**SMART DOOR BELL**

***Mini Project Report submitted in partial fulfillment***

***of the requirement for the degree of***

**T. E. (Information Technology)**

Submitted By

Siddharth.P

Ritwik Mukherjee

Chaitanya Desai

Under the Guidance of

Prof. Ajitkumar Khachane

Department of Information Technology



Vidyalankar Institute of Technology

Wadala(E), Mumbai 400 037

## University of Mumbai

# 2019-20

### CERTIFICATE OF APPROVAL

**For**

**Mini Project Report**

This is to certify that

Siddharth P

Ritiwk Mukherjee

Chaitanya Desai

Have successfully carried out Mini Project entitled

“SMART DOOR BELL”

in partial fulfillment of degree course in

Information Technology

As laid down by University of Mumbai during the academic year

2019-20

Under the Guidance of

Prof. Ajitkumar Khachane

Signature of Guide Head of Department

Examiner 1 Examiner 2 Principal

Dr. S. A. Patekar

**ACKNOWLEDGEMENT**

We express our profound gratitude and sincere thanks to **Prof.** **AjitKumar Khachane** our respectable project guide, for his support and guidance. Without hiscounselling our project would not have seen the light of the day.

We extend our sincere thanks to **Dr. Girish Bhole** Head of the Department of Information Technology for offering valuable advice at every stage of this undertaking. We would like to thank all the staff members who willingly helped us. We are grateful to VIDYALANKAR INSTITUTE OF TECHNOLOGY for giving us this opportunity.

Ritwik Mukhejree

Chaitanya Desai

Siddharth P

**Table of contents**

|  |  |
| --- | --- |
| **TOPIC** | **PAGE NO.** |
| 1.ABSTRACT | 5 |
| 2.INTROUDCTION | 6 |
| 3.AIM & OBJECTIVES | 7 |
| 4.PROBLEM STATEMENT | 8 |
| 5.PROPOSED SYSTEM | 9 |
| 6.COMPONENTS  6.1 HARDWARE  6.2 SOFTWARE | 10.11.12 |
| 7.IMPLEMENTATION | 15 |
| 8.RESULTS & DISCUSSION | 15 |
| 9.CONCLSUION & FUTURE SCOPE | 16 |
| REFERENCES | 17 |
| APPENDIX  Code | 18 |

|  |  |
| --- | --- |
| **List of Figures** | **PageNo.** |
| 4.1 BlockDiagram | 5 |
| 4.2 NodeMcu | 10 |
| 4.3 Stepper Motor | 10 |
| 4.4 Double Realy | 11 |
| 4.5 BreadBoard | 11 |
| 4.6 Jumper Wires | 11 |
| 4.7 Arduino IDE | 12 |
| 4.8 Circuit Diagram | 13 |
| 4.9 Deployment & results  4.10 & 4.11 App Preview | 14 |

**1.Abstract**

Security has always been an important issue in the home or office.

A remote home security system offers many more benefits apart from keeping home owners, and their property, safe from intruders

It is a real time smart doorbell notification system for home security.The system is composed of the Doorbell interfaced with NodeMcu.The system combines the functions of a smart and a house network system. It enables the users to monitor visitors in real time via the IOT based doorbell installed in the entrance door to a house. The doorbell can be controlled in a smart way to intimate the user with a picture and a notification of the visitor at the doorstep. When a visitor rings the doorbell,the image of visitor is captured via any phone & the notification is sent to the user(someone’s at the door)and can act accordingly. This system is also intended to serve old age people and to identify unauthorized intruders. In the age of technology, it is necessary to update our security systems and to make living easier.In this project we have used stepper motor for door opening,relay for bell & nodemcu.

**2.Introduction**

This project is a significant step towards smart home and living. With the increase in trend of online shopping combined with conventional trends of delivery we realized that a major inconvenience is faced by the customer as well as the delivery person if the concerned one is not present at his home at time of delivery.

This also extends to friends and relatives who may visit your place unannounced.

There is also a security concern in the old systems as we cannot see the person outside clearly. The old aged people are also mostly are targeted for crime and looting. Thus the two fold problem identified is as following :

• There is no smart means through which the owner of the house is notified about the visitor in case he is outdoors or unable to hear the bell.

• There is no smart means through which the owner of the house can communicate and pass instructions to the visitor at the door.

Using the mobilephone, ‘NodeMcu, the smart doorbell, hence solves the problem of visitors remaining unattended in case the concerned person is not available. This smart doorbell alerts you when the bell is rung and lets you see and the visitors from your smartphone, anytime and anywhere. With the most important feature being able to live stream the feed of your front door to your device, be that you’re mobile or laptop or your according to how one has configured the NodeMcu.

One such advancement in the field of doorbell is using “The Automatic doorbell system” (ADBS). Door bells have moved from historical switches to modern touch pads and now it is more sophisticated with the usage of sensors and IOT.

**3. AIM & OBJECTIVE**

The increase in demand of IoT based technology that helps make everyday tasks easier and more secure inspired us to follow through with this idea. Ideally, we would want a user to be able to remotely authorize anyone who is at their door and needs access.

The **objective** of this **project** is to facilitate the user with a simple and customised technology to effectively manage visitors flowing to his/her premises. It enables the users to monitor visitors in real time via the **IOT** based **doorbell** installed near the entrance door to a house.

The core objectives are:

* • Gather system requirements
* • Evaluate and study the platform required for the system
* • Evaluate and study suitable development language, technologies and tools
* • Evaluate Methods of Interface
* • Program NodeMCU
* • Interface board for Stepper motor & relay
* • Program our own app
* • Evaluate and test the system
* • Maintain system

**4.PROBLEM STATEMENT**

The idea of a smart door is not new, however, most of these ideas revolve around using some sort of ‘key’ to unlock the door, such as communicating with the phone via Bluetooth, biometrics, or a keypad. Our project will be different. It will allow the person to gain access without using the aforementioned techniques, through the integration of a smart doorbell and app.

Suppose,someone is at the door ringing the door bell & it goes unnoticed because he/she is listening to music or doing some household and the visitor leaves with no output. To avoid this scene we are coming with a solution which says: So, when someone rings the doorbell, a photo of who is on the door is captured via our doorbell application, and you will receive that picture along a push notification to your phone(any with proper \*internet

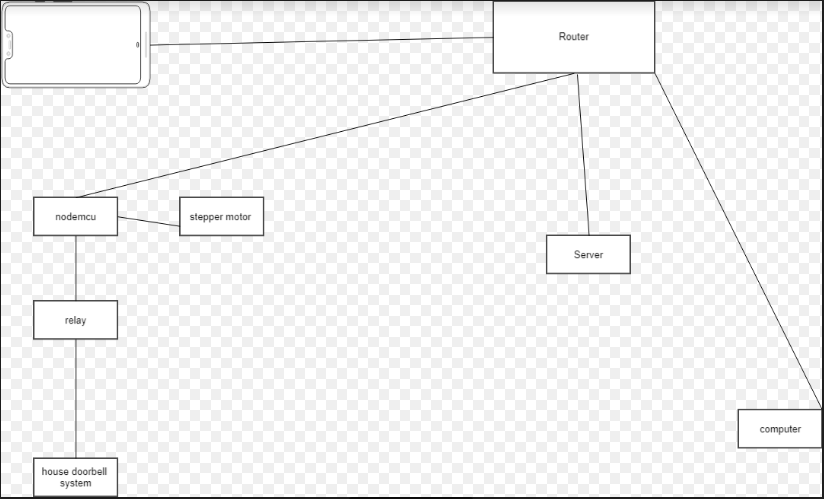
connection).

\*An internet connection is always required for this to work.

**5.PROPOSED SYSTEM**

In this proposed system, we are having our application which says,. Whenever some person DOOR bell switch(application), the user gets a notification that someones at the door and the camera gets triggered and capture the image of the visitor and sends that image to the home user, if that visitor is know to the user he/she can use our 2nd webapp & click the “OPEN DOOR BUTTON”.and can open the door automatically using stepper motor.

4.1:**Block Diagram**



**STEPPER MOTOR**

**NODEMCU**

**House DoorBell**

**RELAY**

**COMPUTER**

**SERVER**

**ROUTER**

**Fig 4.1**

**6.COMPONENTS**

6.1> HARDWARE REQUIREMENTS

* **NodeMcu:**

NodeMCU is an open source [Lua](https://www.lua.org/) based firmware for the [ESP32](http://espressif.com/en/products/hardware/esp32/overview) and [ESP8266 WiFi SOC from Espressif](http://espressif.com/en/products/esp8266/) and uses an on-module flash-based [SPIFFS](https://github.com/pellepl/spiffs) file system. NodeMCU is implemented in C and is layered on the [Espressif ESP-IDF](https://github.com/espressif/ESP-IDF).

The firmware was initially developed as is a companion project to the popular ESP8266-based [NodeMCU development modules](https://github.com/nodemcu/nodemcu-devkit-v1.0), but the project is now community-supported, and the firmware can now be run on any ESP module



Fig: 4.2

* **Stepper Motor:**

A **stepper motor**, also known as **step motor** or **stepping motor**, is a [brushless DC electric motor](https://en.wikipedia.org/wiki/Brushless_DC_electric_motor) that divides a full rotation into a number of equal steps. The motor's position can then be commanded to move and hold at one of these steps without any [position sensor](https://en.wikipedia.org/wiki/Rotary_encoder) for [feedback](https://en.wikipedia.org/wiki/Feedback) (an [open-loop controller](https://en.wikipedia.org/wiki/Open-loop_controller)), as long as the motor is carefully sized to the application in respect to [torque](https://en.wikipedia.org/wiki/Torque) and speed.



Fig:4.3

* **Double Relay:**

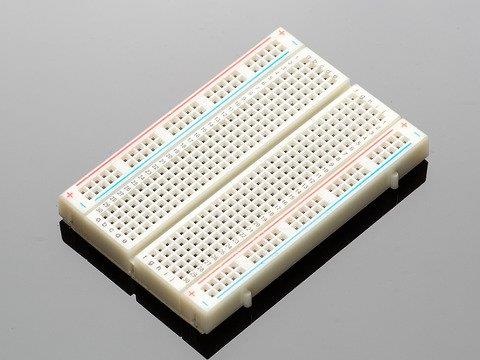
A **relay** is an [electrically](https://en.wikipedia.org/wiki/Electric) operated [switch](https://en.wikipedia.org/wiki/Switch). It consists of a set of input terminals for a single or multiple control signals, and a set of operating contact terminals. The switch may have any number of contacts in multiple [contact forms](https://en.wikipedia.org/wiki/Electrical_contact#Contact_form), such as make contacts, break contacts, or combinations thereof.



Fig:4.4

* **BreadBoard**

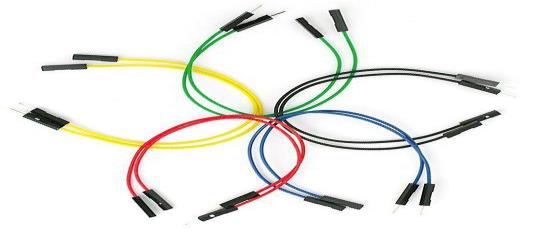
A breadboard is a solderless device for temporary prototype with electronics and test circuit designs. Most electronic components in electronic circuits can be interconnected by inserting their leads or terminals into the holes and then making connections through wires where appropriate. The breadboard has strips of metal underneath the board and connect the holes on the top of the board. Note that the top and bottom rows of holes are connected horizontally and split in the middle while the remaining holes are connected vertically.



**Fig:4.5**

* **Jumper Wires:**

Jumper wires are simply wires that have connector pins at each end, allowing them to be used to connect two points to each other without soldering.



**Fig:4.6**

**6.2 >SOFTWARE REQUIREMENTS**

* **Arduino IDE**

The Arduino integrated development environment (IDE) is a cross-platform application (for Windows, macOS, Linux) that is written in the programming language Java. It is used to write and upload programs to Arduino compatible boards, but also, with the help of 3rd party cores, other vendor development boards



**Fig :4.7**

* **Application**

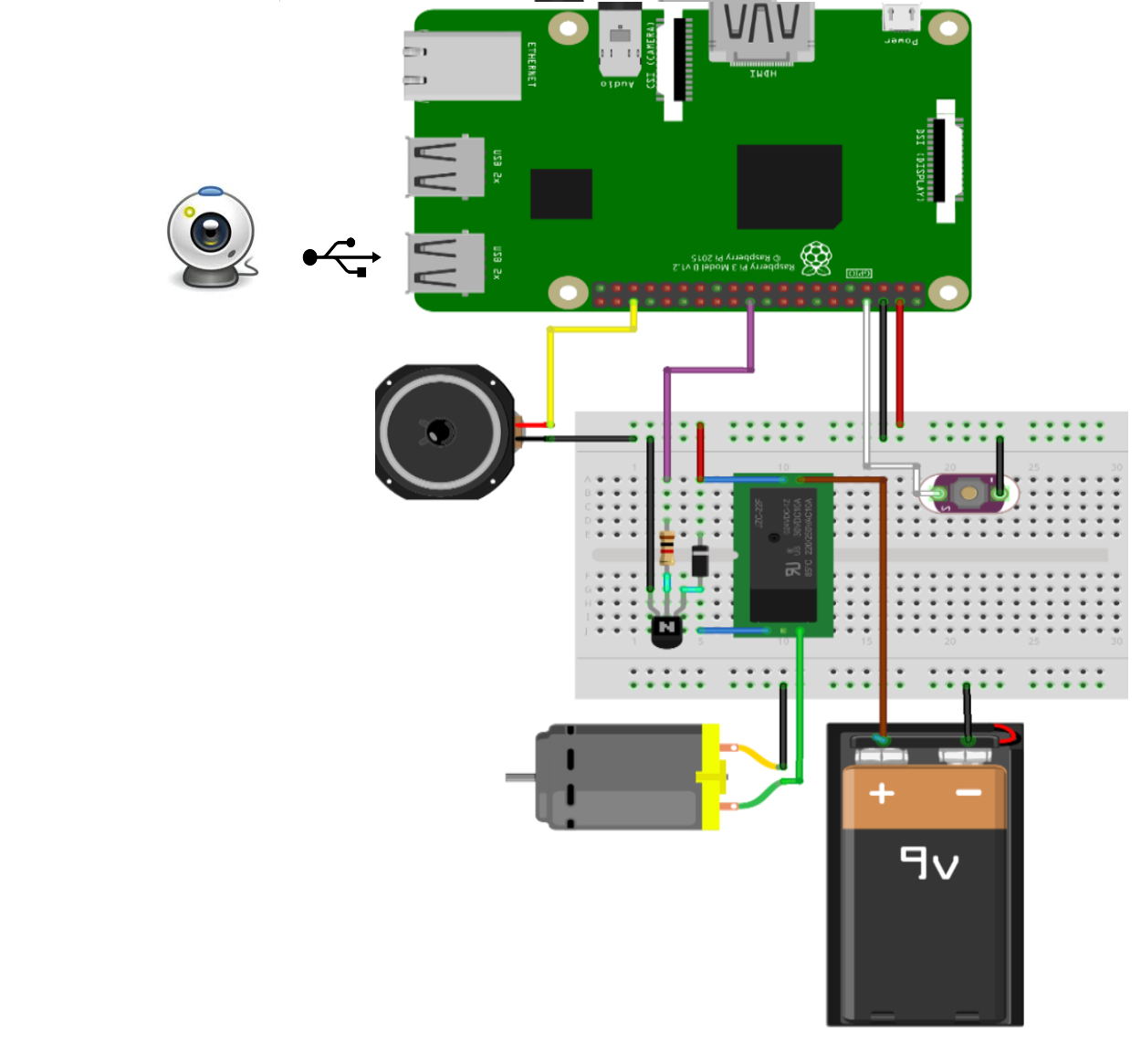
DoorBell app is a new platform that allows you to quickly build interfaces for controlling and monitoring your hardware projects from your Android device. After downloading the app, we can use it for ringing the bell ( tap the bell icon)

**7. IMPLEMENTATION**

The list of modules that are being implemented are given below:

* Configuration of NodeMcu and Web Sockets
* Connecting NodeMcu to Doorbell
* Configuring NodeMcu to Stepper Motor (for auto door open)
* Congfiguring NodeMcu to Double channel relay

7.2 Circuit Diagram



Computer/Laptop

DoubleRelay

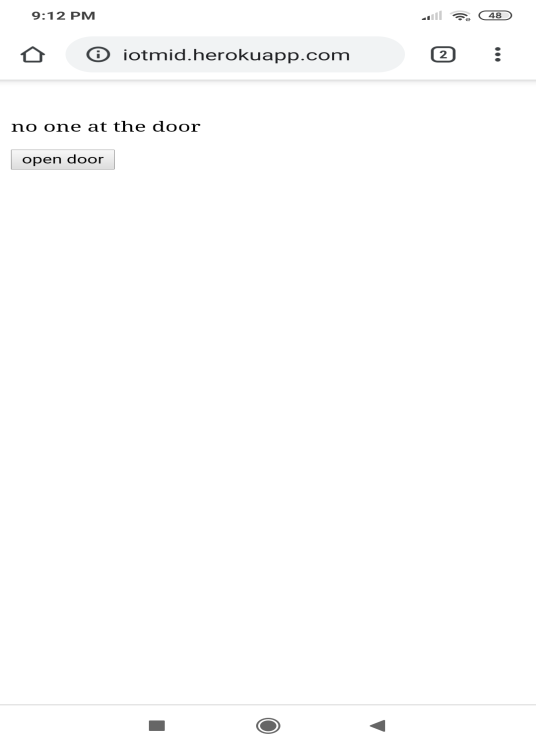
Stepper Motor

Nodecu

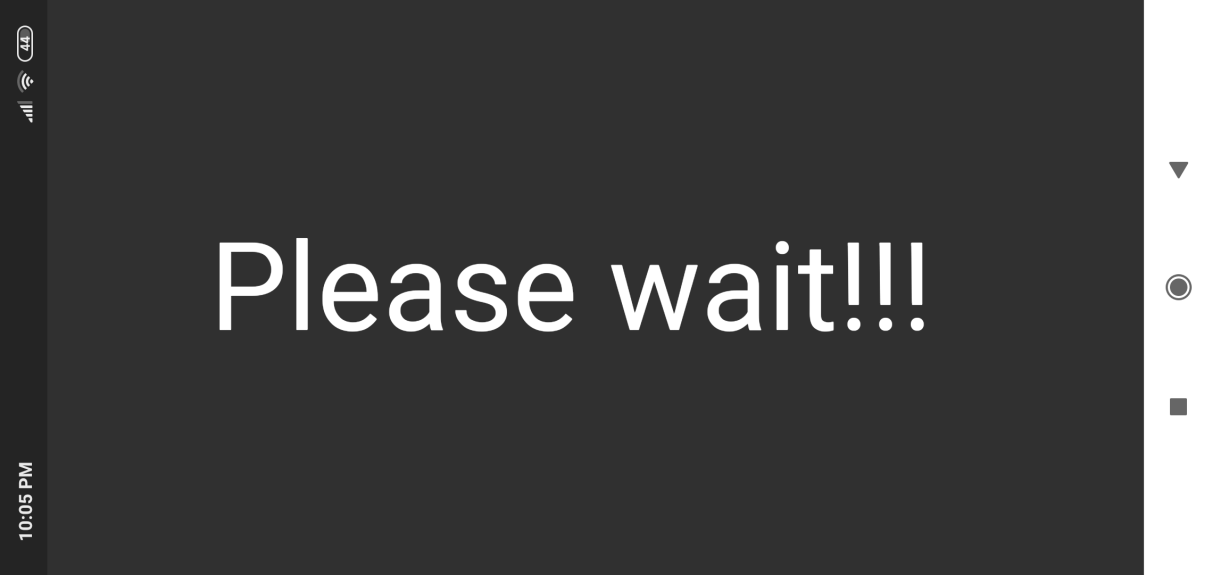
**Fig:4.8**

**DEPLOYMENT TESTING**

**APP & WEBAPP PREVIEW**

**** 

**Fig:4.9 Fig:4.10**

****

**Fg 4.11**

**8. RESULTS AND DISCUSSION**

**8.1 RESULT**

We have successfully created a Smart Door Bell which is able to accurately know the unkown person/visitor using our phone (camera) and display it on the application which on the mobile and send the notifications accordingly.

**8.2 DISCUSSION:**

During the whole period of the project we gained of lot of knowledge on the NodeMCU board and programming in Arduino IDE. If we talk about the achievements out of the project when starting to do the project it was to interface our stepper motor,double relay. We came to know a lot about the components that they only work at their given input voltage.

To make better IoT projects in the future and also it can send data to its server and able to collect the data and send the alerts accordingly . Out of which all the work was completed successfully.

**9.CONCLUSION & FUTURE SCOPE**

The project named "Smart IOT Doorbell Surveillance" has been designed with the domain as Internet of Things.

The basic concepts and working of IOT has been displayed in the running of the project. The project uses mainly an NodeMcu board and OOPS programming concept.

Since, today, in a technologically enhancing environment, security issues is of utmost concern, this project shows how technology can be used to enhance the security features of people's homes.

A doorbell is constructed which has the feature to send a notification to the owner when somebody is at the door, with an attached picture of the person. It uses materials such as an Arduino Board, an Ethernet Shield (to send notifications across services), a doorbell, relay channel and a webapp.

This project enables users to stay connected to their homes and ensure safety, even when they're travelling.

REFERENCES

[[1] https://www.instructables.com/id/Smart-DoorBell-System-Using-Internet-of-/](https://www.instructables.com/id/Smart-Garbage-Monitoring-System-Using-Internet-of-/)

[2] <https://www.pantechsolutions.net/smart-doorbell-system-using-iot>

[3] <https://www.hackster.io/taiyuk/iot-doorbell-faee18>

[4] [https://nodemcu.readthedocs.io/en/master/](%20https://nodemcu.readthedocs.io/en/master/)

[/](https://nodemcu.readthedocs.io/en/master/)

**APPENDIX: CODE**

**Nodemcu:**

#include <SocketIoClient.h>

#include <ESP8266WiFi.h>

#include <ESP8266HTTPClient.h>

#include <ArduinoJson.h>

#define LED 2

const char\* ssid = "G";

const char\* password = "";

SocketIoClient webSocket;

uint8\_t wire1 = 0;

uint8\_t wire2 = 4;

uint8\_t wire3 = 5;

uint8\_t wire4 = 16;

const uint16\_t \_delay = 5; /\* delay in between two steps. minimum delay more the rotational speed \*/

void sequence(bool a, bool b, bool c, bool d){  /\* four step sequence to stepper motor \*/

  digitalWrite(wire1, a);

  digitalWrite(wire2, b);

  digitalWrite(wire3, c);

  digitalWrite(wire4, d);

  delay(\_delay);

}

void event(const char \* payload, size\_t length) {

  digitalWrite(16, LOW);

  delay(1000);

  digitalWrite(16, HIGH);

  Serial.println("ring it");

}

 void openD(const char \* payload, size\_t length) {

  for(int i = 0; i<12; i++)

  {

    sequence(HIGH, HIGH, LOW, LOW);

    sequence(LOW, HIGH, HIGH, LOW);

    sequence(LOW, LOW, HIGH, HIGH);

    sequence(HIGH, LOW, LOW, HIGH);

  }

  delay(15000);

  /\* Rotation in opposite direction \*/

  for(int j = 0; j<12; j++)

  {

    sequence(LOW, LOW, HIGH, HIGH);

    sequence(LOW, HIGH, HIGH, LOW);

    sequence(HIGH, HIGH, LOW, LOW);

    sequence(HIGH, LOW, LOW, HIGH);

  }

  Serial.println("Opening Door");

}

 void setup() {

  //pinMode(5, OUTPUT);

  //pinMode(16, OUTPUT);

  //digitalWrite(16, LOW);

  pinMode(wire1, OUTPUT); /\* set four wires as output \*/

  pinMode(wire2, OUTPUT);

  pinMode(wire3, OUTPUT);

  pinMode(wire4, OUTPUT);

  Serial.begin(115200);

  WiFi.begin(ssid, password);

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

    delay(1000);

    Serial.println("Connecting...");

  }

  webSocket.on("ring", event);

  webSocket.on("openIt", openD);

  webSocket.begin("[iotmid.herokuapp.com](http://iotmid.herokuapp.com/)");

  Serial.println("Connected.");

}

void loop() {

  webSocket.loop();

}

**Android:**

import 'dart:async';

import 'dart:io';

import 'dart:convert';

import 'package:camera/camera.dart';

import 'package:flutter/material.dart';

import 'package:path/path.dart' show join;

import 'package:path/path.dart';

import 'package:path\_provider/path\_provider.dart';

import 'package:flutter/services.dart';

import 'package:adhara\_socket\_io/adhara\_socket\_io.dart';

Future<void> main() async {

  // Obtain a list of the available cameras on the device.

  final cameras = await availableCameras();

  // Get a specific camera from the list of available cameras.

  final firstCamera = cameras[1];

  runApp(

    MaterialApp(

      theme: ThemeData.dark(),

      home: TakePictureScreen(

        // Pass the appropriate camera to the TakePictureScreen widget.

        camera: firstCamera,

      ),

    ),

  );

}

// A screen that allows users to take a picture using a given camera.

class TakePictureScreen extends StatefulWidget {

  final CameraDescription camera;

  const TakePictureScreen({

    Key key,

    @required this.camera,

  }) : super(key: key);

  @override

  TakePictureScreenState createState() => TakePictureScreenState();

}

class TakePictureScreenState extends State<TakePictureScreen> {

  CameraController \_controller;

  Future<void> \_initializeControllerFuture;

  @override

  void initState() {

    super.initState();

    // To display the current output from the Camera,

    // create a CameraController.

    \_controller = CameraController(

      // Get a specific camera from the list of available cameras.

      widget.camera,

      // Define the resolution to use.

      ResolutionPreset.medium,

    );

    // Next, initialize the controller. This returns a Future.

    \_initializeControllerFuture = \_controller.initialize();

  }

  @override

  void dispose() {

    // Dispose of the controller when the widget is disposed.

    \_controller.dispose();

    super.dispose();

  }

  @override

  Widget build(BuildContext context) {

    SystemChrome.setPreferredOrientations([

      DeviceOrientation.portraitUp,

      DeviceOrientation.portraitDown,

    ]);

    return Scaffold(

      //appBar: AppBar(title: Text('Take a picture')),

      // Wait until the controller is initialized before displaying the

      // camera preview. Use a FutureBuilder to display a loading spinner

      // until the controller has finished initializing.

      body: Center(

      child: Column(

       mainAxisSize: MainAxisSize.min,

       children: <Widget>[

       SizedBox(

     height: 100.0,

     width: 100.0,

     child: RotatedBox(

      quarterTurns: 1,

     child: IconButton(

icon: Icon(Icons.add\_alert),

iconSize: 100.0,

tooltip: 'Ring the doorbell',

onPressed: () async {

// Take the Picture in a try / catch block. If anything goes wrong,

// catch the error.

try {

// Ensure that the camera is initialized.

await \_initializeControllerFuture;

// Construct the path where the image should be saved using the

// pattern package.

final path = join(

// Store the picture in the temp directory

// Find the temp directory using the `path\_provider` plugin.

(await getTemporaryDirectory()).path

'${DateTime.now()}.png',

);

// Attempt to take a picture and log where it's been saved.

await \_controller.takePicture(path);

init(path, context);

// If the picture was taken, display it on a new screen.

Navigator.push(

context,

MaterialPageRoute(

builder: (context) =>

DisplayPictureScreen(imagePath: path),

),

);

} catch (e) {

// If an error occurs, log the error to the console.

print(e); }                    },

                  ),

                )),

            // FutureBuilder<void>(

          ],

        ),

      ),

    );

  }

  void init(path, context) async {

  SocketIOManager manager = SocketIOManager();

  SocketIO socket = await manager

  createInstance(SocketOptions('<http://iotmid.herokuapp.com/')>);

  socket.onConnect((data) {

  print("connected...");

   File imageFile = new File(path);

   List<int> imageBytes = imageFile.readAsBytesSync();

   String base64Image = base64.encode(imageBytes);

   socket.emit("doorBell", [base64Image]);

   Navigator.pop(context);

    });

    socket.connect();

  }

}

// A widget that displays the picture taken by the user.

class DisplayPictureScreen extends StatelessWidget {

  final String imagePath;

  const DisplayPictureScreen({Key key, this.imagePath}) : super(key: key);

  @override

  Widget build(BuildContext context) {

    return Scaffold(

        //appBar: AppBar(title: Text('Display the Picture')),

        //Image.file(File(imagePath)),

        body: Center(

      child: RotatedBox(

        quarterTurns: 1,

        child: Text(

          "Please wait!!!",

          textScaleFactor: 6,

        ),

      ),

    ));

  }

}

**Sockets:**

const express = require('express');

const bodyParser = require('body-parser');

const cors = require('cors');

const app = express();

app.use(cors());

app.use(bodyParser.urlencoded({extended: true}));

app.use(bodyParser.json());

app.set('view engine', 'ejs');

app.use(express.static('public'));

app.get('/', (req, res) => {

    res.render('index');

});

const server = app.listen(process.env.PORT || 80);

require('./api/sockets/default.sockets')(server, app)

module.exports = (server, app) => {

    const io = require('[socket.io](http://socket.io/)')(server);

    let arr = [];

  io.on('connection', (socket) => {

  console.log('someone connected');

 socket.on('doorBell', (event) => {

   console.log(event);

            io.emit('ring', "true");

            io.emit('imageData', event);

            arr.push(event);

        });

        socket.on('orderNo', (event) => {

            io.emit('eOrderId', (event - 0));

        });

        socket.on('openDoor', (mes) => {

            io.emit('openIt', true);

        });

        socket.on('disconnect', () => {

            console.log('someone disconnected');

        });

    });

    app.get('/app', (req, res) => {

        return res.json({

            img: arr[arr.length-1]

        });

    });

}

**Notification web page:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <meta http-equiv="X-UA-Compatible" content="ie=edge">

    <script src="<https://cdnjs.cloudflare.com/ajax/libs/push.js/1.0.5/push.js>"></script>

    <title>test</title>

</head>

<body onload="onload()">

    <img id="img" src="" alt="">

    <p id="para"></p>

    <button onclick="openit()">open door</button>

     <script src="/[socket.io/socket.io.js](http://socket.io/socket.io.js)"></script>

    <script>

  var full = location.protocol + '//' + location.hostname + (location.port ? ':' + location.port : '');

let socket = io.connect(full);

let baseString

socket.on('imageData', (mes) => {

baseString = mes;

 baseString = "data:image/png;base64," + baseString;

 document.querySelector('#img').src = baseString;

   Push.create("Someone's at the door", {

   body: "open or nahh",

    timeout: 5000,

   onClick: function () {

  console.log(this);

                }

            });

        });

  async function onload() {

  Push.Permission.request(onGranted, onDenied)

   function onGranted() {

  console.log('granted');

     }

   function onDenied() {

  console.log('denied');

   }

   let res = await (await fetch('/app')).json(

   if (typeof res.img !== 'undefined') {

   baseString = res.img;

   baseString = "data:image/png;base64," + baseString;

   document.querySelector('#img').src = baseString;

            } else {

                document.querySelector('#para').innerHTML = "no one at the door";

            }

        }

        function openit() {

            socket.emit("openDoor", "true");

        }

    </script>

</body>

</html>