**CHAPTER 1**

**INTRODUCTION**

A counter that can change its state in either direction, under control of an up–down selector input, is known as an [up–down counter](http://engineersgarage.com/content/microcontroller-at89c51-based-down-counter). The circuit given here can count numbers in up and down modes depending upon the state of the selector. It can be used to count the number of persons entering a hall in the up mode at entrance gate. In the down mode, it can count the number of persons leaving the hall by decrementing the count at exit gate. It can also be used at gates of parking areas and other public places.

The project consists of two phases: the first phase consists of finding the count value and the second phase consists of uploading the count value to a cloud database.

The circuit in the first phase is divided into three parts: sensor, controller and counter display. The sensor would observe an interruption and provide an input to the controller which would run the counter in up/down mode depending upon the selector setting. The same count is displayed on screen.

The second phase updates the database in real-time, thus giving us the count value at any time, which can be accessed or checked by the user who has the specific user id and secure key to the database.

**BLOCK DIAGRAM DESCRIPTION**

From the block diagram it is clear that the sensor pairs are placed face to face so that the IR radiations from IR LED are continuously received by photo detector. In this circuit, two [infrared (IR) sensor](http://engineersgarage.com/content/infrared-led) modules are used each for up and down counting, respectively.

Whenever an interruption is observed by the first IR sensor, it increments the counter value. Similarly, when the second sensor detects an obstacle, the count is decremented. Once the count value is found it is first locally updated i.e., the count variable in NodeMCU is incremented or decremented, then this value is uploaded to the firebase database in real time .

**CHAPTER 2**

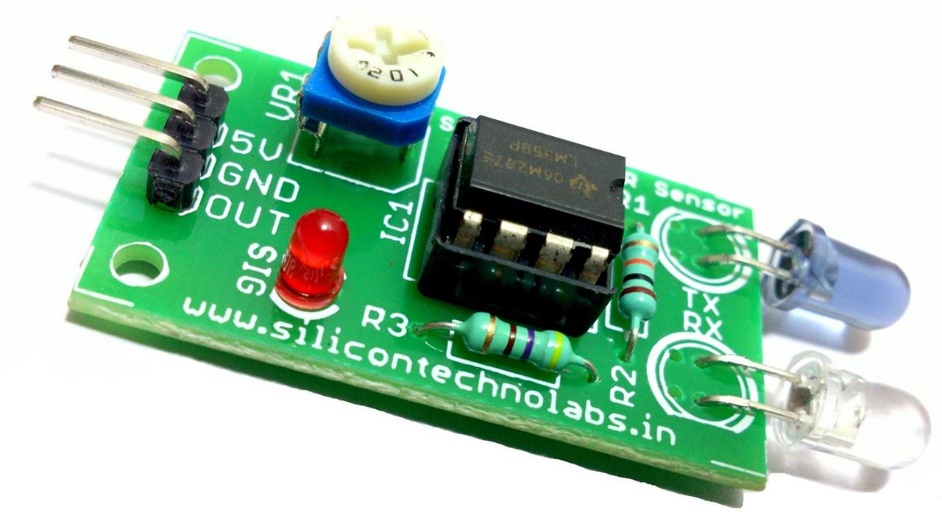
**COMPONENTS USED**

1. NodeMCU LoLin ESP8266 CH340G Wi-Fi Development Board Module with USB Cable

NodeMCU is an open source IoT platform. It includes firmware which runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which is based on the ESP-12 module. The term "NodeMCU" by default refers to the firmware rather than the development kits. The firmware uses the Lua scripting language. It is based on the eLua project, and built on the Espressif Non-OS SDK for ESP8266. It uses many open source projects, such as lua-cjson and SPIFFS.

We are using Arduino platform for compiling and uploading the C program code into the NodeMCU for execution. It is an Open-source electronic prototyping platform enabling users to create interactive electronic objects.

2. Infrared IR Sensor



An infrared sensor is an electronic device that emits in order to sense some aspects of the surroundings. An IR sensor can measure the heat of an object as well as detects the motion. These types of sensors measures only infrared radiation, rather than emitting it that is called as a passive IR sensor. Usually in the infrared spectrum, all the objects radiate some form of thermal radiations. These types of radiations are invisible to our eyes, that can be detected by an infrared sensor. The emitter is simply an IR LED (Light Emitting Diode) and the detector is simply an IR photodiode which is sensitive to IR light of the same wavelength as that emitted by the IR LED. When IR light falls on the photodiode, the resistances and these output voltages, change in proportion to the magnitude of the IR light received.

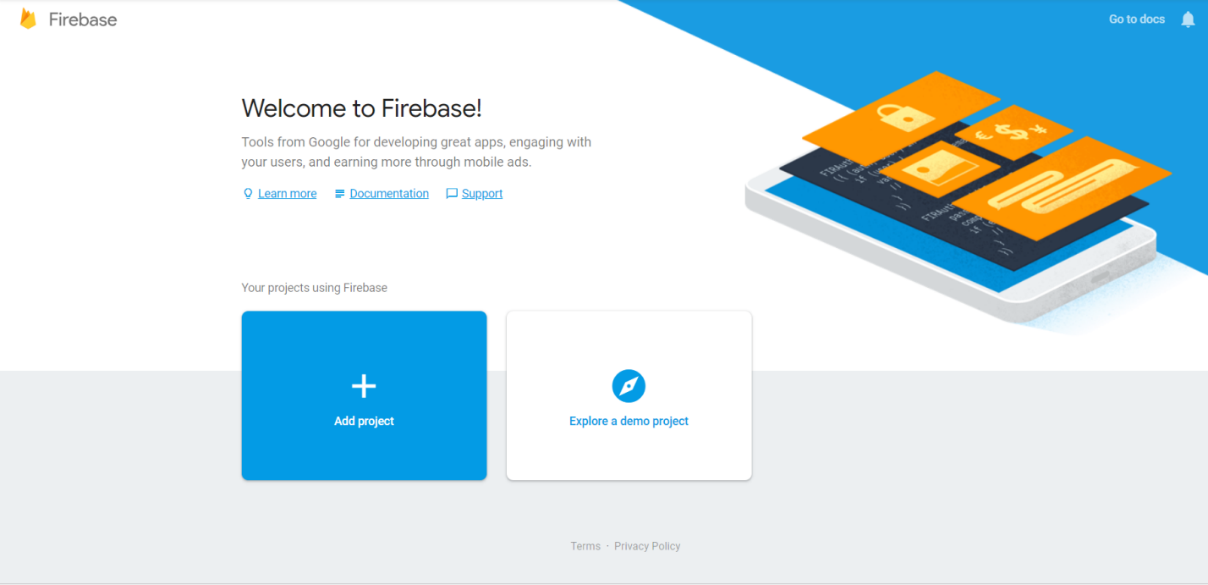
This circuit comprises of the following components

* [LM358 IC](https://www.elprocus.com/op-amp-ics-pin-configuration-features-working/) 2 IR transmitter and receiver pair
* Resistors of the range of kilo ohms.
* Variable resistors.
* LED (Light Emitting Diode).

**CHAPTER 3**

**FIREBASE:**

Firebase is a scalable, real-time support for web-based application. It allows developers to build rich, collaborative applications without the hassle of managing servers or writing server-side code. It is a mobile and web application platform with tools and infrastructure designed to help developers build high-quality apps. Envolve provided developers an API that let them integrate online chat into their websites. But they founded that the service was being used to pass application data that wasn't chat messages. Developers were using Envolve to sync application data such as game state in real-time across their users.



**HOW DOES FIREBASE WORK**

* Firebase is essentially a real time database. The data appears as JSON files and allows real time changes to occur on the connected client side. When you build cross-platform apps using iOS, Android, JavaScript SDKs, your clients end up getting all the data that was updated.
* Firebase has support for the web, iOS, OS X, and Android clients. It has a Node.js and a Java library designed for server-side use.
* Developers install firebase by including a library in their applications. This library provides a data structure that is automatically synchronized between all of your clients and with our servers.
* If one client changes a piece of data, every other client observing the same piece of data will be updated as well within milliseconds.

**FIREBASE AUTHENTICATION**

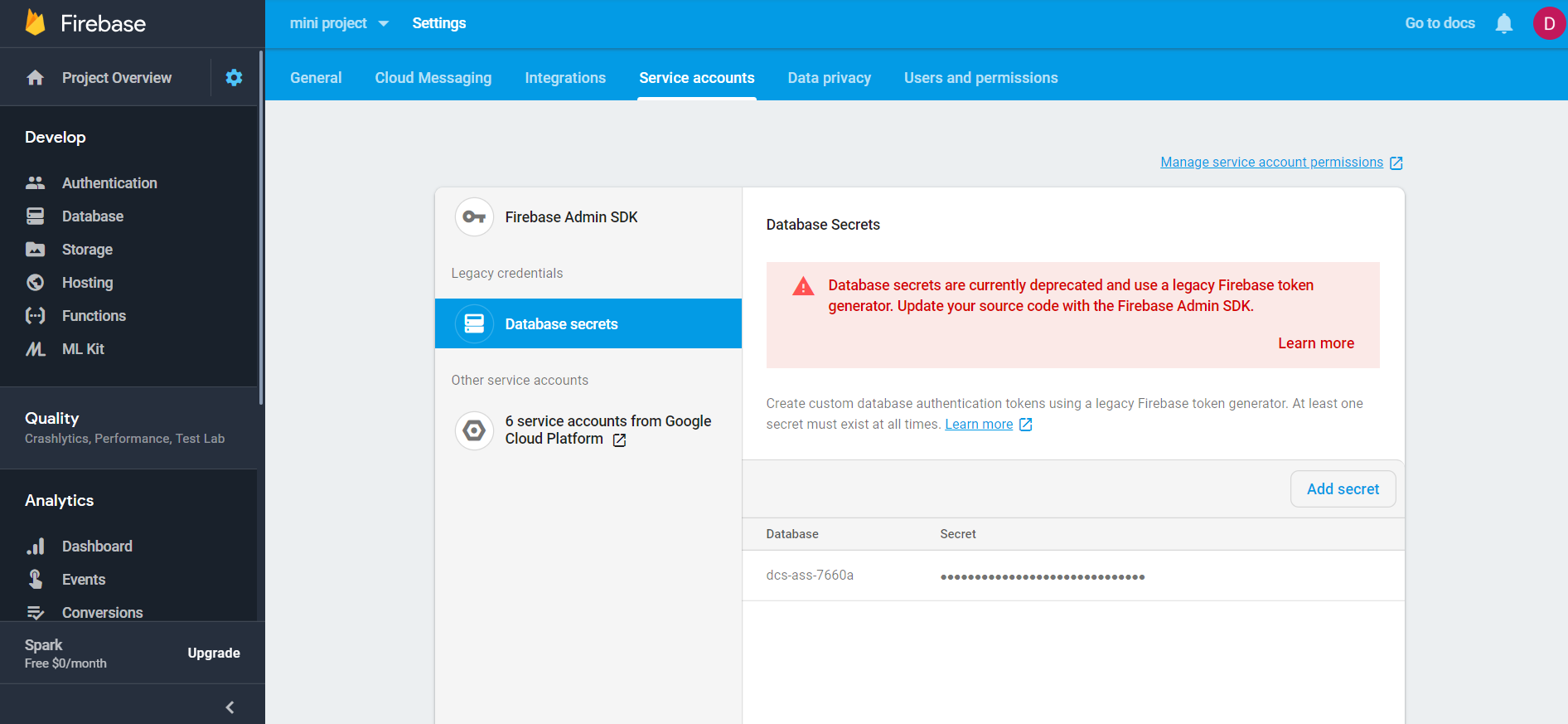
• Firebase Authentication provides backend services, easy-to-use SDKs, and ready- made UI libraries to authenticate users to your app.

• To sign a user into your app, you first get authentication credentials from the user. These credentials can be the user's email address and password. Then, you pass these credentials to the Firebase Authentication SDK. The backend services will then verify those credentials and return a response to the client.

• After a successful sign in, you can access the user's basic profile information, and you can control the user's access to data stored in other Firebase products. The Firebase Realtime Database is a cloud-hosted database. Data is stored as JSON and synchronized in real-time to every connected client. When you build cross-platform apps with iOS, Android, and JavaScript SDKs, all of your clients share one Realtime Database instance and automatically receive updates with the newest data.

• Instead of typical HTTP requests, the Firebase Realtime Database uses data synchronization—every time data changes, any connected device receives that update within milliseconds.

• Offline: Firebase apps remain responsive even when offline because the Firebase Realtime Database SDK persists your data to disk. Once connectivity is reestablished, the client device receives any changes it missed, synchronizing it with the current server state. The Firebase Real-time Database can be accessed directly from a mobile device or web browser; there’s no need for an application server.



**CHAPTER 4**

**CODE:**

#include <ESP8266WiFi.h>

#include <FirebaseArduino.h>

#define sensor1 D2

#define sensor2 D1

#define FIREBASE\_HOST "link to firebase real-time database"

#define FIREBASE\_AUTH "your secret key"

#define WIFI\_SSID "your wifi ssid"

#define WIFI\_PASSWORD "your password"

int count, avail, cap;

void availUpdate()

{

avail = cap - count;

Firebase.setInt("Availability",avail); //SET THE INTEGER VALUE LOCATED AT THE PATH

Serial.println(avail);

}

void setup() {

Serial.begin(9600); //9600 BITS PER SECOND

WiFi.begin(WIFI\_SSID, WIFI\_PASSWORD); //INITIALISES THE WIFI LIBRARY SETTINGS

while (WiFi.status() != WL\_CONNECTED)

{

delay(500);

}

ESP.wdtDisable(); //DISABLES THE SOFTWARE WATCHDOG

pinMode(sensor1, INPUT);

pinMode(sensor2, INPUT);

Serial.println("Start");

Firebase.begin(FIREBASE\_HOST, FIREBASE\_AUTH);

}

void loop()

{

ESP.wdtFeed(); //RESTART WATCHDOG

cap = Firebase.getInt("Capacity"); //GET THE INTEGER VALUE

if (digitalRead(sensor1) == HIGH) {

while (digitalRead(sensor2) == LOW) {

ESP.wdtFeed();

}

Serial.println("in");

if (count < cap)

{

count++;

}

availUpdate();

} else if (digitalRead(sensor2) == HIGH) {

while (digitalRead(sensor1) == LOW) {

ESP.wdtFeed();

}

Serial.println("out");

if (count >0)

{

count--;

}

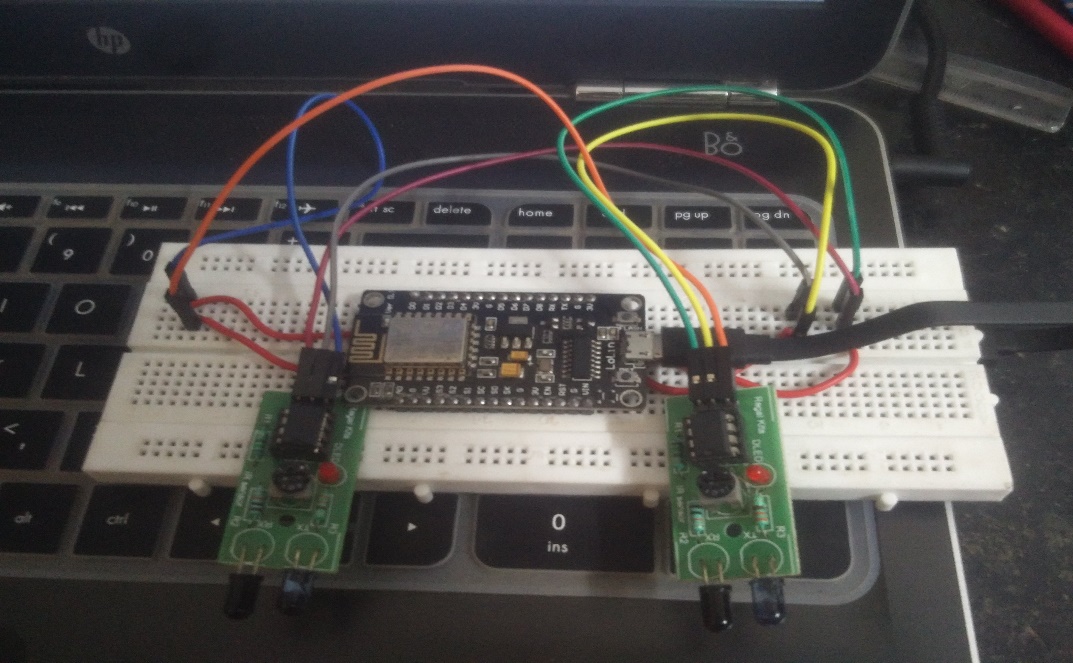
availUpdate();

}

}

**CHAPTER 5**

**WORKING**

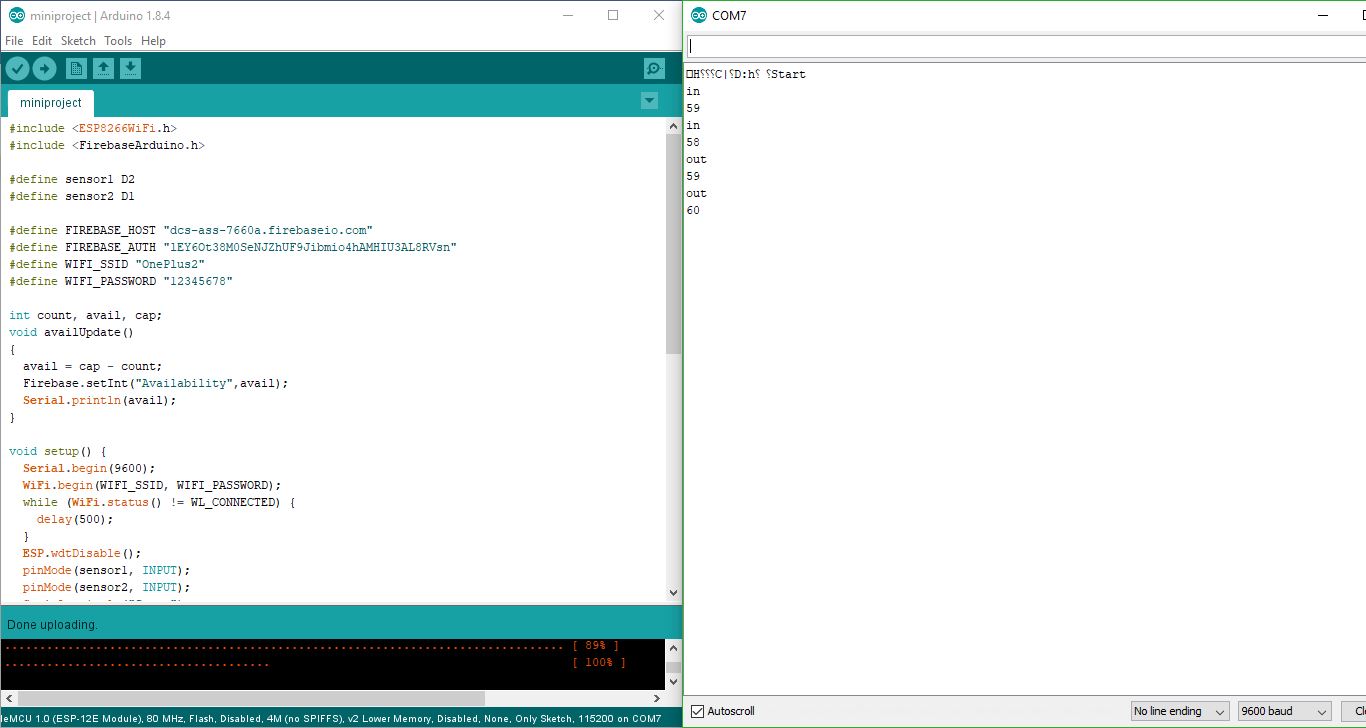
The problem statement was analysed and the algorithm was framed. The code was written in C language in the Arduino compiler. The code was uploaded to the NodeMCU. Two IR sensors were connected to the microcontroller at a distance so that they don’t overlap and can detect the passing of a person.

The capacity of the coach is initialized. One main variable is involved – “Availability”. According to the code, when a person enters the coach (i.e.) passes from sensor-1 to sensor-2, the variable gets decremented by 1, which indicates one less seat is available. Similarly, when a person exits the coach (i.e.) passes from sensor-2 to sensor-1, the variable gets incremented by 1, indicating one more seat is available. The formula involved is

***Availability = Capacity - Count***

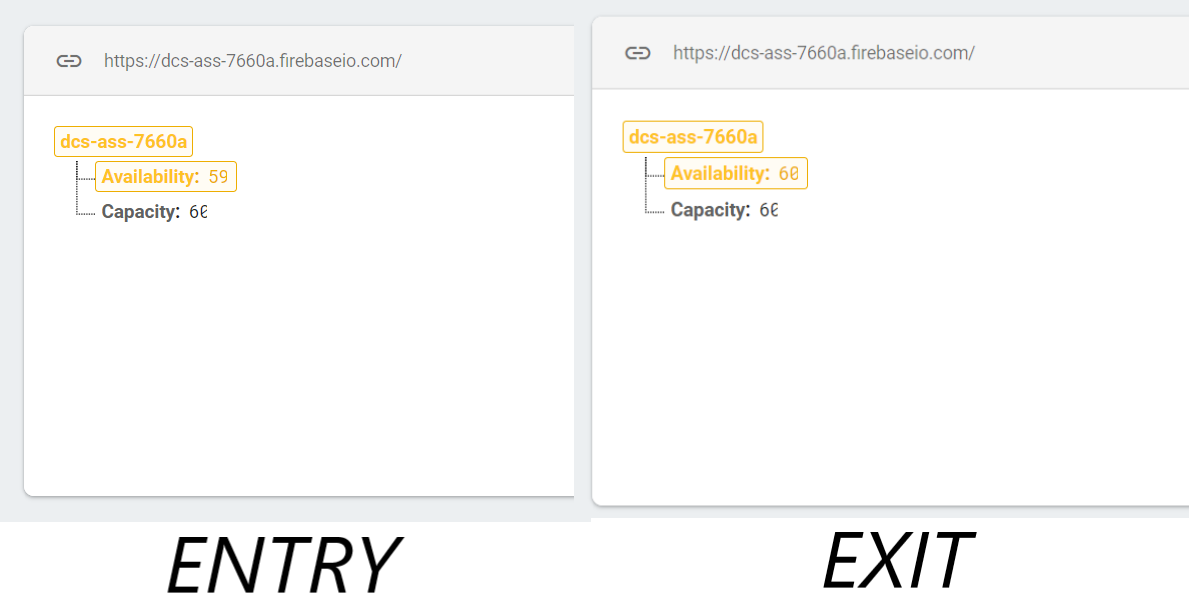
The value of “count” is 1.

The link and the secure key to the Firebase Database are written into the microcontroller. The NodeMCU is connected to a Wireless Network via its ESP8266 Wi-Fi module. This helps it to upload the data to the cloud. It is programmed in such a way that the data is updated every time a change in value occurs (i.e.) real-time updating. The Availability can be checked under the “Database” section of the Firebase console.



ENTRY-EXIT UPDATION

When a person enters or leaves the room or captivity the Infrared (IR) sensors detect the movement and decreases or increases the availability respectively to one. The new availability is instantaneously updated to the firebase from the NodeMCU and the updation can be seen in the following figure.



**CHAPTER 6**

**SCOPE OF THE PROJECT**

The project setup can be installed at the door (i.e.) entry/exit points of any room or in this case, a train coach. This helps us to have a check on the number of passengers entering the train. Any change in the count gets updated real-time in the database. This count can be cross-referenced with the data in the Railways database and if any discrepancy occurs, the particular coach can be inspected. We believe that this project will make public transportation more secure.

REFERENCES

[www.iisrp.org/research-paper-0716/ijsrp-p5552](http://www.iisrp.org/research-paper-0716/ijsrp-p5552)

“Bidirectional Visitor Counter”, IETE, New Delhi.

forum.arduino.cc