RECEDE: A SOCIAL DISTANCING SYSTEM USING ESP32 MICROCONTROLLER AND BLUETOOTH LOW ENERGY FOR PEER-TO-PEER WARNING SIGNAL

An Undergraduate Thesis
Presented to the Faculty of the
College of Information and Communications Technology
West Visayas State University
La Paz, Iloilo City

In Partial Fulfillment
of the Requirements for the Degree
Bachelor of Science In Information Technology

by
Catherine G. Duero
Jason P. Esperela
Jellie Marie J. Jover
John Ray T. Godin
Albert S. Parreño

June 2023

DISCLAIMER

This software project and its corresponding documentation titled "*Recede: A Social Distancing System using ESP32 Microcontroller and Bluetooth for Peer to Peer Warning System*" is submitted to the College of Information and Communications Technology, West Visayas State University, in partial fulfillment of the requirements for the degree, Bachelor of Science in Information Technology. It is the product of our own work, except where indicated text.

We hereby grant the College of Information and Communications Technology permission to freely use, publish in local or international journal/conferences, reproduce, or distribute publicly the paper and electronic copies of this software project and its corresponding documentation in whole or in part, provided that we are acknowledged.

Catherine G. Duero
Jason P. Esperela
Jellie Marie J. Jover
John Ray T. Godin
Albert S. Parreño

TABLE OF CONTENTS

INTRODUCTION

Recede was created to help mitigate the spread of Covid-19 virus using an ESP32 microcontroller and its Bluetooth Low Energy capability. The researchers created a system that could aid in the strict implementation of social distancing and contact tracing of individuals who may have contracted the disease to isolate immediately the infected and their close contacts.

RECEDE SYSTEM REQUIREMENT

Software

Microcontroller sketch

Arduino IDE version:

Any version will do

Additional Board Manager:

https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json

HTML(Frontend)

Sublime Text 3 version:

Any version will do

HTML version:

version 5

PHP(Backend)

PHP version:

8.1.4

Sublime Text 3 version:

Any version will do

Hardware

ESP32 Microcontroller

Passive Buzzer

Toggle Switch

Lithium-ion Battery (1200mah)

Micro USB Charging Module

Jumper cable

Plastic Hard Case

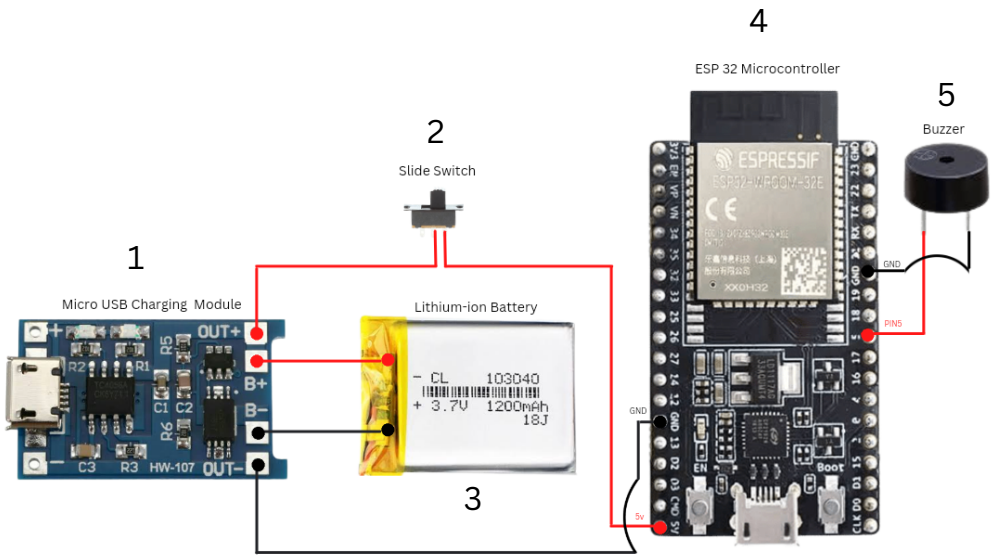
RECEDE INITIATING THE SYSTEM

1. Slide the toggle switch to turn on the Recede.

2. If you see the red light indicator glowing then the device is turned on.

3. Next, wear it on your neck or carry the recede in your pocket or like a keychain. Then, you're ready to go. If you hear the buzzer emit a pulsing sound, observe proper social distancing. If not, you will hear a continuous sound which indicates that you have been logged into the database that is subject for contact tracing if any positive individual is detected in your area.

RECEDE SCHEMATIC DIAGRAM



1. Micro USB Charging Module - it is connected to the lithium-ion battery to provide power if the battery runs out.

2. Slide Switch - used to turn on and off the Recede.

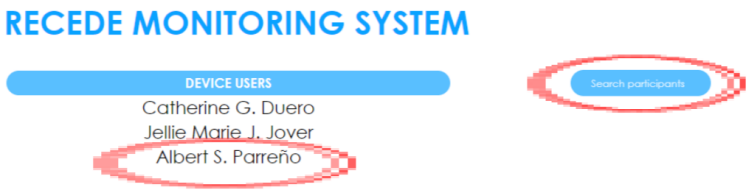
3. Lithium-ion Battery - it is the power source of the device. It has 3.7 volts with 1200mAh capacity.

4. ESP32 Microcontroller - it is the main component of the device. It is used to program the device like setting the allowed distance, configure the information per device, and assign to which Wi-Fi it should be connected.

5. Buzzer - it emits sound whenever the social distancing is violated.

RECEDE DASHBOARD GUIDE

The dashboard contains sensitive data and is hosted locally via XAMPP. To access it please follow these steps:
1. Make sure to install XAMPP follow the link https://www.apachefriends.org/
2. Locate the folder htdocs inside the XAMPP folder in file explorer and paste the files. Download the files in github under the folder name recede/src/ database using this link https://github.com/johnraygodin/recede/tree/main/src/database.
3. Open XAMPP and run apache web server and MySQL.
4. Open browser and type in the address http://localhost/recede/dashboardver_landing_page.php.
5. There you have it you now have access to the Dashboard that contains participants and there responding close contacts.



In the upper right corner of the dashboard, you can click the search bar to search for participants that use Recede. This will come in handy if there are a lot of participants if this was implemented inside the campus of West Visayas State University. If you press a participant under device users column it will redirect you to a separate page.

This will be the landing page if you perform the previous action where it will provide you with lists of individuals who came
close contact with Albert. This provides the name, year and section, cell phone number, and the date and time of contact. Also this page will let you filter the date of contact in the upper right corner of the page.

RECEDE ARDUINO IDE GUIDE

To upload a sketch to the microcontroller these are the steps you need to take:
1. Install Arduino IDE via this link: https://www.arduino.cc/en/software
2. You can follow this link that would redirect you to a video for a detailed set up of the environment https://youtu.be/CD8VJl27n94.
3. After setting up necessary prerequisites connect the ESP32 Microcontroller using a Micro usb type-b cable to your laptop's usb port.
4. After completing the steps above you can now download the sketch for the RECEDE system under recede/src/arduino folder in github with a filename arduino_code.ino using this link https://github.com/johnraygodin/recede.
5. Locate the file and then click upload to flash the firmware to the ESP32 Microcontroller.

The researchers highly suggests that the user/programmer should study the Bluetooth Low Energy codes for ESP32 inside the examples located inside the Arduino IDE. File > Examples >
ESP32 BLE Arduino. The codes regarding the system closely relates to the programming language of Java and C++.

Here are some codes explained that are interchangeable according to your system or preference that will serve as some examples/guide in modifying the code.
(The “n” below stands as a placeholder for a number. One number corresponds to one device)

```
const char* ssid = "Recede";
const char* password = "09870987";
const char* serverName = "http://192.168.30.57/albert.php"; //"http://192.168.30.57/jellie.php"; //"http://192.168.30.57/catherine.php"; //
```

1. const char* “ssid”
   - Data type: String
   - This is where we declare the name of the Wi-Fi that we are using in our system.
2. const char* “password”
   - Data type: String
   - This is where we declare the password of the Wi-Fi that we are using in the system.
3. “serverName”
   - Data type: String
   - This is where we declare the IP address of the Wi-Fi we are connected to and the name of the .php file of the respondent in the database table.

```
String deviceName;
const int pin = 5;
int secondsCount1, secondsCount2, secondsCount3 = 0;
int awayCount1, awayCount2, awayCount3 = 0;
```

4. int “secondsCount(n)” and “awayCount(n)”
   - Data type: integer
   - These are the placeholder for the number of seconds (counting of the seconds) for the device if it is below one meter or over one meter.

```
String value = "";
String value1 = "Catherine G. Duero | BSIT-4A | 09508716087";
String value2 = "Jellie Marie J. Jover | BSIT-4A | 09927203982";
String value3 = "Albert S. Parreño | BSIT-4A | 09567000352";
```

5. String “value(n)”
   - Data type: String
   - This is where we declare the details of the user that holds a certain Recede module. One string of details equates to one user.

```
#define SERVICE_UUID1 "4fafc201-1fb5-459e-8fcc-c5c9c331914a" //declaration of service uuids to advertise (server side, suppose to be)
#define SERVICE_UUID2 "4fafc201-1fb5-459e-8fcc-c5c9c331914b"
#define SERVICE_UUID3 "4fafc201-1fb5-459e-8fcc-c5c9c331914c"
//////////////////////////////////////////////////////////////////////

BLEUUID serviceUUID1("4fafc201-1fb5-459e-8fcc-c5c9c331914a"); //declaration of known service uuids to scan (client side, suppose to be)
BLEUUID serviceUUID2("4fafc201-1fb5-459e-8fcc-c5c9c331914b");
BLEUUID serviceUUID3("4fafc201-1fb5-459e-8fcc-c5c9c331914c");
```

6. #define SERVICE_UUID(n)
   - Data type: String
   - This is the declaration of the UUID that is being used or advertised by the device, this is a server-side code declaration.
   - We only use one UUID (unique) for each device.

7. BLEUUID serviceUUID(n)
   - Data type: String
   - This is a declaration of the Bluetooth Low Energy Universal Unique Identifier (BLEUUID) of another device involved in the Recede system. One declared BLEUUID corresponds to one Recede device. This is a client side declaration.

```
server = BLEDevice::createServer();
BLEAdvertising* advertising = BLEDevice::getAdvertising();
advertising->addServiceUUID(SERVICE_UUID3);
BLEDevice::startAdvertising();
```

8. BLEDevice::init(“bluetoothDeviceName”)
   - Declared within the parameter is the device’s name that will be shown when scanned.

9. advertising->addServiceUUID(SERVICE_UUID(n));
   - Declared within the parameter is the current device’s service UUID that will be advertised to other devices when they too scans.

```
#include <BLEDevice.h>
#include <WiFi.h>
#include<HTTPClient.h>

const char* ssid = "Recede";
const char* password = "09870987";
const char* serverName = "http://192.168.30.57/albert.php"; //"http://192.168.30.57/jellie.php"; //"http://192.168.30.57/catherine.php"; //

String deviceName;
const int pin = 5;
int secondsCount1, secondsCount2, secondsCount3 = 0;
int awayCount1, awayCount2, awayCount3 = 0;

String value = "";
String value1 = "Catherine G. Duero | BSIT-4A | 09508716087";
String value2 = "Jellie Marie J. Jover | BSIT-4A | 09927203982";
String value3 = "Albert S. Parreño | BSIT-4A | 09567000352";


int cutOff = -52;
int rssi;
String dot = "";
//////////////////////////////////////////////////////////////////////
BLEServer* server;

#define SERVICE_UUID1 "4fafc201-1fb5-459e-8fcc-c5c9c331914a" //declaration of service uuids to advertise (server side, suppose to be)
#define SERVICE_UUID2 "4fafc201-1fb5-459e-8fcc-c5c9c331914b"
#define SERVICE_UUID3 "4fafc201-1fb5-459e-8fcc-c5c9c331914c"
//////////////////////////////////////////////////////////////////////

BLEUUID serviceUUID1("4fafc201-1fb5-459e-8fcc-c5c9c331914a"); //declaration of known service uuids to scan (client side, suppose to be)
BLEUUID serviceUUID2("4fafc201-1fb5-459e-8fcc-c5c9c331914b");
BLEUUID serviceUUID3("4fafc201-1fb5-459e-8fcc-c5c9c331914c");

BLEScan* scan;
BLEClient* pClient;

BLEAdvertisedDevice device;
```

```
void setup() {
  Serial.begin(115200);
  pinMode(pin,OUTPUT);
  Serial.println("Scanning...");
  BLEDevice::init("Recede3");       //Per ESP, we change the .php file, BLE advertising name,and the advertised UUID

  server = BLEDevice::createServer();
  BLEAdvertising* advertising = BLEDevice::getAdvertising();
  advertising->addServiceUUID(SERVICE_UUID3);
  BLEDevice::startAdvertising();
}

void loop() {

  scan = BLEDevice::getScan();
  scan->setActiveScan(true);
  BLEScanResults results = scan->start(1);

  for(int i = 0; i < results.getCount(); i = i + 1){
    device = results.getDevice(i);
    rssi = device.getRSSI();

    /*Serial.println("RSSI Value is: ");
    Serial.println(rssi);*/

    deviceName = (device.getName().c_str());

////////////////////////////////////////////////////////////////////////

     if(device.isAdvertisingService(serviceUUID1) && rssi > cutOff){
       digitalWrite(pin,HIGH);
       delay(500); //Delay between loops
       secondsCount1 = secondsCount1 + 1;
       Serial.print("Recede 1's Count: ");
       Serial.println(secondsCount1);

       if(secondsCount1 == 10){
         Serial.print("The value of value1 is: ");
         Serial.println(value1);

    //if (value1 != ""){

       Serial.print("We have uploaded ");
       Serial.print(value1);
       Serial.print(" to the database.");

       Serial.print("Uploading to the Database" );
       WiFi.begin(ssid, password);
       Serial.print("Connecting to " );
       Serial.println(ssid);

       while(WiFi.status() != WL_CONNECTED){
         delay(1000);
         Serial.print(dot);
         dot = dot + ".";

         if(dot == "....."){
           Serial.print("/////////RESTARTING WIFI/////////");
           Serial.print("Uploading to the Database " );
           WiFi.begin(ssid, password);
           Serial.print("Connecting to " );
           Serial.println(ssid);
           dot = "";
         }
       }//while
           Serial.println("");
           Serial.print("Connected to WiFi network with IP Address: ");
           Serial.println(WiFi.localIP());

           if(WiFi.status()== WL_CONNECTED){
             HTTPClient http;

             http.begin(serverName);

             http.addHeader("Content-Type", "application/x-www-form-urlencoded");

             value = value1;
             String httpRequestData =  "&value= " + value;
             Serial.print("httpRequestData: ");
             Serial.println(httpRequestData);
```

```
            int httpResponseCode = http.POST(httpRequestData);

            if (httpResponseCode > 0){
              Serial.print("HTTP Response code: ");
              Serial.println(httpResponseCode);
            }
            else{
              Serial.print("Error code: ");
              Serial.println(httpResponseCode);
            }
            http.end();

          }//wifi status
          else{
            Serial.println("WiFi Disconnected");
          }

          WiFi.disconnect();

          dot = "";
          value = "";
          awayCount1 = 0;
          secondsCount1 = 0;

      //}//if value

    }//secondscount
    if(secondsCount1 > 10){
      secondsCount1 = 10;//if the countdown exceeds 10, return count to 9
    }
  }// if device name && rssi > cutOff

  if(device.isAdvertisingService(serviceUUID1) && rssi < cutOff){
    digitalWrite(pin,LOW);
    delay(500); //Delay between loops
    awayCount1 = awayCount1 + 1;
    Serial.print("Recede 1's Away Count: ");
    Serial.println(awayCount1);
      if(awayCount1 == 3){//3 seconds of being away
```

```
        secondsCount1 = 0;
        awayCount1 = 0;

      }
  }// if device name && rssi < cutOff
  else{
    digitalWrite(pin,LOW);
  }// else
```

RECEDE TROUBLESHOOTING GUIDE

Device does not turn on.
Check if the slide switch is turned on. If the device still does not glow its red indicator,
charge the device for 20 minutes and turn it on again.

Device does not emit a warning noise.
Open the device and click the EN button. There could just be a bug in the starting of the device.

Device got stuck in a steady buzzing state
Open the case and click the EN button to restart the device.

RECEDE FAQS

How long does it take for my device to be fully charged?
Devices are completely operable at 20 minutes of charging using a 5 watts charging adapter.

Am I allowed to modify the code of the device to update or modify my details?
No. The system admin is the sole manager and modifier of the codes within the device.

Am I allowed to open the case of the device?
Yes. You are allowed to open the case of the device if you were to reset/restart the device's code.

Will other people who are not involved in the system who hold an ESP32 be able to read/get my details?
No. Your devices have unique identifiers embedded on each of them so no external devices are able to intrude with the system's operation.