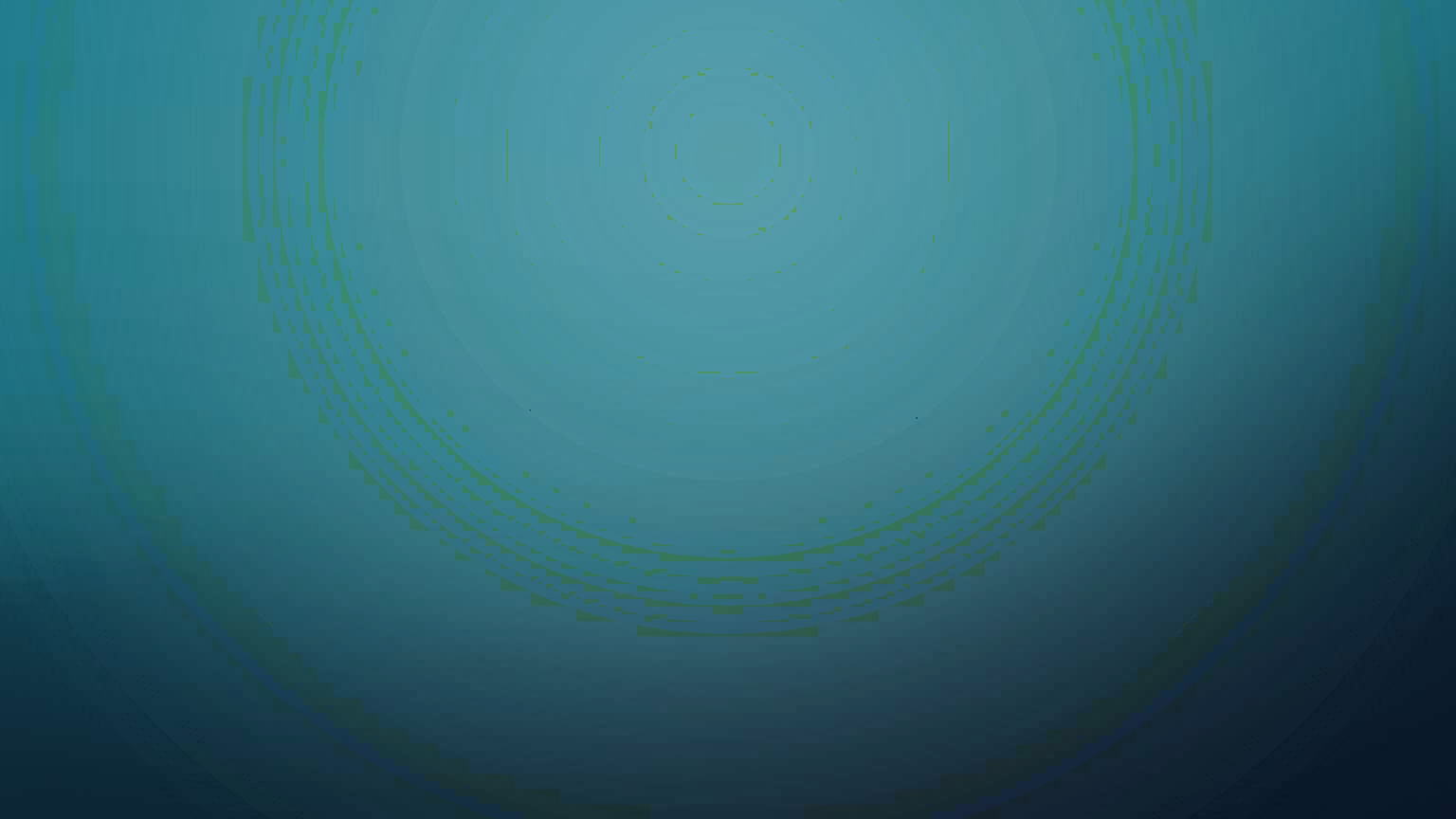
* + - Arduino UNO R3 CH340

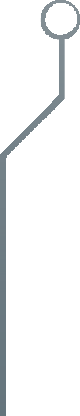
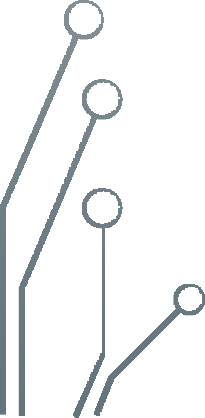
 LCD display

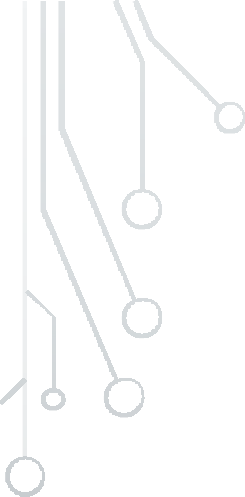
* + - Ultrasonic Sensor HC-SR04



* + - Male to Male Jumper Wires





CODE FOR ARDUINO:



import RPi.GPIO as GPIO import time

echoPin = 12

trigPin = 13

GPIO.setmode(GPIO.BCM) GPIO.setup(trigPin, GPIO.OUT) GPIO.setup(echoPin, GPIO.IN)

def measure\_distance():

GPIO.output(trigPin, GPIO.LOW) time.sleep(0.2)

GPIO.output(trigPin, GPIO.HIGH) time.sleep(0.00001) GPIO.output(trigPin, GPIO.LOW)

while GPIO.input(echoPin) == 0: pulse\_start = time.time()

while GPIO.input(echoPin) == 1: pulse\_end = time.time()

pulse\_duration = pulse\_end - pulse\_start

distance\_cm = pulse\_duration \* 34300 / 2 distance\_inch = pulse\_duration \* 13503.9 / 2

return distance\_cm, distance\_inch

try:

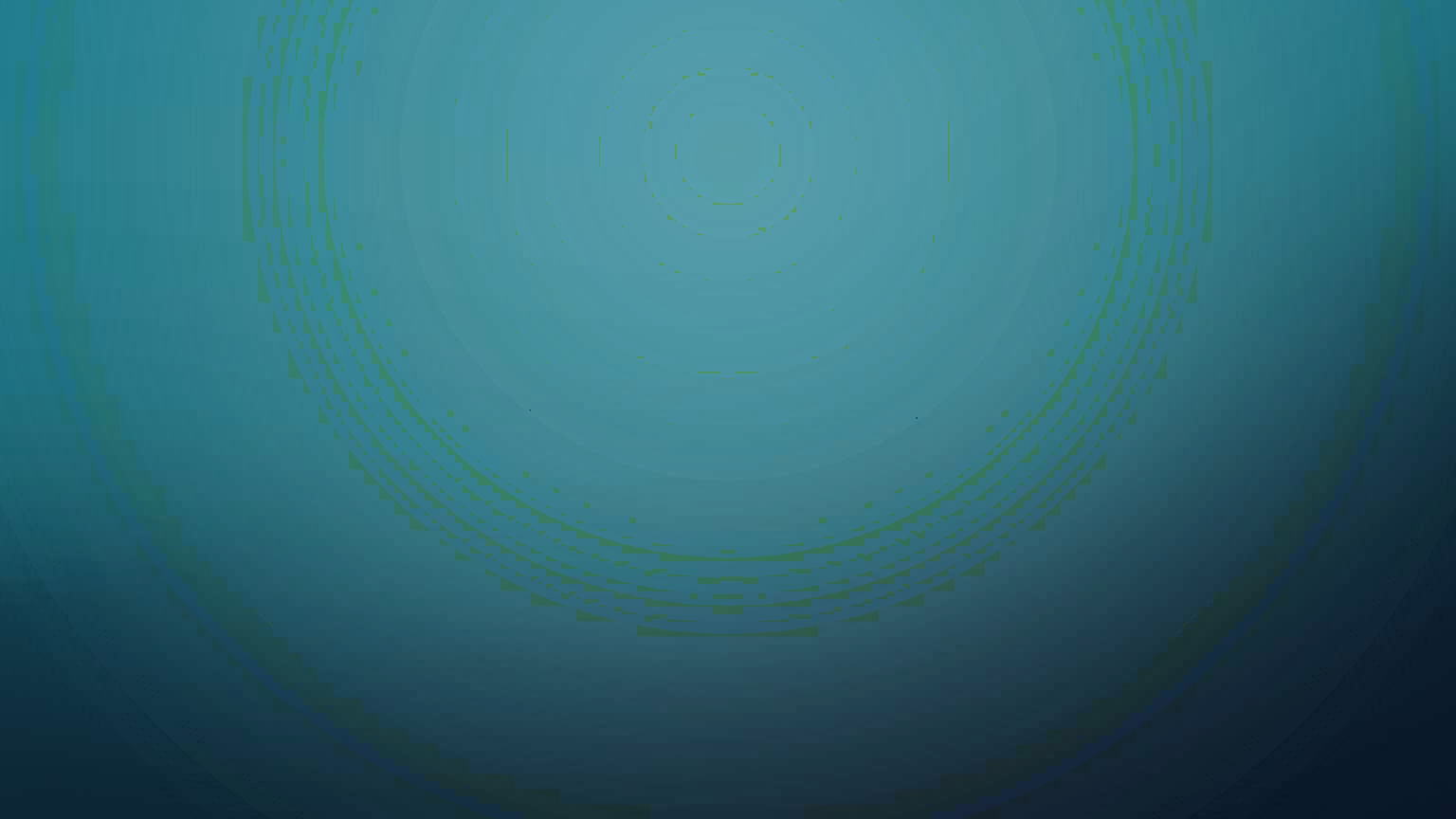
while True:

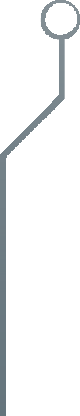
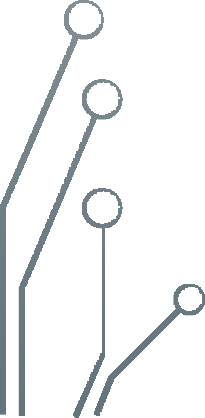
distance\_cm, distance\_inch = measure\_distance()

print("Distance: {} cm".format(distance\_cm)) print("Distance: {} inch".format(distance\_inch)) # lcd.print("Distance: {} cm".format(distance\_cm))

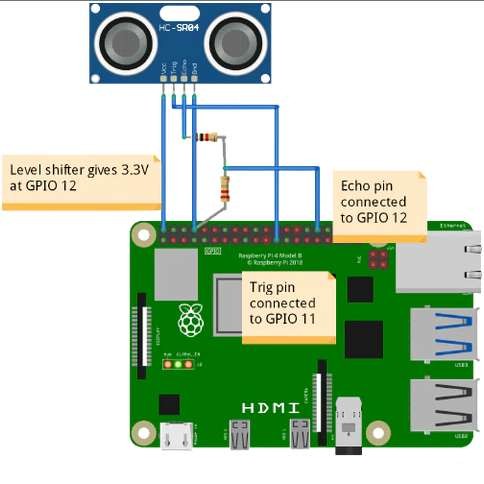
# lcd.print("Distance: {} inch".format(distance\_inch)) time.sleep(1)

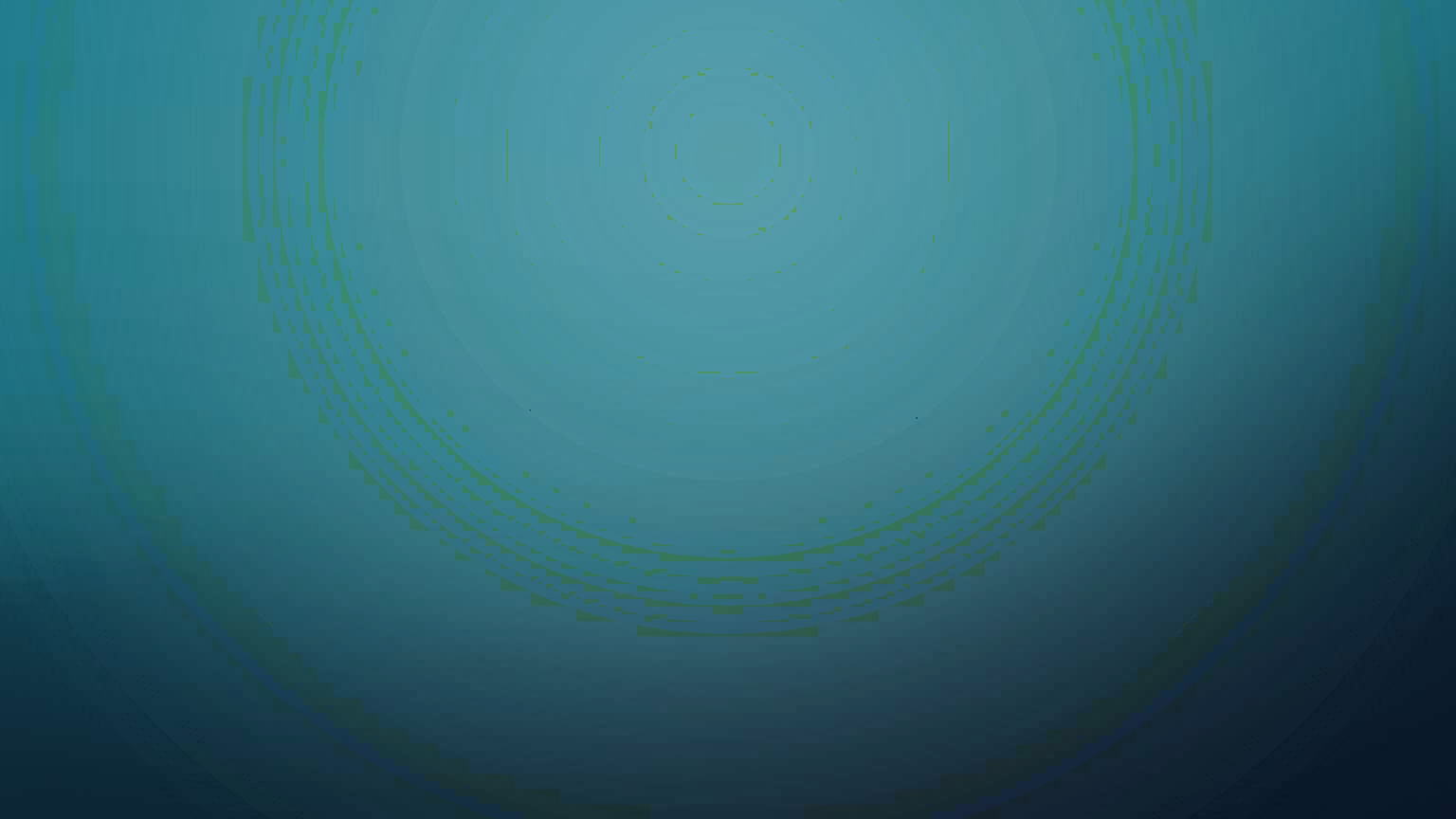
except KeyboardInterrupt: GPIO.cleanup()}

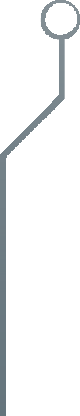
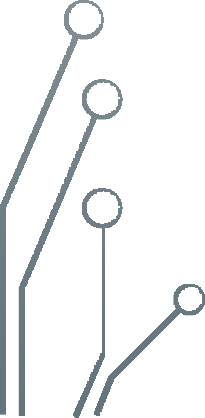


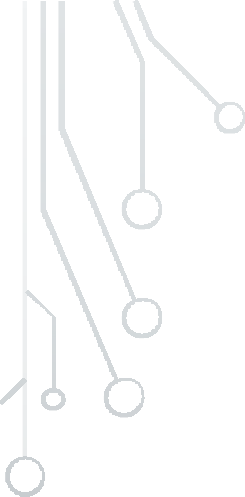


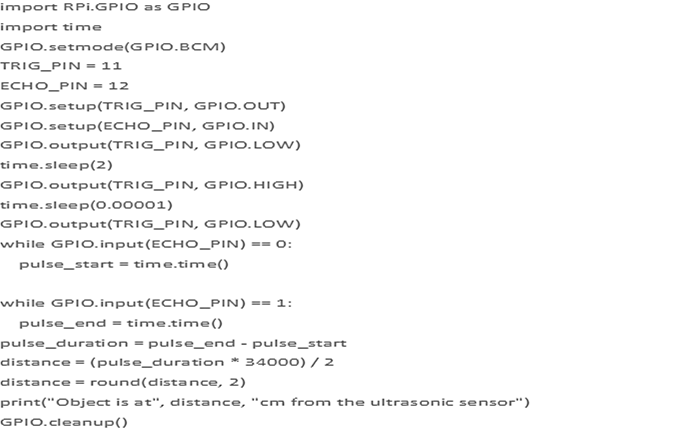
# RASPBERRY PI INTEGRATION







RASPBERRY PI CODE:





import RPi.GPIO as GPIO import time GPIO.setmode(GPIO.BCM) TRIG\_PIN = 11

ECHO\_PIN = 12 GPIO.setup(TRIG\_PIN, GPIO.OUT) GPIO.setup(ECHO\_PIN, GPIO.IN) GPIO.output(TRIG\_PIN, GPIO.LOW)

time.sleep(2) GPIO.output(TRIG\_PIN, GPIO.HIGH)

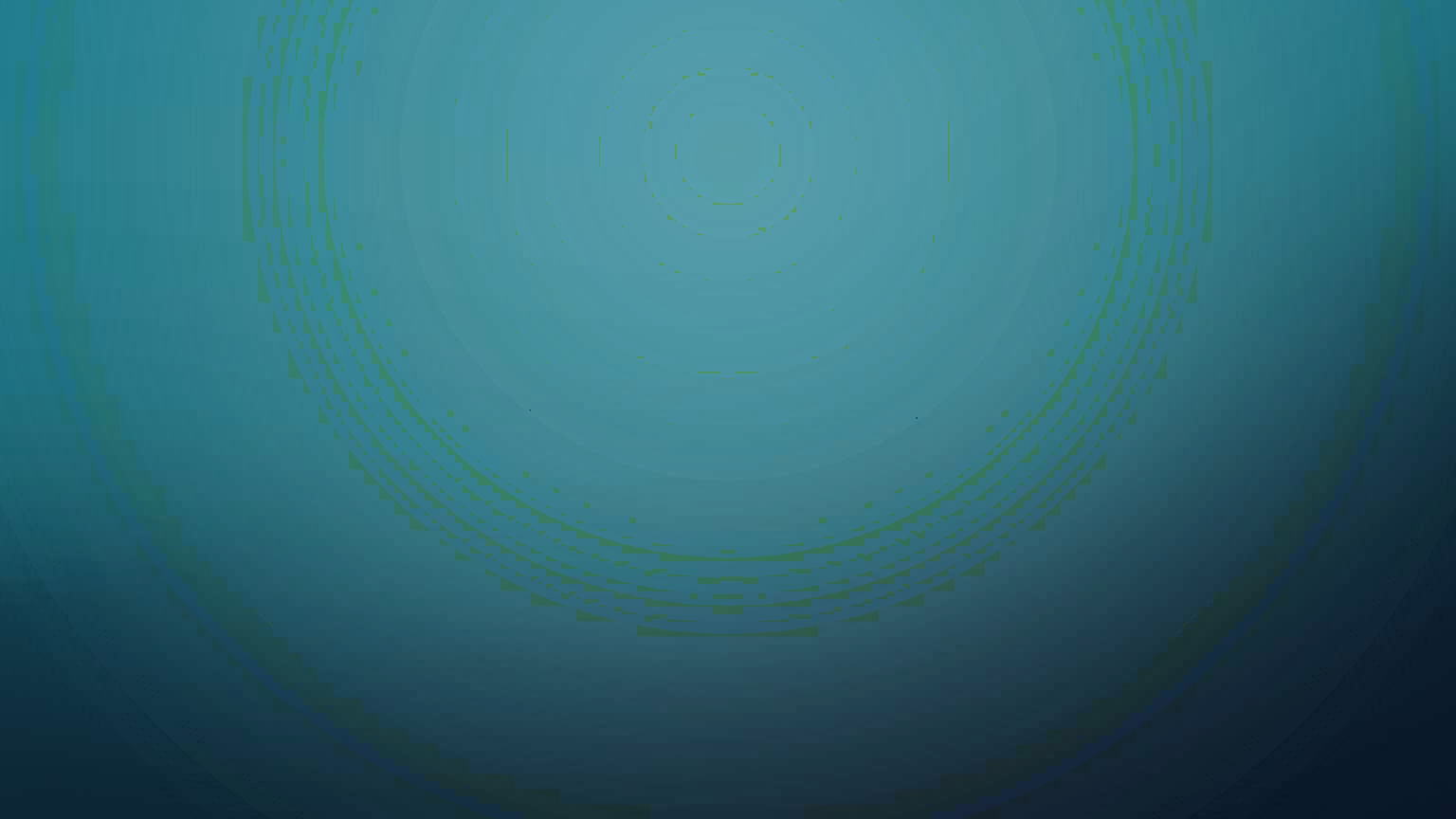
time.sleep(0.00001) GPIO.output(TRIG\_PIN, GPIO.LOW)

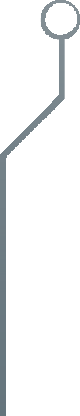
while GPIO.input(ECHO\_PIN) == 0: pulse\_start = time.time()

while GPIO.input(ECHO\_PIN) == 1: pulse\_end = time.time()

pulse\_duration = pulse\_end - pulse\_start distance = (pulse\_duration \* 34000) / 2 distance = round(distance, 2)

print("Object is at", distance, "cm from the ultrasonic sensor") GPIO.cleanup()





## PLATFORM:

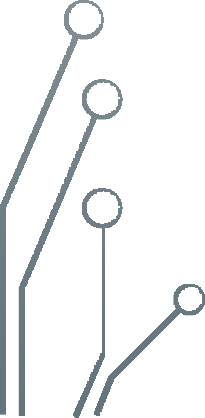
* **Operating System Platforms:** Examples include Windows, macOS, and Linux, which provide the

foundation for running various applications.

* **Cloud Computing Platforms:** Providers like Amazon Web Services (AWS), Microsoft Azure, and Google

Cloud offer cloud platforms that allow businesses to build, deploy, and scale applications in the cloud.

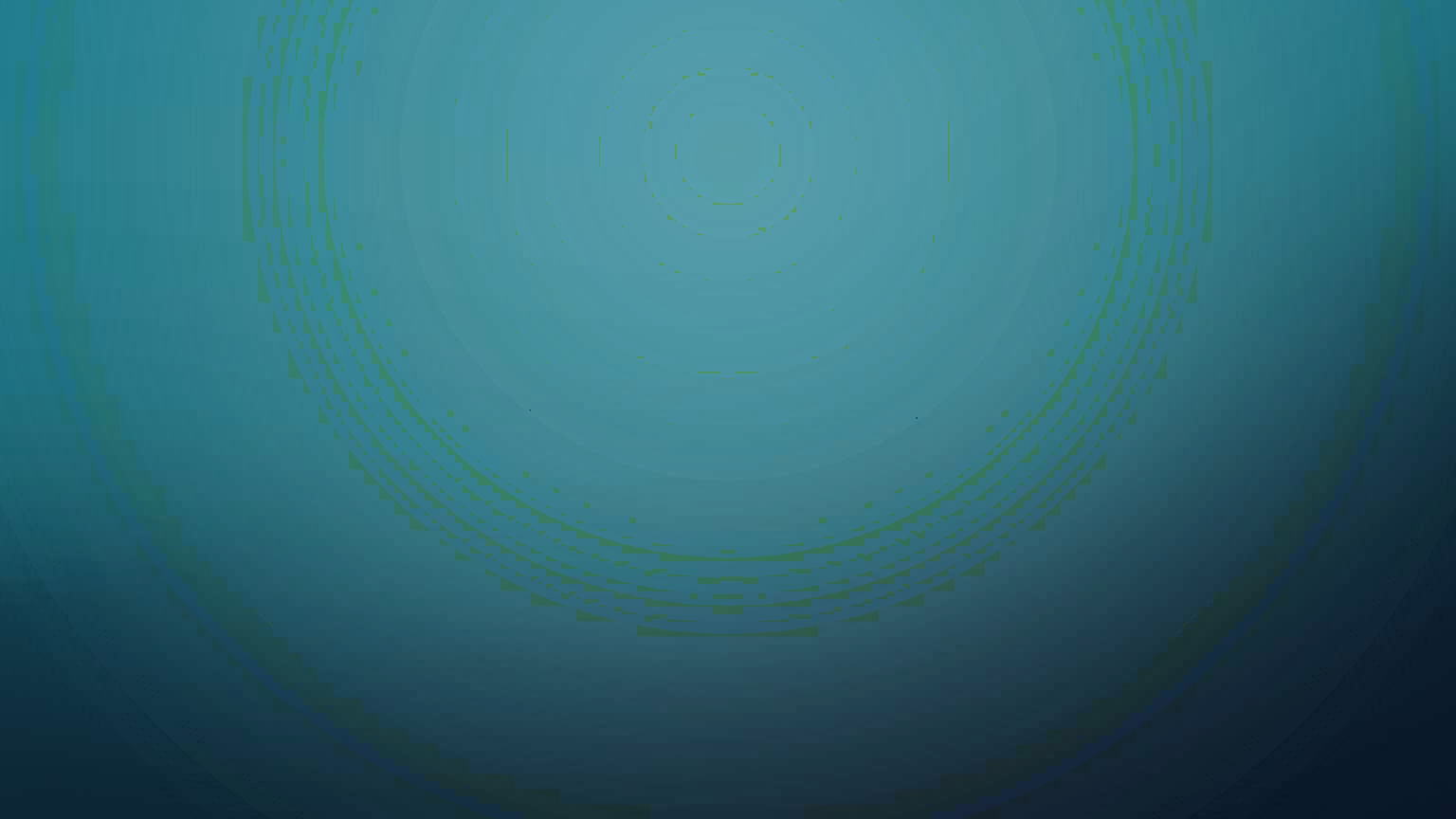
* **Mobile App Development Platforms**: iOS and Android are platforms for developing mobile

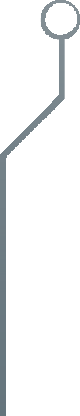


applications, each with its own set of development tools and environments.

* **Web Development Platforms:** Web development platforms like WordPress, Drupal, and Ruby on Rails

provide tools and frameworks for building websites and web applications.





## CODE FOR WEBSITE:

* + **Server-Side Python Code:**

from flask import Flask, render\_template, request

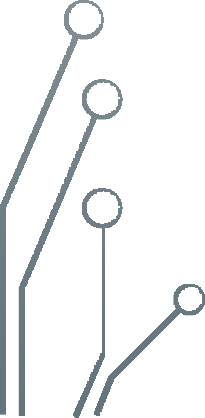
app = Flask( name )

# Store received sensor data in memory for demonstration. sensor\_data = []

@app.route('/') def index():

return render\_template('index.html', sensor\_data=sensor\_data) @app.route('/receive\_data', methods=['POST'])

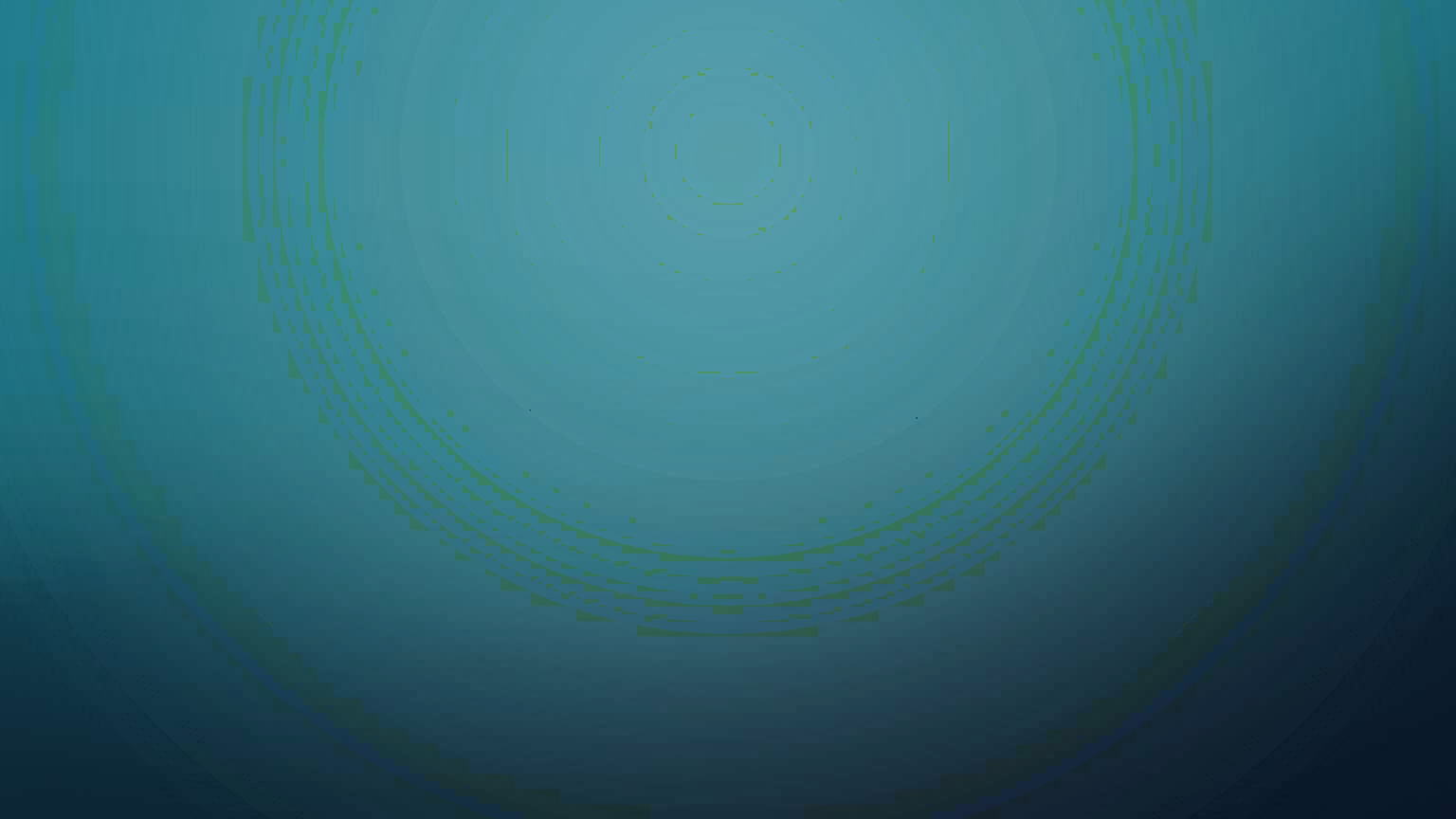
def receive\_data():

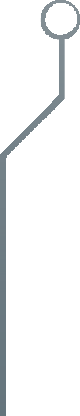


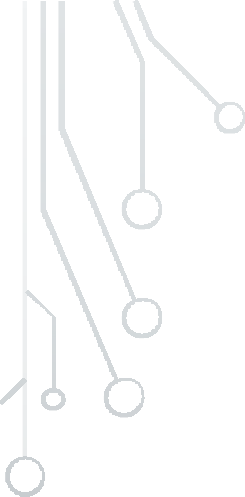
data = request.get\_json() sensor\_data.append(data) return 'Data received and stored'

if name == ' main ’:

app.run(debug=True)





* **HTML and Template for Web Page:**

**<!-- templates/index.html -->**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<title>Traffic Congestion Monitor</title>**

**</head>**

**<body>**

**<h1>Traffic Congestion Monitor</h1>**

**<div id="congestion-display">**

**<table>**

**<tr>**

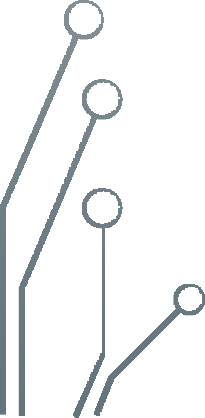
**<th>Timestamp</th>**

**<th>Congestion Level</th>**

**</tr>**

**{% for data in sensor\_data %}**

**<tr>**



**<td>{{ data.timestamp }}</td>**

**<td>{{ data.congestion\_level }}</td>**

**</tr>**

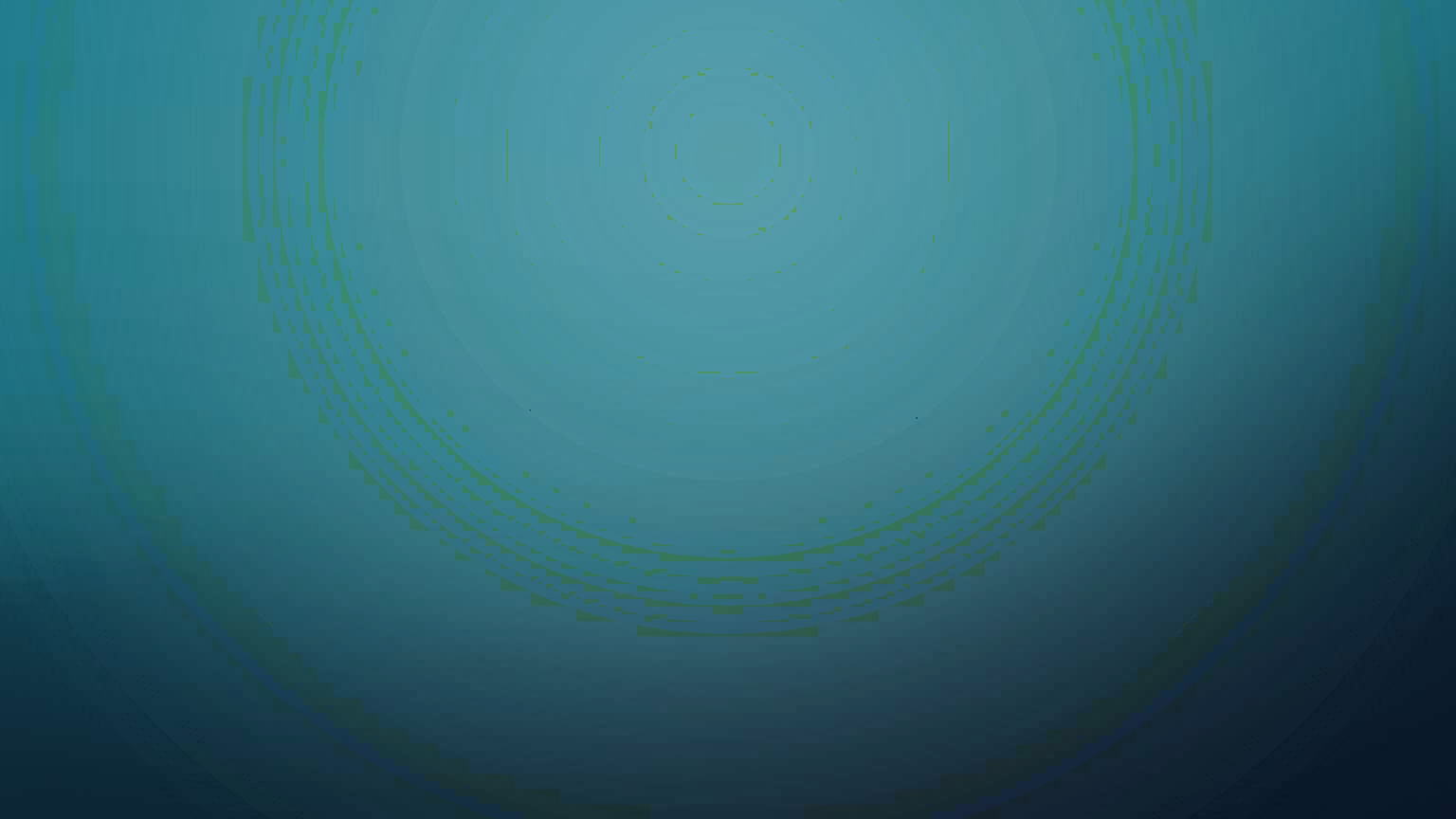
**{% endfor %}**

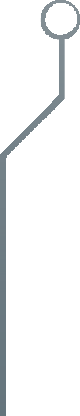
**</table>**

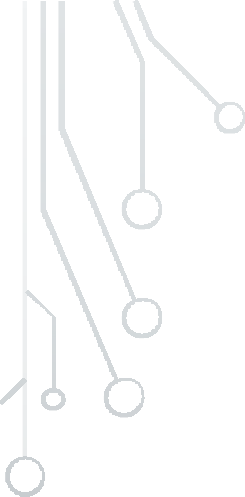
**</div>**

**</body>**

**</html>**





* + **Python Code for Bluetooth (Bluetooth LE):**

import pygatt

import json

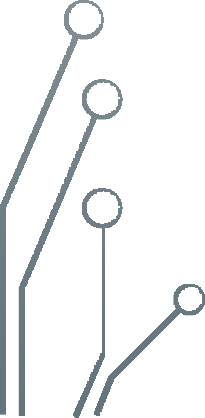
def send\_data\_via\_bluetooth(data): device\_address = 'your\_device\_address' client = pygatt.GATTToolBackend() client.start()

try:

device = client.connect(device\_address)

device.char\_write('your\_characteristic\_uuid', json.dumps(data).encode('utf-8'))

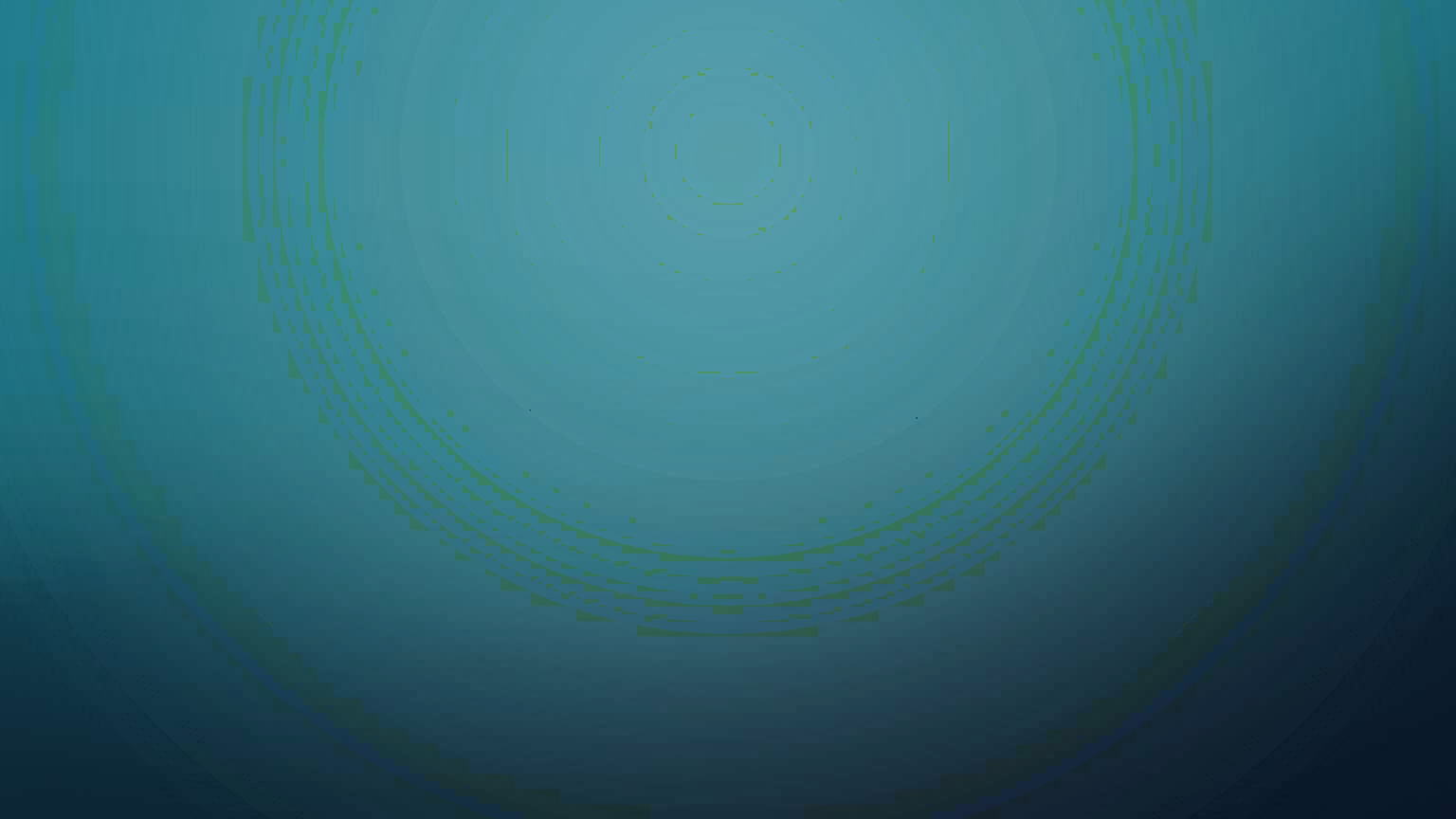
finally:

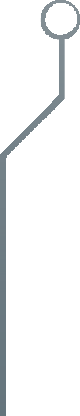


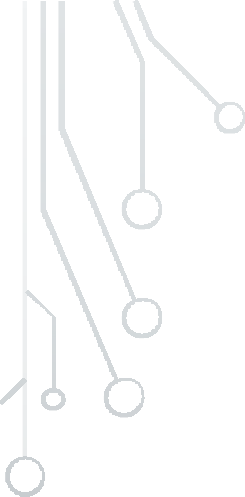
client.stop() # Usage:

data = {"timestamp": "2023-10-25 10:00:00", "congestion\_level": 30}

send\_data\_via\_bluetooth(data)





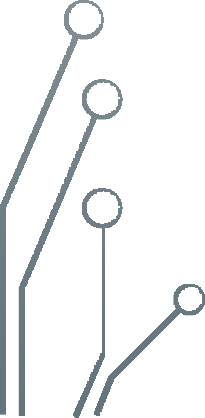
* **Python Code for Wi-Fi (HTTP Post Request):**

Import requests import json

def send\_data\_via\_wifi(data):

server\_url = ['http://your](http://your-server-ip-or-domain/receive_data%27)-[server-ip-or-domain/receive\_data'](http://your-server-ip-or-domain/receive_data%27)  headers = {'Content-Type': 'application/json'}

response = requests.post(server\_url, data=json.dumps(data),



headers=headers)

# Usage:

data = {"timestamp": "2023-10-25 10:00:00", "congestion\_level": 30} send\_data\_via\_wifi(data)

