***PROJECT REPORT***

PUBLIC TRANSPORT OPTIMIZATION:

It is defined as the data on the number of public transport users,collected with IoT sensors,allows routes to be optimized or new ones to be designed.These sensors can be in the vehicle or also in the security cameras.

By defining these project objectives and detailing the IoT sensor design, real-time transit information platform, and integration approach, We will have a comprehensive roadmap for improving public transportation services with real-time data and information for passengers.

*Technologies used:*

HTML:

IoT devices often have web-based user interfaces that allow users to interact with and control them remotely through a web browser. HTML is used to create the web pages and user interfaces that users access to monitor and control their IoT devices.

HTML can be a useful tool for creating user interfaces and visualizing data in IoT applications, the core functionality and communication between IoT devices often rely on other technologies and protocols, such as MQTT, CoAP, HTTP, and various networking protocols. HTML is just one component of the overall IoT ecosystem, and its use is typically limited to the presentation layer of IoT application

***C++:***

Designwise, C++ lends itself to embedded development because the language lies in between higher-level software and hardware, allowing you to access and control hardware directly without cutting the advantages of a high-level language. It’s particularly useful for hardware that will need to be around for a while, as programs written in C++ can operate for a long time due to the language’s high stability.

Here are some more reasons for choosing C++ to build smart IoT applications.

Low-level manipulation: Since C++ is closely related to C, which is a procedural language closely linked to the machine language, C++ provides low-level manipulation of data at a specific level. That is the reason why Embedded systems and compilers are developed with the help of C++.

Multi-paradigm: C++ is a multi-paradigm programming language. The phrase “Paradigm” refers to programming style. It involves the logic, structure, and procedure of the program. The three paradigms of C++ are generic, imperative, and object-oriented. Generic programming means using a single approach to serve several objectives. Imperative programming, on the other hand, means to use the statements that change a program’s status.

Memory management: C++ gives the developer the capability of total control over memory management. It can be an asset and a liability as it increases the engagement of the user to manage memory rather than being managed itself by the Garbage collector. This method is implemented using DMA (Dynamic memory allocation) using pointers.

Object-oriented: The greatest advantage of C++ is the feature of object-oriented programming that involves concepts like classes, inheritance, polymorphism, data abstraction, and encapsulation that enable code reusability and makes a program even more secure. It also helps us deal with real-world problems by interpreting data as an object. C was previously lacking this feature and so it was created, proving to be highly significant. This feature gave birth to numerous job prospects and the future of c++ programming and development.

Portability: C++ offers the feature of portability also known as platform independence which lets the user run the same program on different operating systems or interfaces easily. For example, you write a program in LINUX OS, and for some reason, you switch to Windows, in that case, you would be able to run the same program in windows as well without any restraint.

***JAVASCRIPT:***

It is the most popular lightweight, interpreted compiled programming language. It can be used for both Client-side as well as Server-side developments. JavaScript is also known as a scripting language for web pages.

However, it has become increasingly popular for building Internet of Things (IoT) applications. IoT refers to the connection of everyday devices to the internet, allowing them to communicate with each other and with us. In this blog, we will explore how to use JavaScript to build IoT applications, including controlling hardware devices such as sensors and actuators, and how to use popular platforms such as Raspberry Pi and Arduino.

To build an IoT application with JavaScript, we need to use a hardware platform that supports it. Two popular hardware platforms that support JavaScript are Raspberry Pi and Arduino.

const sensor = require('ds18b20');

// Replace with your sensor ID

const sensorId = '28-00000abcdefg';

sensor.temperature(sensorId, (err, value) => {

if (err) {

console.error(err);

return;

}

console.log(`Temperature: ${value}°C`);

});

**Temperature and humidity sensor:**

#include <DHT.h>;

//Constants

#define DHTPIN 7 // what pin we're connected to

#define DHTTYPE DHT22 // DHT 22 (AM2302)

DHT dht(DHTPIN, DHTTYPE); //// Initialize DHT sensor for normal 16mhz Arduino

//Variables

int chk;

float hum; //Stores humidity value

float temp; //Stores temperature value

void setup()

{

Serial.begin(9600);

dht.begin();

}

void loop()

{

delay(2000);

//Read data and store it to variables hum and temp

hum = dht.readHumidity();

temp= dht.readTemperature();

//Print temp and humidity values to serial monitor

Serial.print("Humidity: ");

Serial.print(hum);

Serial.print(" %, Temp: ");

Serial.print(temp);

Serial.println(" Celsius");

delay(1000); //Delay 2 sec.

}

**Monitor sensor:**

var Accessory = require('../').Accessory;

var Service = require('../').Service;

var Characteristic = require('../').Characteristic;

var uuid = require('../').uuid;

// here's a fake temperature sensor device that we'll expose to HomeKit

var FAKE\_MOTIONSENSOR = {

isPresent: false,

getState: function() {

console.log("Getting the current state!");

return FAKE\_MOTIONSENSOR.isPresent;

},

randomState: function() {

// randomize temperature to a value between 0 and 100

FAKE\_MOTIONSENSOR.isPresent = !FAKE\_MOTIONSENSOR.isPresent;

}

}

// Generate a consistent UUID for our Temperature Sensor Accessory that will remain the same

// even when restarting our server. We use the `uuid.generate` helper function to create

// a deterministic UUID based on an arbitrary "namespace" and the string "temperature-sensor".

var sensorUUID = uuid.generate('hap-nodejs:accessories:motion-sensor');

// This is the Accessory that we'll return to HAP-NodeJS that represents our fake lock.

var sensor = exports.accessory = new Accessory('BedRoom Motion Sensor', sensorUUID);

// Add properties for publishing (in case we're using Core.js and not BridgedCore.js)

sensor.username = "11:33:33:AE:53:3A";

sensor.pincode = "031-45-154";

// Add the actual TemperatureSensor Service.

// We can see the complete list of Services and Characteristics in `lib/gen/HomeKitTypes.js`

sensor

.addService(Service.MotionSensor)

.getCharacteristic(Characteristic.MotionDetected)

.on('get', function(callback) {

// return our current value

callback(null, FAKE\_MOTIONSENSOR.getState());

});

// randomize our temperature reading every 3 seconds

// PIRSensor.watch(function(error, input) {

// if (error) {

// throw error;

// }

// console.log(input);

//

// })

var mqtt = require('mqtt');

var client = mqtt.connect('mqtt://localhost');

client.on('connect', function() {

client.subscribe('MOTION');

client.on('message', function(topic, message) {

if (message == 'ON') {

//publishMqtt('LIGHTOFF', '^^ motion from PIR');

sensor

.getService(Service.MotionSensor)

.setCharacteristic(Characteristic.MotionDetected, true);

console.log(message.toString());

} else {

sensor

.getService(Service.MotionSensor)

.setCharacteristic(Characteristic.MotionDetected, false);

console.log(message.toString());

}

});

});