IOT-CODE:

```
import RPi.GPIO as GPIO
import smtplib
from email import encoders
from email.mime.text import MIMEText
from email.mime.base import MIMEBase
from email.mime.multipart import MIMEMultipart
from picamera import PiCamera
from flask import Flask, render_template, Response
import time
import io
import logging
import socketserver
from threading import Condition
from http import server


PAGE="""\
<html>
<head>
<title>Web Streaming</title>
</head>
<body>
<center><h1>Web Streaming</h1></center>
<center><img src="stream.mjpg" width="640" height="480"></center>
</body>
```

```
</html>
"""

class StreamingOutput(object):
    def _init_(self):
        self.frame = None
        self.buffer = io.BytesIO()
        self.condition = Condition()

    def write(self, buf):
        if buf.startswith(b'\xff\xd8'):
            # New frame, copy the existing buffer's content and notify all
            # clients it's available
            self.buffer.truncate()
            with self.condition:
                self.frame = self.buffer.getvalue()
                self.condition.notify_all()
            self.buffer.seek(0)
        return self.buffer.write(buf)

class StreamingHandler(server.BaseHTTPRequestHandler):
    def do_GET(self):
        if self.path == '/':
            self.send_response(301)
            self.send_header('Location', '/index.html')
            self.end_headers()
        elif self.path == '/index.html':
            content = PAGE.encode('utf-8')
            self.send_response(200)
            self.send_header('Content-Type', 'text/html')
            self.send_header('Content-Length', len(content))
```

```
            self.end_headers()
            self.wfile.write(content)
        elif self.path == '/stream.mjpg':
            self.send_response(200)
            self.send_header('Age', 0)
            self.send_header('Cache-Control', 'no-cache, private')
            self.send_header('Pragma', 'no-cache')
            self.send_header('Content-Type', 'multipart/x-mixed-replace; boundary=FRAME')
            self.end_headers()
            try:
                while True:
                    with output.condition:
                        output.condition.wait()
                        frame = output.frame
                    self.wfile.write(b'--FRAME\r\n')
                    self.send_header('Content-Type', 'image/jpeg')
                    self.send_header('Content-Length', len(frame))
                    self.end_headers()
                    self.wfile.write(frame)
                    self.wfile.write(b'\r\n')
            except Exception as e:
                logging.warning(
                    'Removed streaming client %s: %s',
                    self.client_address, str(e))
        else:
            self.send_error(404)
            self.end_headers()

class StreamingServer(socketserver.ThreadingMixIn, server.HTTPServer):
    allow_reuse_address = True
    daemon_threads = True
```

```
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
buzzer=18
ir = 16
GPIO.setup(ir,GPIO.IN)
GPIO.setup(buzzer,GPIO.OUT)
print("Sensor is ready!")
while True:
   dectmotion = GPIO.input(ir)
   time.sleep(2)
   if dectmotion == 1:
      print("Motion Detected")
      GPIO.output(buzzer,GPIO.HIGH)
      picam = PiCamera()
      picam.rotation = 180
      picam.start_preview()
      picam.resolution = (960, 480)
      time.sleep(1)
      picam.capture("test.png")
      picam.stop_preview()
      picam.close()

      server = smtplib.SMTP('smtp.gmail.com',587)

      server.ehlo()
      server.starttls()
      server.ehlo()
```

```
        server.login('revanth.damisetty@gmail.com','loxzniiucosnspby')


        text = MIMEMultipart()
        text['From']='Revanth'
        text['To']='jatinwarner@gmail.com'
        text['Subject']='SMTP in action'


        text.attach(MIMEText("Motion Detected"))



        filename = 'test.png'
        attachment = open(filename, 'rb') #Here rb denotes read bytes becz we are dealing with image not text
        p = MIMEBase('application', 'octet-stream') #for processing image data
        p.set_payload(attachment.read())
        encoders.encode_base64(p)
        p.add_header('Content-Disposition', f'attachment; filename = {filename}')
        text.attach(p)


        server.sendmail("revanth.damisetty@gmail.com","jatinwarner@gmail.com",text.as_string())
        print("Mail Sent")
        try:
           with PiCamera(resolution='640x480', framerate=24) as camera:
              camera.rotation = 180
              output = StreamingOutput()
              camera.start_recording(output, format='mjpeg')
              GPIO.output(buzzer,GPIO.LOW)
              app = Flask(_name_)
              @app.route('/')
              def index():
                 return render_template('index.html')
              def generate():
```

```
            while True:
                frame = np.empty((camera.resolution[1] * camera.resolution[0] * 3,), dtype=np.uint8)
                camera.capture(frame, 'bgr', use_video_port=True)
                if motion_detection.detect_motion(frame):
                    rotate_servo()
                ret, jpeg = cv2.imencode('.jpg', frame)
                yield (b'--frame\r\n'
                       b'Content-Type: image/jpeg\r\n\r\n' + jpeg.tobytes() + b'\r\n')
        @app.route('/video_feed')
        def video_feed():
            return Response(generate(),
                            mimetype='multipart/x-mixed-replace; boundary=frame')
        address = ('', 8000)
        server = StreamingServer(address, StreamingHandler)
        server_thread = server.serve_forever()
        app.run(host='192.168.168.30 ', port=8000, debug=False, threaded=True)
    except KeyboardInterrupt:
        GPIO.cleanup()
        server.shutdown()
        server.server_close()

else:
    print("motion not dectected")
    GPIO.output(buzzer,GPIO.LOW)


GPIO.cleanup()
```