EEE 598 - Programming for the Internet of Things

Prof. Dr. Tejaswi Linge Gowda

Nishad Kale (ASU ID: 1226362919)
Rahil Hastu (ASU ID:1225356132)

December 3, 2023

Smart Ambient Lighting and Motion-Activated System

The embedded code showcases a comprehensive IoT project focused on smart ambient lighting and motion detection using an ESP32 microcontroller, a PIR sensor, and a NeoPixel LED strip. The program begins by initializing essential components, such as the WiFi connection, PIR sensor pin, and server details for data transmission. During execution, the system continuously monitors the PIR sensor for motion detection. Upon detecting motion, the NeoPixel strip illuminates in dynamic RGB sequences, providing visual feedback. Simultaneously, a timestamp of the motion event is captured using `millis()` and sent to a server through an HTTP GET request. The server's response is then printed to the Serial Monitor for monitoring purposes. Conversely, in the absence of motion, the NeoPixel strip is cleared.

This IoT project, driven by motion detection and user preferences, caters to the unique needs of elderly patients in hospitals. By utilizing PIR sensors, the system responds to their physical motion, triggering lighting and generating signals for nurses to assist and log patient data efficiently. Similarly, in educational settings like libraries, the system enables students and library management to remotely understand whether a certain study room is available or not since data will be available on the server by motion detection. This dual functionality demonstrates the versatility of the project, offering practical solutions in healthcare and education by enhancing security, resource utilization, and overall user experience.

Overview

This IoT project is a sophisticated integration of cutting-edge technologies, incorporating a Passive Infrared (PIR) sensor designed for motion detection, a WS2812 RGB LED for visual feedback, and an ESP32 microcontroller for seamless control. The primary functionality revolves around the detection of motion using the PIR sensor, triggering a series of responses facilitated

by the ESP32. In response to detected motion, the system activates lighting for practical purposes, simultaneously providing visual cues through the WS2812 RGB LED.

This dynamic interplay of motion detection and LED feedback allows for the creation of versatile and interactive ambient lighting scenarios. The project's versatility makes it a valuable asset in various applications, from enhancing home automation systems to fortifying security measures or contributing to environmental control. With its comprehensive features, this IoT project represents a holistic approach to intelligent home systems, adding practical functionality and aesthetic appeal to modern living spaces.

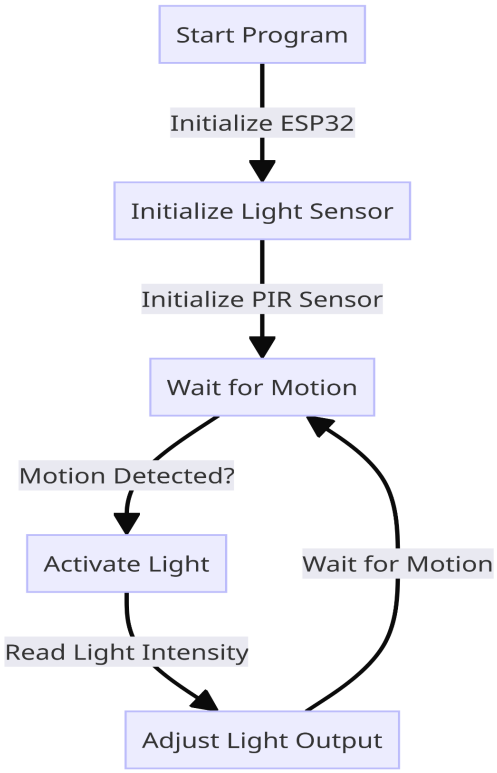


Image 1.1 System Architecture

Methodology

In this IoT system, a PIR sensor diligently monitors its designated area for motion, promptly signaling the ESP32 microcontroller upon detection. The ESP32, armed with predefined settings, processes the information and commands the WS2812 RGB LED to emit specific lighting patterns or colors, providing dynamic ambient lighting scenarios. Notably, the system's versatility extends to remote control and monitoring through an AWS EC2 instance, enhancing user accessibility. Meanwhile, all activity data is meticulously logged in a MongoDB database,

paving the way for detailed analysis. This holistic integration of motion detection, responsive lighting, and remote connectivity positions the project as a comprehensive solution for applications ranging from home automation to security, showcasing the synergy of real-time control and data-driven insights.

The embedded code showcases a comprehensive IoT project focused on smart ambient lighting and motion detection using an ESP32 microcontroller, a PIR sensor, and a NeoPixel LED strip. The program begins by initializing essential components, such as the WiFi connection, PIR sensor pin, and server details for data transmission. During execution, the system continuously monitors the PIR sensor for motion detection. Upon detecting motion, the NeoPixel strip illuminates in dynamic RGB sequences, providing visual feedback. Simultaneously, a timestamp of the motion event is captured using `millis()` and sent to a server through an HTTP GET request. The server's response is then printed to the Serial Monitor for monitoring purposes. Conversely, in the absence of motion, the NeoPixel strip is cleared.

```
void loop() {
  Serial.println("Sample Motion Sensing");
  int motion = digitalRead(pirPin);

  if (motion == HIGH) {

    for (int i = 0; i < 2; i++) {
      for (int j = 0; j < 2; j++) {
        for (int k = 0; k < 2; k++) {
          strip.setPixelColor(0, strip.Color(i * 255, j * 255, k * 255));
          strip.setPixelColor(1, strip.Color(i * 255, j * 255, k * 255));
          strip.setPixelColor(2, strip.Color(i * 255, j * 255, k * 255));
          strip.setPixelColor(3, strip.Color(i * 255, j * 255, k * 255));


          strip.show();
          delay(500);
        }
      }
    }

    unsigned long currentTime = millis();

    String url = String(serverName) + "?motion_time=" + String(currentTime);
    Serial.println(url);

    String response = httpGETRequest(url.c_str());
    Serial.println(response);
  } else {
    strip.clear();
    strip.show();
  }

  delay(1000);
}
```

Figure 1.2 ESP32 Code

```
var express = require("express");
var app = express();
var bodyParser = require("body-parser");
var errorHandler = require("errorhandler");
var methodOverride = require("method-override");
var MS = require("mongoskin");
var hostname = process.env.HOSTNAME || "localhost";
var port = 8080;
            Blame Rahil Hastu (4 days ago)
var motion_time;

var db = MS.db("mongodb://127.0.0.1:27017/sensorData");

app.get("/", function (req, res) {
  res.redirect("index.html");
});

app.get("/sendData", function (req, res) {
  motion_time = req.query.motion_time;
  req.query.currentDate = new Date();
  console.log("Hello the server is being called!");
  console.log(req.query);

  db.collection("data").insertOne(req.query, function (result, err) {
    console.log(result, err);
    res.send("1");
  });
});

app.get("/getData", function (req, res) {
  var startDate = req.query.startDate;
  var endDate = req.query.endDate;

  var query = {};
  if (startDate !== "undefined" && endDate !== "undefined") {
    query = {
      currentDate: {
        $gte: new Date(startDate),
        $lte: new Date(endDate),
      },
    };
  }
```

Figure 1.3 Server Code

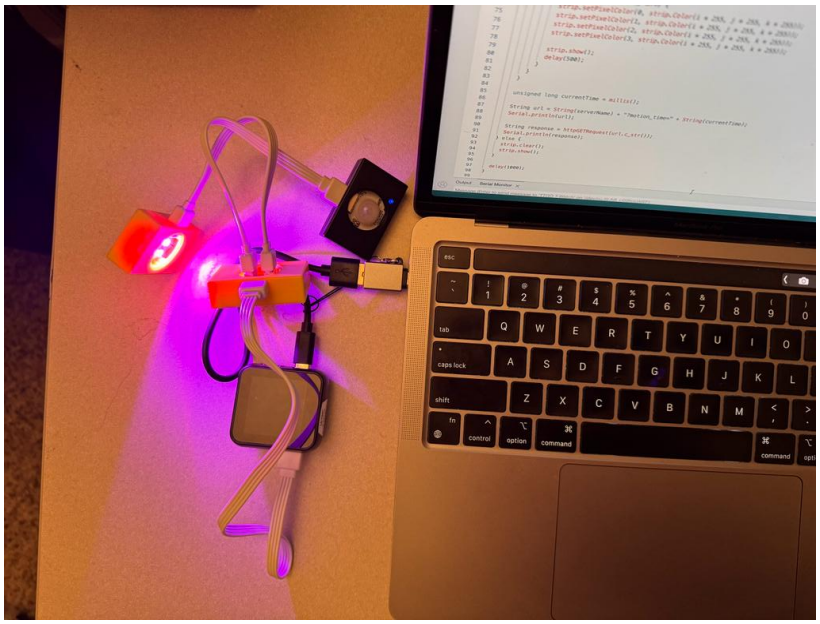
The clear structure of the program, from hardware initialization to server communication, aligns with key IoT concepts taught during the course. This project serves as a tangible demonstration of the integration of motion detection, responsive lighting, and remote connectivity, emphasizing the synergy of real-time control and data-driven insights. Additionally, the implementation of a motion-activated system introduces possibilities for diverse applications, including enhancing home automation, fortifying security measures, and contributing to environmental control. This two-fold approach, combining both functionality and aesthetics, positions the project as a holistic solution for intelligent home systems, showcasing its potential societal impact in improving user experiences and optimizing resource utilization.

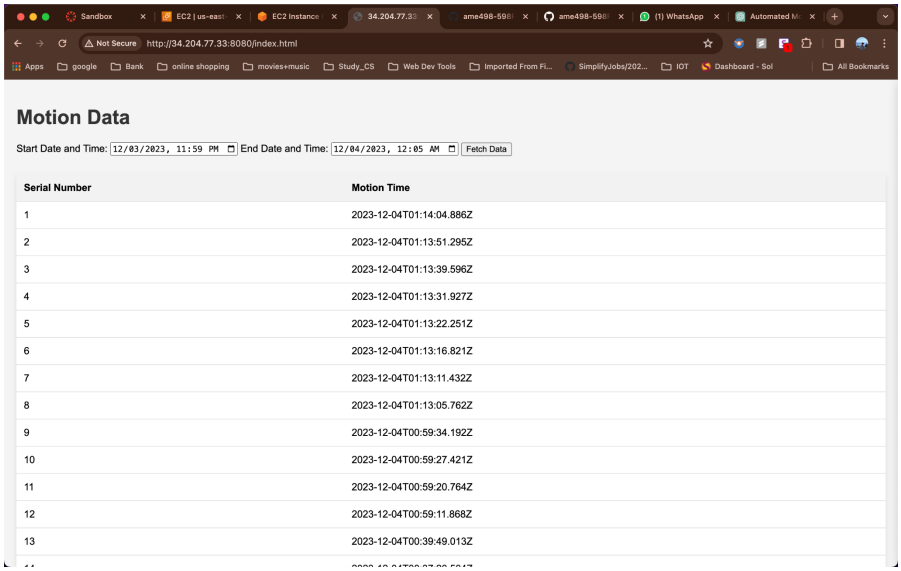
The provided Node.js code establishes a server using the Express framework to facilitate communication between an IoT device and a MongoDB database. The server, hosted on the specified port (8080), employs various middleware, including bodyParser for handling incoming data, errorHandler for error management, and methodOverride for supporting HTTP methods. The MongoDB connection is established through Mongoskin, connecting to a database named "sensorData." The server serves static files from the "public" directory, with the default route redirecting to an "index.html" page.

The server exposes two crucial endpoints. The "/sendData" endpoint receives HTTP GET requests containing motion timestamps from an IoT device, such as an ESP32. The received data, including the motion timestamp and the current date, is logged to the server console and stored in the MongoDB database under the "data" collection. The "/getData" endpoint allows querying the database for motion data within specified date ranges, responding with a sorted array of results.

This server-side code complements the ESP32 device code, creating a complete system for motion detection, visual feedback, and remote data storage. The server's ability to receive and store motion data in MongoDB, as well as retrieve historical data, enhances the overall functionality and utility of the IoT project. The combination of the ESP32 device code and this server-side script showcases a seamless integration of hardware, server, and database components, contributing to a comprehensive solution for motion-sensing applications.

Results





This IoT project successfully integrates an ESP32-based motion-activated system with a Node.js server and MongoDB, creating a responsive ambient lighting solution. The ESP32 device, equipped with a PIR sensor and NeoPixel LED strip, detects motion and provides dynamic visual feedback. The Node.js server serves as the central hub, receiving motion timestamps from the IoT device and storing them in a MongoDB database. A key achievement of this project is the implementation of a user-friendly dashboard that compiles and displays timestamps whenever motion is detected. This dashboard offers valuable insights into motion patterns over time, enhancing the overall utility of the system. The attached images visually capture the project's dynamic lighting responses and showcase the interface of the motion

timestamp dashboard, illustrating the seamless integration of hardware and server components. Together, these elements contribute to a comprehensive and versatile IoT framework suitable for various applications, from home automation to security and beyond.

Feedback

- **How have the readings and workshops in class influenced your work?**
  The readings and workshops in our class have been instrumental in shaping my work and providing a comprehensive understanding of IoT concepts. Learning about sensors, microcontrollers, and cloud services has significantly influenced my project approach, allowing me to integrate cutting-edge technologies like the ESP32 microcontroller and AWS EC2 instance. This exposure has expanded my technical toolkit and empowered me to create more robust and versatile systems.

- **Has the work in class shifted the way you envision your research goals?**
  The work in class has indeed shifted my perspective on research goals. Discussions on real-world applications, practical implementations, and case studies have motivated me to focus on projects addressing tangible challenges. The emphasis on user-friendly interfaces and accessibility features, particularly in remote monitoring and control, has inspired me to incorporate these elements into my future projects. The class has broadened my understanding of the societal impact of IoT, pushing me to explore how these technologies can contribute to sustainability and efficiency.

- **What are your plans for your future research?**
  Looking forward, my plans involve collaborating with fellow researchers and staying up to date on the latest developments in the field are integral components of my future research plans, as I strive to contribute meaningfully to the evolving landscape of IoT applications.