



VIT[®]

Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

ELECTRONIC LETTER BOX

SUBMITTED BY-

PRANAV MANDALA (17BCE0803)

RASHI KASERA (17BCE2421)

AS ADVAITH (17BCE0104)

Abstract

A notification system for incoming letter in the letter box is very important especially for tall building residents and offices. This project aims to solve the daily problems of many households, when they receive a letter, but forget to check that due to their busy schedule.

By using this system, the user can do more efficient mail checking by relying on the notification from the system. With this device installed in their homes, now no longer they need to check their letterboxes in hope of any letter. If there's one, you would get notified.

The objective of this project is to develop an intelligent mail notification system that capable to alert the user whenever there are incoming letters in the letter box through Short Message Services (SMS). The designed system consists of an infra- red (IR) sensor, a microcontroller and a transceiver. We use an Arduino Uno board as the microcontroller and a GSM module as the transceiver.

HARDWARE REQUIREMENTS

Necessary list Of Components:

1. Arduino Uno Price- 499

The Arduino UNO is an open-source microcontroller board based on the Microchip ATmega328P microcontroller and developed by Arduino.cc. The board is equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards (shields) and other circuits. The board has 14 Digital pins, 6 Analog pins, and programmable with the Arduino IDE (Integrated Development Environment) via a type B USB cable. It can be powered by a USB cable or by an external 9-volt battery, though it accepts voltages between 7 and 20 volts. It is also similar to the Arduino Nano and Leonardo. The hardware reference design is distributed under a Creative Commons Attribution Share-Alike license and is available on the Arduino website. Layout and production files for some versions of the hardware are also available. "Uno" means one in Italian and was chosen to mark the release of Arduino Software (IDE) 1.0.

The Uno board and version 1.0 of Arduino Software (IDE) were the reference versions of Arduino, now evolved to newer releases. The Uno board is the first in a series of USB

Arduino boards and the reference model for the Arduino platform. The ATmega328 on the Arduino Uno comes preprogrammed with a bootloader that allows uploading new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol. The Uno also differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it uses the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter.



General Pin functions of Arduino Uno:

LED: There is a built-in LED driven by digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

VIN: The input voltage to the Arduino/Genuino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.

5V: This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 20V), the USB connector (5V), or the VIN pin of the board (7-20V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage the board.

3V3: A 3.3-volt supply generated by the on-board regulator. Maximum current draw is 50 mA.

GND: Ground pins.

IOREF: This pin on the Arduino/Genuino board provides the voltage reference with which the microcontroller operates. A properly configured shield can read the IOREF pin voltage and select the appropriate power source or enable voltage translators on the outputs to work with the 5V or 3.3V.

Reset: Typically used to add a reset button to shields which block the one on the board.

2. Node MCU Price- 339

ESP8266 is a low-cost, WiFi Module chip that can be configured to connect to the Internet for Internet of Things (IoT) and similar Technology Projects. Basically, normal Electrical and Mechanical equipments cannot connect to the Internet on their own. They don't have the in-built setup to do so. We can setup ESP8266 with these equipments and do amazing stuff like controlling, monitoring, analysis and much more.



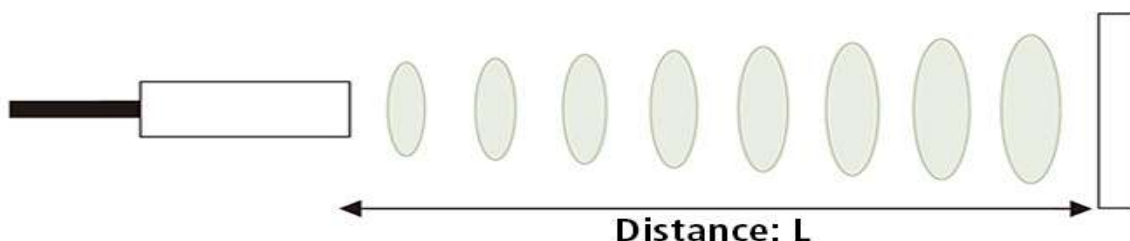
NodeMCU is a Firmware on ESP8266. It is a wifi SOC (system on a chip) produced by Espressif Systems. It is based ESP8266 -12E WiFi module. It is a highly integrated chip designed to provide full internet connectivity in a small package. It can be programmed directly through USB port using LUA programming or Arduino IDE. By simple programming we can establish a WiFi connection and define input/output pins according to your needs exactly like arduino, turning into a web server and a lot more. NodeMCU is the WiFi equivalent of ethernet module. It combines the features of WiFi access point and station + microcontroller. These features make the NodeMCU extremely powerful tool for WiFi networking. It can be used as access point and/or station, host a web server or connect to internet to fetch or upload data.

3. Ultrasonic Sensor Price- 200

As the name indicates, ultrasonic sensors measure distance by using ultrasonic waves. The sensor head emits an ultrasonic wave and receives the wave reflected back from the target. Ultrasonic Sensors measure the distance to the target by measuring the time between the emission and reception. An optical sensor has a transmitter and receiver, whereas an ultrasonic sensor uses a single ultrasonic element for both emission and reception. In a reflective model ultrasonic sensor, a single oscillator emits and receives ultrasonic waves alternately. This enables miniaturization of the sensor head.



The distance can be calculated with the following formula: Distance $L = \frac{1}{2} \times T \times C$, where L is the distance, T is the time between the emission and reception, and C is the sonic speed. (The value is multiplied by $\frac{1}{2}$ because T is the time for go-and-return distance.)



Ultrasonic Sensor Pin Configuration

Pin Number	Pin Name	Description
1	Vcc	The Vcc pin powers the sensor, typically with +5V
2	Trigger	Trigger pin is an Input pin. This pin has to be kept high for 10us to initialize measurement by sending US wave.
3	Echo	Echo pin is an Output pin. This pin goes high for a period of time which will be equal to the time taken for the US wave to return back to the sensor.
4	Ground	This pin is connected to the Ground of the system.

4. Necessary wires to connect the components Price- 89

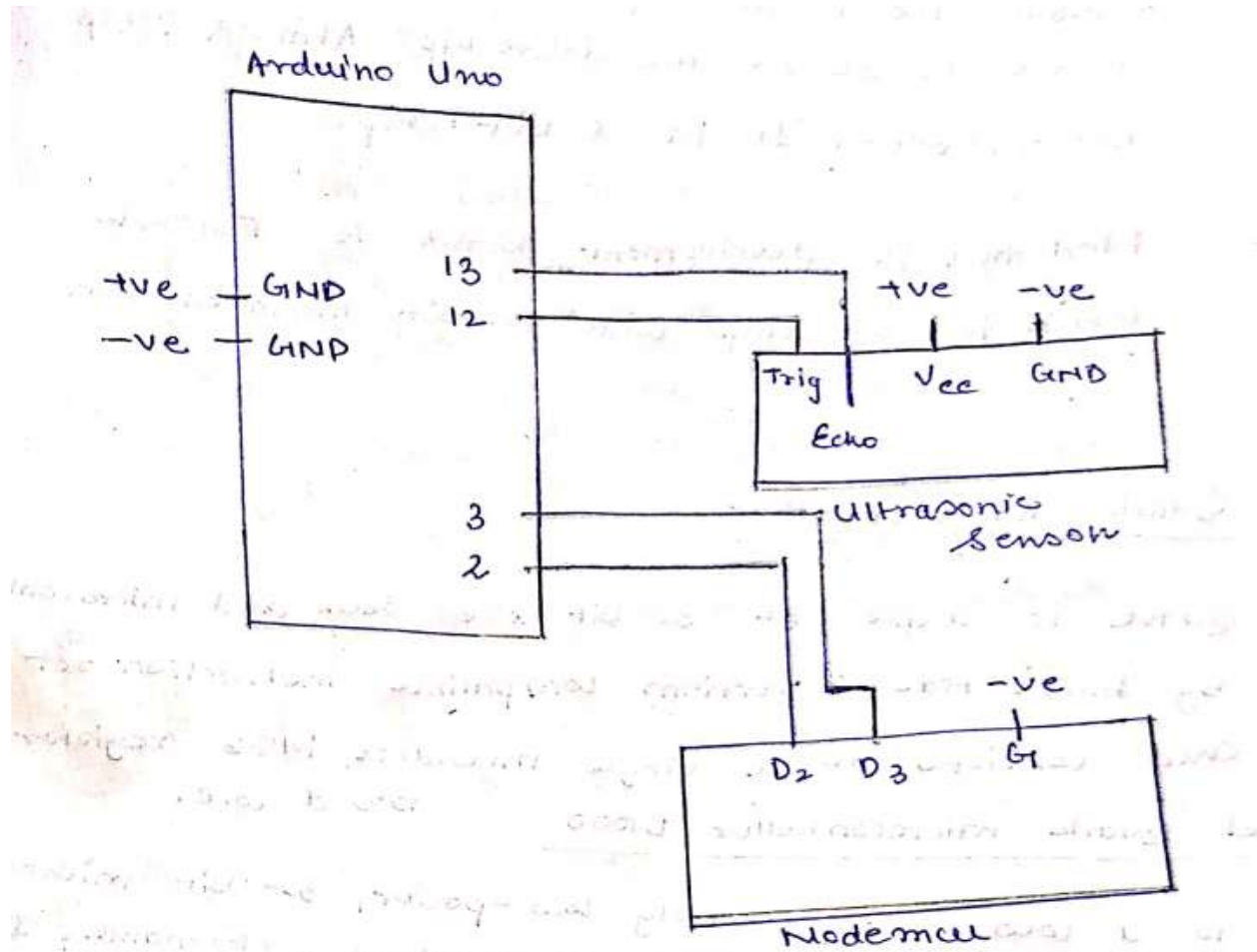
SOFTWARE REQUIREMENTS

1. Arduino IDE

Arduino IDE where IDE stands for Integrated Development Environment – An official software introduced by Arduino.cc, that is mainly used for writing, compiling and uploading the code in the Arduino Device. Almost all Arduino modules are compatible with this software that is open source software.

- It is available for operating systems like MAC, Windows, and Linux and runs on the Java Platform that comes with inbuilt functions and commands that play a vital role for debugging, editing and compiling the code in the environment.
- A range of Arduino modules available including Arduino Uno, Arduino Mega, Arduino Leonardo, Arduino Micro and many more.
- Each of them contains a microcontroller on the board that is actually programmed and accepts the information in the form of code.
- The main code, also known as a sketch, created on the IDE platform will ultimately generate a Hex File which is then transferred and uploaded in the controller on the board.
- The IDE environment mainly contains two basic parts: Editor and Compiler where former is used for writing the required code and later is used for compiling and uploading the code into the given Arduino Module.
- This environment supports both C and C++ languages.

DESIGN DIAGRAM



Circuit Diagram

FUNCTIONALITY OF THE PROJECT

TO SEND SMS TO ANY REGISTERED MOBILE NUMBER FROM ESP8266:

The ESP8266 is a powerful Wifi module. We directly programmed the ESP8266 module using the Arduino IDE. Once it's programmed we can send text messages from it to any pre-programmed mobile number. We used the **IFTTT Applets** to accomplish this task.

ESP8266 module can be configured both as AP or STA. Here we have configured it to work as station and have connected it to our Wifi Router. Once the connection is establish we have to find out a way to send SMS online. This online must also be easily accessible by out ESP8266 module. This is where we leverage the power of IFTTT (If This Then That) website. Using this website we can send SMS, E-mail, Whatsapp messages, Facebook updates, and Twitter tweets.

Creating an IFTTT Applet:

Step 1: Visit **www.IFTTT.com** and sign up for new ID if you don't have one already. After registering you will be sent a mail to your E-mail ID verify it and you will be logged into IFTTT.

Step 2: Search for **SMS Applet** or visit this link. Now Register your Mobile number with that Applet here I have used the number "009196*****". Always include the leading "00" followed by your country code and then your mobile number. Here I am from India hence my country code is "91" and my mobile number is "96*****". Once the number is entered click on "Send Pin" and verify your mobile number



Connect SMS

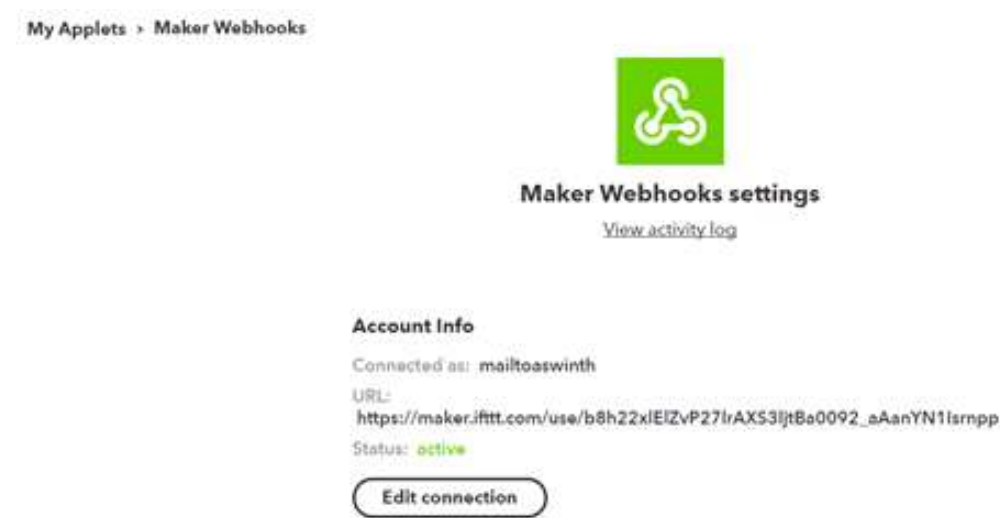
Enter your cell phone number to receive an activation PIN via SMS. All US carriers are supported. Some carriers outside of the US* are not supported yet. *For non-US numbers, include the leading "00" and country code. If you do not receive the PIN, your carrier may not be supported.

Your phone number:

00919612365498

Send PIN

Step 3: Now we have configured one Applet, we will configure another Applet called **Maker Webhooks**, so search for it. Now click on “Connect” and you will get the following Screen.

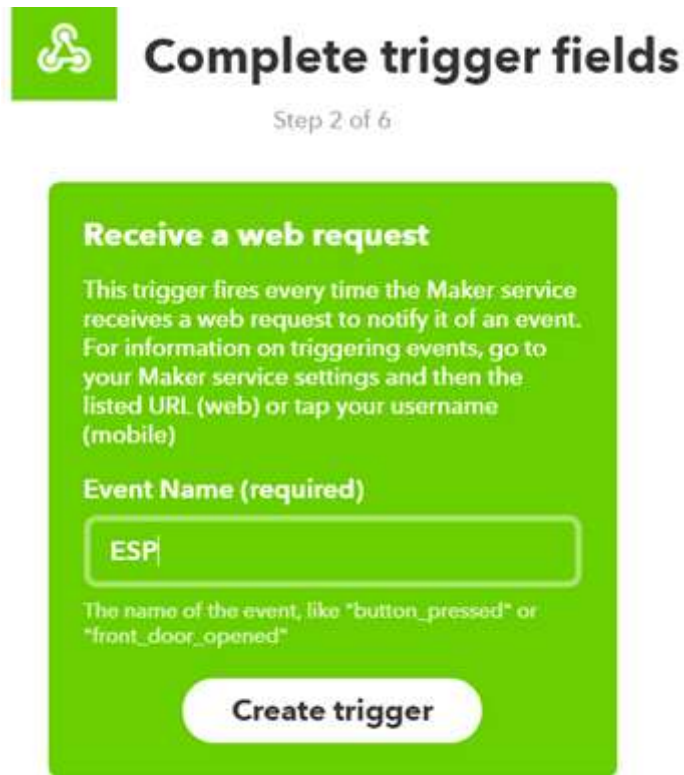


Step 4: Now it is time to **create our own Applet** that could sync both the above applets. To do this navigate to My Applets -> New Applet, or follow this link. You will be taken to this page.



Step 5: The term **IF THIS THEN THAT** means if something happens on the “This” then we have to do something on “that”. Here if the Maker Webhooks Applet is Triggered then an SMS must be sent. So click on “this” (the blue colour plus icon) and search for Maker Webhooks then click on it. Now, you will be asked to choose a Trigger, so click on “Receive a web request”

Step 6: Now you have to configure the Trigger by giving it an Event Name. I have named it “ESP” as shown below. You can use any event name, but remember this name for we have to use it later. Finally click on “Create Trigger”



Complete trigger fields

Step 2 of 6

Receive a web request

This trigger fires every time the Maker service receives a web request to notify it of an event. For information on triggering events, go to your Maker service settings and then the listed URL (web) or tap your username (mobile)

Event Name (required)

ESP

The name of the event, like "button_pressed" or "front_door_opened"

Create trigger

Step 7: You should get the below Screen where, you have to configure the “That” Applet. Click on the Plus icon near “that”

if  then  that

Step 8: Search for SMS Applet and click it. Then for choosing an Action click on “Send me SMS”. You will be taken to the below screen, where you have to enter the text message that has to be sent to your mobile. Finally click on “Create action”



Complete action fields

Step 5 of 6

Send me an SMS

This Action will send an SMS to your mobile phone.

Message (required)

The event named "{{EventName}}" occurred on the Maker service. SMS sent from ESP8266 - Circuit Digest

Add ingredient

Create action

Step 9: You can review and Finish your Applet, it should be looking something like this below. Click on "Finish"

Review and finish

Step 6 of 6



If maker Event "ESP", then send me an SMS at 00919688373635 (update phone number)

81/140

by mailtoaswinth

works with



Receive notifications when this Applet runs



Finish

Want to build even richer Applets?

Step 10: Now, search for Maker Webhooks and click on “Documentation”. You should see something like this below



This is a very important page. This page will show you the key and instructions on how to trigger an event. Your page will display a unique key for your ID.

Step 11: Now under “Make a Post or get web request”, you can see that we have an option to add the event name. Remember that in step 6 we created an Event named “ESP” so we have to use the same name and configure our URL like below.



Once you have changed the Event name click on “Test it”. You should receive a message to your registered mobile number. In our case the message should be “The even name ESP occurred on the Maker Service. SMS sent from ESP8266 –CircuitDigest” You can configure your own SMS as per your wish

That is it we now have an HTTPS URL which when triggered will send a particular message to a specified number.

WORKING OF THE PROJECT:

The ultrasonic sensor is programmed in such a way that as soon as it detects any object which is at a distance of 10cm or less it triggers the link. So as soon as a letter is dropped in the letter box the distance of the object being detected by the ultrasonic sensor falls in the range of 0 cm to 10cm and hence it triggers the link which in turn sends message to the registered mobile number.

SOFTWARE INTERFACE:

Arduino on ESP8266 (for NODEMCU)

This project brings support for ESP8266 chip to the Arduino environment. It lets you write sketches using familiar Arduino functions and libraries, and run them directly on ESP8266, no external microcontroller required.

ESP8266 Arduino core comes with libraries to communicate over WiFi using TCP and UDP, set up HTTP, mDNS, SSDP, and DNS servers, do OTA updates, use a file system in flash memory, work with SD cards, servos, SPI and I2C peripherals.

Installing with Boards Manager

Starting with 1.6.4, Arduino allows installation of third-party platform packages using Boards Manager.

- Install the current upstream Arduino IDE at the 1.8.7 level or later. The current version is at the Arduino website.
- Start Arduino and open Preferences window.
- Enter http://arduino.esp8266.com/stable/package_esp8266com_index.json into *Additional Board Manager URLs* field. You can add multiple URLs, separating them with commas.
- Open Boards Manager from Tools > Board menu and install *esp8266* platform (and don't forget to select your ESP8266 board from Tools > Board menu after installation).

CODE:

ARDUINO CODING:

```
#include<SoftwareSerial.h>

SoftwareSerial ArduinoUno(3,2);

const int trigPin = 12;

const int echoPin = 13;

long duration;

int distance;

void setup() {
```

```
pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output

pinMode(echoPin, INPUT); // Sets the echoPin as an Input

Serial.begin(115200);

ArduinoUno.begin(9600);

}

void loop() {

// Clears the trigPin

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

// Sets the trigPin on HIGH state for 10 micro seconds

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

// Reads the echoPin, returns the sound wave travel time in microseconds

duration = pulseIn(echoPin, HIGH);

// Calculating the distance

distance= duration*0.034/2;

// Prints the distance on the Serial Monitor

ArduinoUno.print(distance);

ArduinoUno.println("\n");

Serial.println(distance);

delay(1000);

}
```

NODEMCU CODING:

```
#include <ESP8266WiFi.h>

#include <WiFiClientSecure.h>

#include <SoftwareSerial.h>

const char* ssid = "Redmi4";

const char* password = "";


const char* host = "maker.ifttt.com";

const int httpsPort = 443;

const int API_TIMEOUT = 10000;

SoftwareSerial NodeMCU(D2,D4);

void setup() {

    // Open serial communications and wait for port to open:

    Serial.begin(115200);

    NodeMCU.begin(9600);

    pinMode(D2,INPUT);

    pinMode(D3,OUTPUT);

}


void loop() { // run over and over

    if(NodeMCU.available()>0){

        float val = NodeMCU.parseFloat();

        if(NodeMCU.read()=='\n'){
```



```
Serial.println(val);

if(val<10)

{

    Serial.println("activity");

    Serial.println();

    Serial.print("connecting to ");

    Serial.println(ssid);

    WiFi.begin(ssid, password);

    if (WiFi.status() != WL_CONNECTED) {

        delay(500);

        Serial.print(".");

    }

    Serial.println("");

    Serial.println("WiFi connected");

    Serial.println("IP address: ");

    Serial.println(WiFi.localIP());

    BearSSL::WiFiClientSecure client;

    client.setInsecure();

    client.setTimeout(API_TIMEOUT);

    Serial.print("connecting to ");

    Serial.println(host);

    if (!client.connect(host, httpsPort)) {

        Serial.println("connection failed");
```

```
        return;
    }

    String url = "https://maker.ifttt.com/trigger/ESP/with/key/oiUoLXmHdqUEkH25RtbFFZV_-jHSQcB9LApT5DgYBOP";

    Serial.print("requesting URL: ");

    Serial.println(url);


    client.print(String("GET ") + url + " HTTP/1.1\r\n" +

        "Host: " + host + "\r\n" +

        "User-Agent: BuildFailureDetectorESP8266\r\n" +

        "Connection: close\r\n\r\n");


    Serial.println("request sent");

    while (client.connected()) {

        String line = client.readStringUntil('\n');

        if (line == "\r") {

            Serial.println("headers received");

            break;

        }

    }

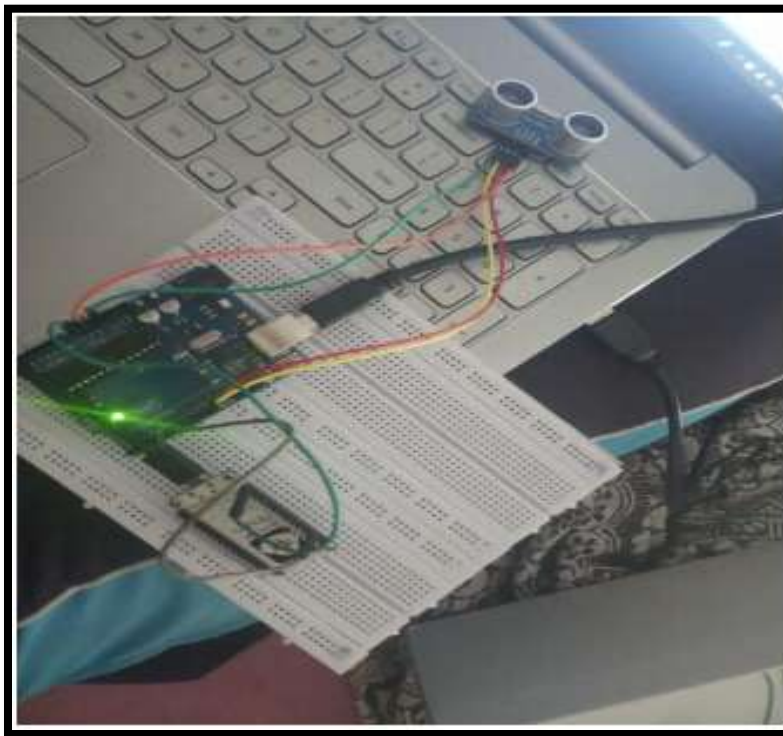
    String line = client.readStringUntil('\n');


    Serial.println("reply was:");

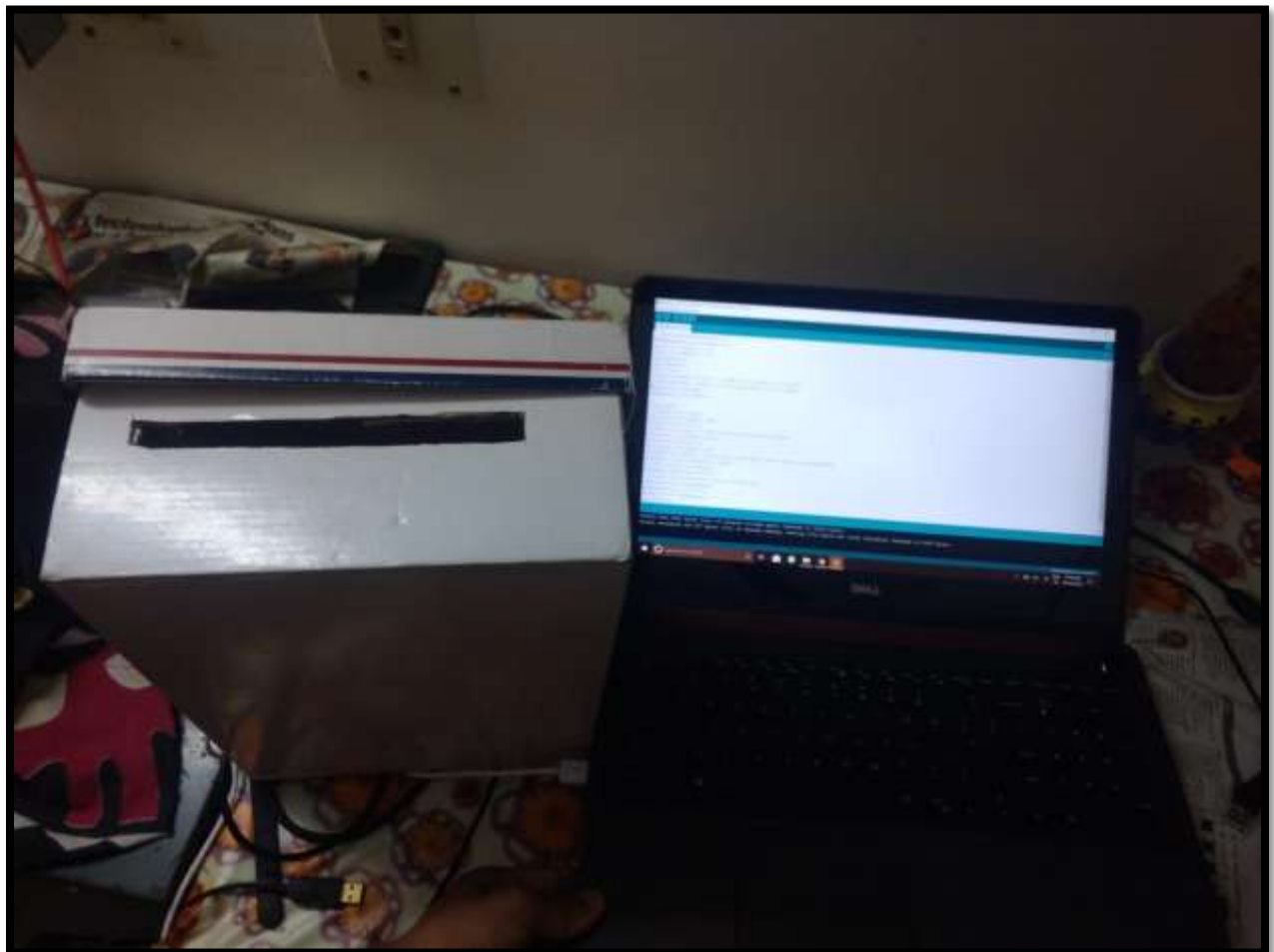
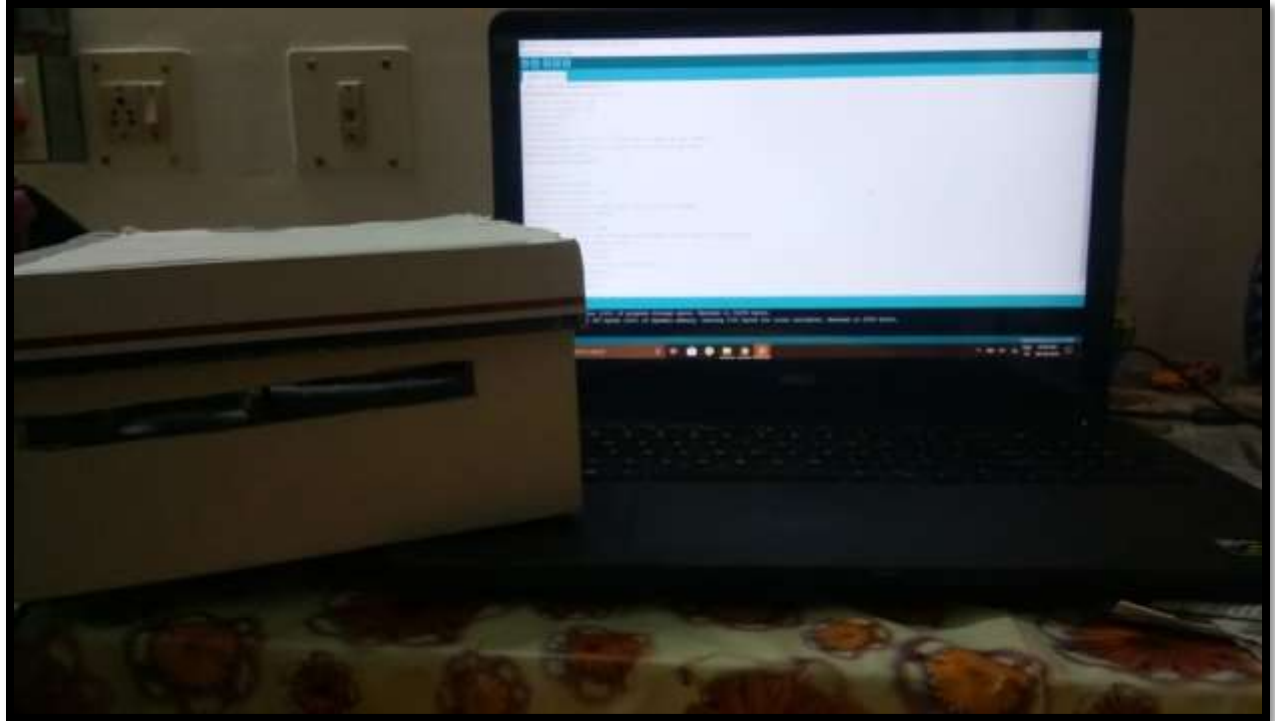
    Serial.println("=====");
```

```
Serial.println(line);  
  
Serial.println("=====");  
  
Serial.println("closing connection");  
  
    delay(10000);  
  
}  
  
}  
  
}  
  
delay(1000);  
  
}
```

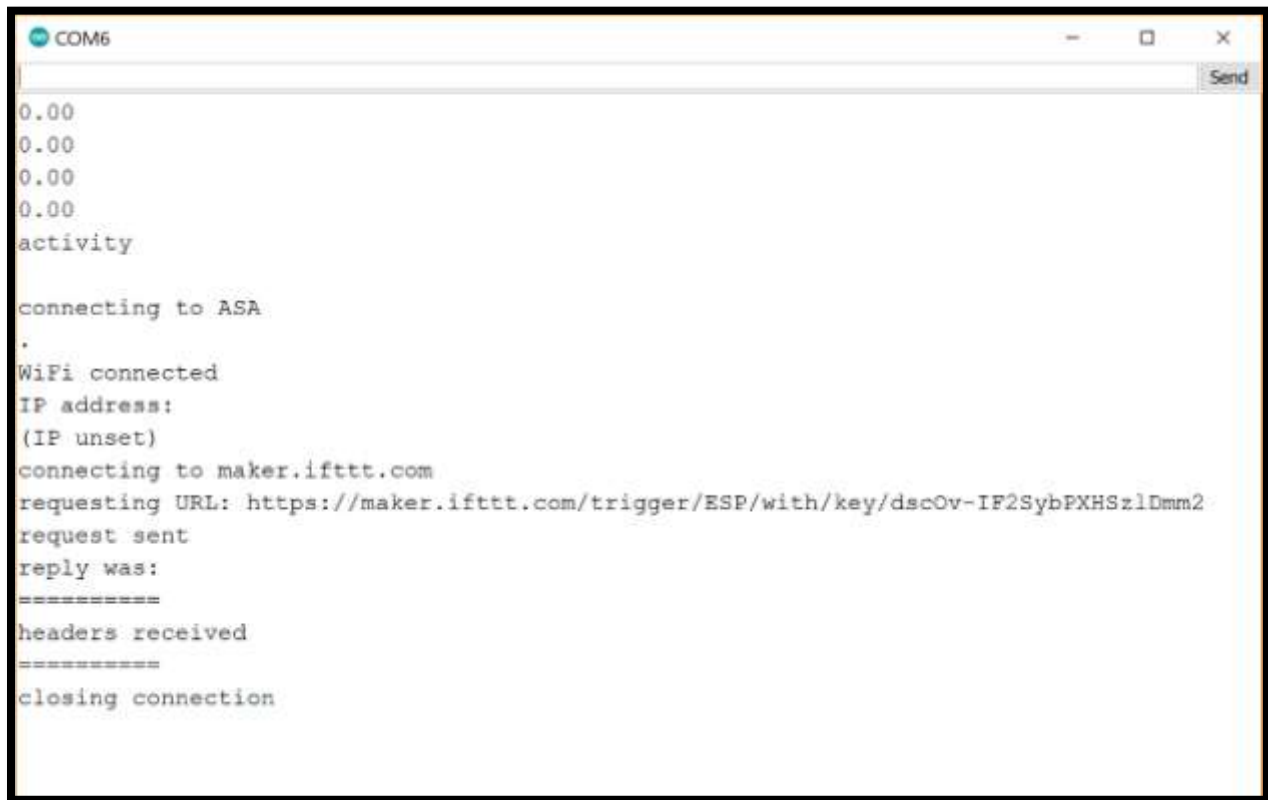
SNAPSHOTS:







OUTPUT:



```
COM6
0.00
0.00
0.00
0.00
activity

connecting to ASA
.
WiFi connected
IP address:
(IP unset)
connecting to maker.ifttt.com
requesting URL: https://maker.ifttt.com/trigger/ESP/with/key/dscOv-IF2SybPXHSzlDmm2
request sent
reply was:
=====
headers received
=====
closing connection
```

1:07

4G 58



VK-047015



28-2 9:00

YOU HAVE RECEIVED A NEW LETTER.

28-2 9:59

YOU HAVE RECIEVED A NEW LETTER
IN YOUR LETTER BOX

11-3 17:56

YOU HAVE RECEIVED A NEW LETTER.
[March 11, 2019](#) at [05:56PM](#)

11-3 18:06

YOU HAVE RECEIVED A NEW LETTER.
[March 11, 2019](#) at [06:05PM](#)

YOU HAVE RECEIVED A NEW LETTER.
[March 11, 2019](#) at [06:07PM](#)

YOU HAVE RECEIVED A NEW LETTER.
[March 11, 2019](#) at [06:06PM](#)

11-3 18:28

Hey Rashi! you got a new letter at
[March 11, 2019](#) at [06:28PM](#)



Text message

