Bangabandhu Sheikh Mujibur Rahman Digital University

Course Title: Industrial IoT

Course Code: ICT 4251

Project Report

Project Title: IoT Based Smart Area

SUBMITTED TO:

Farzana Akter
Lecturer, Internet of Things (IoT) Program,
Department of Information & Communication Technology (ICT), BDU.

SUBMITTED BY:

Sharika Khan ID: 1901013

MD. Al-amin ID: 1901027

Tasnima Hamid ID: 1901043

Session: 2019-2020
Second Year, Second Semester,
Internet of Things (IoT) Program.

Date of Submission: 26th July 2022

Table of Contents

Acknowledgements:

We would like to express our sincere gratitude to several individuals for supporting us throughout the project. First, we wish to express our sincere gratitude to our teacher, Farzana Akter, for her guidance, insightful comments, helpful information, practical advice and unceasing ideas that have helped us tremendously at all times in our project. We also wish to express our sincere thanks to our honorable teachers, our families and our friends who were with us all the time.

Abstract:

Considering Bangladesh, arson is a risky accident. We can get a general idea of how dangerous it is if we look at the statistics. Additionally, the country has lost money as a result of these fires. These events can be decreased if we take the right steps at the right time. Because of this, we developed our system. Each home in the community/area will have a flame sensor. There will be a tower with an alarm system nearby. The flame sensor will measure the intensity of the fire and issue the initial alerts. The nearby tower will then sound an alarm to notify everyone in the area after receiving a signal from the first device. Fire detectors detect one or more byproducts of fire. Systems for fire detection shorten response times by notifying the proper parties to put out the fire. The amount of property damage is thereby decreased. Sprinklers can be connected to fire detection systems so that they will turn on instantly if a fire is found. One of the finest and least expensive ways to provide an early warning of a potentially deadly fire is through the use of fire system alarms, which can reduce the risk of mortality from a fire in your house by over half when installed and maintained properly.

Introduction:

Arson is a dangerous accident in the context of Bangladesh. If we see the statistics, we can see a glimpse of its danger level. Between January 1, 1999, and December 31, 2020, there were approximately 285,000 fires in the country, according to data published by the Fire Service and Civil Defense. The country has also suffered a financial loss of roughly Tk 69 billion as a result of these fires. Furthermore, according to fire service data, at least 2,308 persons died in fires across the country between 2004 and 2020. With 24,074 fire events, 2019 was the year with the most, followed by 2020 with 21,073. According to Fire Service and Civil Defense figures, there were roughly 99,752 fires in the country between January 1, 2016 and December 31, 2020. In that time, 71,684 fires (or more than 71% of all fires) were caused by electric disturbances, various forms of burners, and burning cigarettes. Electric disturbances were responsible for 37,044 (37%) of the total events, followed by 19,124 (19.17%) from various types of burners and 15,536 (15.57%) from burning cigarettes. Even while proposing this project, a fire incident was reported in Boubajar, Lalbagh, Dhaka. Fortunately, no casualties were reported, but the factory and goods were all burnt. These incidents can be reduced if we take appropriate measures and act on time. To act on time, we have created our system.

Requirement analysis

The components needed for this project is mentioned below:

For each house,

- Esp8266
- Flame Sensor
- Buzzer
- Led
- Breadboard
- Jumper wires

For towers,

- Esp8266
- Buzzer
- Led
- LCD Display (Optional)

- Breadboard
- Jumper wires

Design and Implementation

Fire Detection: Fire detectors pick up on one or more fire-related byproducts or phenomena, such as smoke, heat, infrared and/or UV radiation, gas, or smoke. Smoke detectors are frequently standalone equipment in homes.

Our project works in the following procedure-

- Every house in the area/society will have a flame sensor. It will be covered with a thin cover(silk/paper). In the area, there will be a tower with an alarm system.
- If any house catches fire, the cover will melt and expose the flame sensor to the fire.
- The flame sensor will receive the fire intensity and will give primary alerts.
- Then it will send signal to the tower in the area, which will ring an alarm to alert the whole area. It will also send a message to the fire service.

For this, we used the following products:

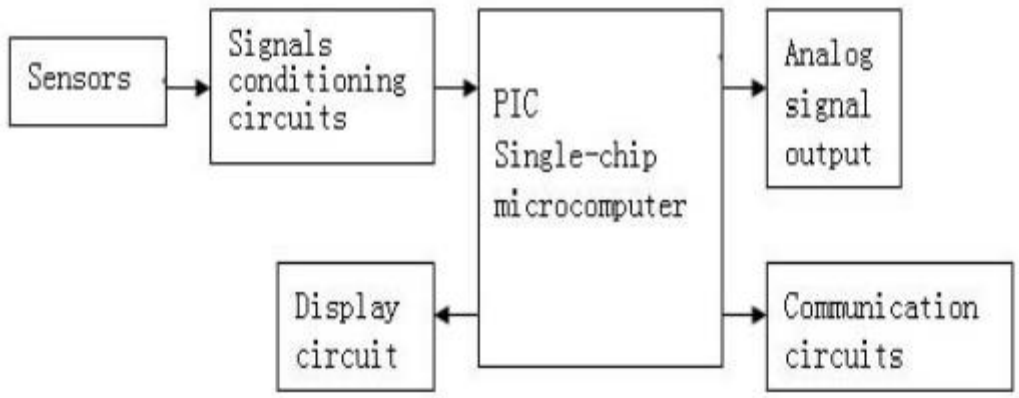
About Flame Detector: A flame detector is a sensor created to recognize the presence of a flame or fire and act accordingly, enabling flame detection. This gadget can distinguish between smokeless liquid and smoke that can start an open fire. Flame detectors are frequently used in boiler furnaces; typically, a flame detector can detect heat, smoke, and fire. According to the temperature and velocity of the air, this device can also detect fires. Typically optical devices, flame detectors may react to flame in less than a second. Depending on the installation, the flame detector would react to the flame by either activating a fire suppression system or sounding an alert and deactivating the fuel line. Depending on the installation, possible responses to a flame detection include sounding an alarm, turning off a fuel line (such as a propane or a natural gas line), and turning on a fire suppression system. When employed in industrial furnaces, for example, their purpose is to certify that the furnace is operating properly. They can also be used to turn off the ignition system, however frequently they don't do anything more than alert the operator or control system. Due to the mechanics it uses to detect flames, a flame detector can frequently react quicker and more precisely than a smoke or heat detector.

Figure: Flame Sensor

How Flame Detector Works: The sensors in the flame detector will pick up the radiation that the flame emits. The flame detector's sensor is a photoelectric sensor, which will convert the flame's radiant intensity signal to an appropriate voltage signal. A single chip microcomputer will then process this signal and convert it to an output. A UV/IR flame detector is made up of UV and IR sensors that are attached to one another in a single device. UV sensors are sensitive to a variety of combustible fuels, including hydrocarbons, sulfur, hydrazine, and ammonia, and they operate by sensing the UV light released by the flame. A furnace flame sensor operates by determining whether a flame is present inside the furnace. The sensor, which is a brief, thin metallic rod, generates a tiny electrical current to verify that there is fire burning inside the furnace.

A flame detector, which is the main part of a flame detection system, is made up of photoelectric detective circuits, signal conditioning circuits, microprocessor systems, I/O circuits, and wind cooling circuits

How Flame Detectors Detect: Flame detectors that use infrared (IR) or wideband infrared (1.1 m and higher) technology scan the infrared spectrum for distinct patterns emitted by hot gasses. These are detected using a specific sort of thermographic camera called a fire-fighting thermal imaging camera (TIC).



Block diagram of the flame detector

ESP8266: A low-cost Wi-Fi microchip with microcontroller and TCP/IP networking capabilities is called the ESP8266. The maker community in English helped to popularize the chip. With the use of Hayes-style commands, this tiny module enables microcontrollers to join a Wi-Fi network and establish straightforward TCP/IP connections. However, initially, there was hardly any information available in English on the chip and the orders it would receive. Many hackers were drawn to the module, the chip, and the software on it as well as to translate the Chinese documentation because of the extremely low price and the possibility that it may ultimately be produced in large quantities at very low cost.Microcontrollers may connect to 2.4 GHz Wi-Fi using IEEE 802.11 thanks to the ESP8266 module. It can be used as a self-sufficient MCU by running an RTOS-based SDK, or it can be utilized with ESP-AT firmware to offer Wi-Fi connectivity to external host MCUs.

How ESP8266 Works: A self-contained SOC with an integrated TCP/IP protocol stack, the ESP8266 WiFi Module allows any microcontroller to access your WiFi network. The ESP8266 can offload all Wi-Fi networking tasks from another application processor or hosting an application.

ESP8266 In IoT: Espressif Systems makes the ESP8266, a system on a chip (SOC) Wi-Fi microprocessor for Internet of Things (IoT) applications. The ESP8266 is currently widely used throughout Internet of Things devices due to its low price, compact size, and compatibility with embedded devices. Although the ESP8266 has been replaced by the ESP32 microcontroller chip of a newer generation, IoT developers and manufacturers still like it.

ESP8266 Modules: Espressif, Ai-Thinker, and a few other manufacturers produce the surface-mountable modules with the chip that are prepared to be placed into an MCU. The FCC has typically shielded and permitted their use.

Figure: Esp8266

ESP8266 Development Boards: These are fully assembled IoT MCU development boards with preloaded modules. They are employed by designers and manufacturers to produce prototypes before beginning production. Development boards are made by numerous companies, and each model has a particular set of specs. When comparing ESP8266 IoT development board possibilities, it's important to be aware of some key parameters, such as:

- GPIO pins
- ADC pins
- Wi-Fi antennas
- LEDs
- Shielding*
- Flash Memory

Many international markets demand protected Wi-Fi equipment because shielding reduces the amount of Radio Frequency Interference (RFI) that Wi-Fi generates. Because of this, producers of embedded devices and developers alike should give this careful thought.

ESP Communication: Espressif created the connectionless communication protocol known as ESP-NOW, which supports brief packet transmission. Without the need of Wi-Fi, many devices can communicate with one another using this protocol.Small messages (up to 250 bytes) can be exchanged quickly between ESP32 boards using this protocol. With ESP-NOW, you can have one-way or two-way communication in a variety of settings.

Master Slave Communication: A hardware communication type known as master/slave allows for unidirectional control of one or more devices by one device. This is frequently employed in the realm of electronic gear, where one item serves as the controller and all other devices are the ones that are controlled. In essence, one is the master, while the rest are his or her slaves to be manipulated. The master/slave configuration of IDE disk drives connected on the same cable, where the master is the primary drive and the slave is the secondary drive, is the most typical illustration of this. One device or process (the "master") controls one or more other devices or processes (the "slaves") and acts as their communication hub in the master/slave model of asymmetric communication or control. In some systems, a master is chosen from a selection of eligible devices, while the remaining devices serve as slaves. The phrase "master" and "slave" were first used in 1904. Due to its relationship with slavery, it has been a contentious topic since the beginning of the twenty-first century, and organizations and companies have started using other terminology in their place. One device or process (referred to as the master) has complete control over one or more other devices or processes in the master/slave model for communication protocol (known as slaves).

Testing and deployment

Figure: The setup and real-life simulation

Figure: Home esp8266 is continuously sending data to tower esp8266

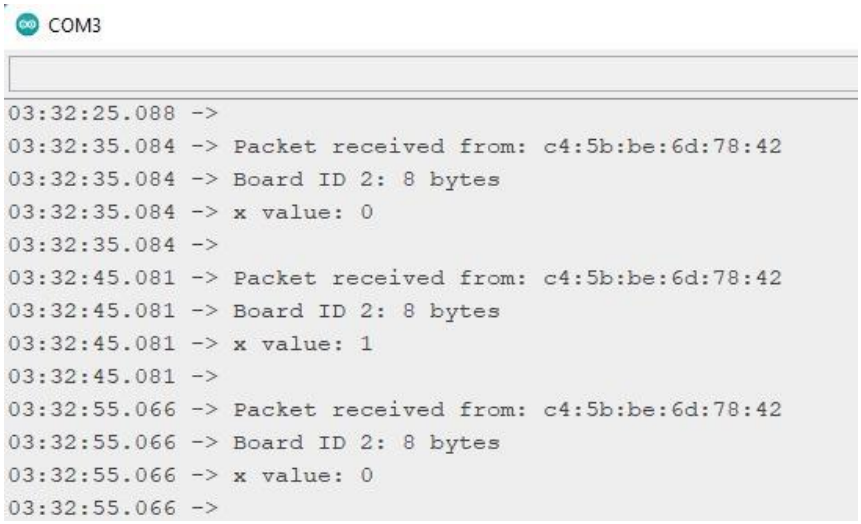


Figure: Tower esp8266 is continuously receiving data

Future enhancements

Fire detection systems reduce reaction times by alerting the appropriate persons to put out the fire. As a result, the quantity of property damage is reduced. Fire detection systems can be linked to sprinklers, which will activate immediately if a fire is detected. Fire System alarms that are properly installed and maintained are one of the best and least expensive ways to provide an early warning of a potentially dangerous fire, and they can lower the risk of death from a fire in your home by nearly half.

References:

[1] https://randomnerdtutorials.com/esp-now-esp8266-nodemcu-arduino-ide/
[2] https://randomnerdtutorials.com/esp-now-many-to-one-esp8266-nodemcu/
[3] https://www.survivingwithandroid.com/esp-now-esp32-esp8266/
[4] https://thefinancialexpress.com.bd/national/fire-incidents-increase-four-fold-to-285000-in-two-decades-1636801072

Appendix:

Source Code:

Esp8266 Sender Code:

```
#include <ESP8266WiFi.h>
#include <espnow.h>


char ssid[] = "Tasnima"; //Enter the wifi name
char pass[] = "12345678";// Enter the wifi password

// REPLACE WITH RECEIVER MAC Address
uint8_t broadcastAddress[] = {0xbc, 0xdd, 0xc2, 0x9d, 0x5f, 0x0b};
//bc:dd:c2:9d:5f:0b

// Set your Board ID (ESP32 Sender #1 = BOARD_ID 1, ESP32 Sender #2 = BOARD_ID 2, etc)
#define BOARD_ID 2
int led = 13; // Connected to D7 pin of NodeMCU
int flame = 12; //Connected to D6 pin of NodeMCU
int buzzer = 4; //Connected to D2 pin of NodeMCU

// Structure example to send data
// Must match the receiver structure
typedef struct struct_message {
   int id;
   int x;
} struct_message;

// Create a struct_message called test to store variables to be sent
struct_message myData;

unsigned long lastTime = 0;
unsigned long timerDelay = 10000;

// Callback when data is sent
void OnDataSent(uint8_t *mac_addr, uint8_t sendStatus) {
  Serial.print("\r\nLast Packet Send Status: ");
  if (sendStatus == 0){
    Serial.println("Delivery success");
  }
  else{
    Serial.println("Delivery fail");
  }
}

void setup() {
  // Init Serial Monitor
```

```
  Serial.begin(115200);
  WiFi.begin(ssid, pass);

  // Set device as a Wi-Fi Station
  WiFi.mode(WIFI_STA);

  pinMode(flame, INPUT);
  pinMode(buzzer, OUTPUT);
  pinMode(led, OUTPUT);

  // Init ESP-NOW
  if (esp_now_init() != 0) {
    Serial.println("Error initializing ESP-NOW");
    return;
  }
  // Set ESP-NOW role
  esp_now_set_self_role(ESP_NOW_ROLE_CONTROLLER);

  // Once ESPNow is successfully init, we will register for Send CB to
  // get the status of Trasnmitted packet
  esp_now_register_send_cb(OnDataSent);

  // Register peer
  esp_now_add_peer(broadcastAddress, ESP_NOW_ROLE_SLAVE, 1, NULL, 0);

}

void loop() {
  if ((millis() - lastTime) > timerDelay) {
    // Set values to send
    myData.id = BOARD_ID;
    myData.x = digitalRead(flame);
    // Send message via ESP-NOW
    esp_now_send(0, (uint8_t *) &myData, sizeof(myData));
    lastTime = millis();
  }

  if (digitalRead(flame) == 0)
    {
    //Blynk.virtualWrite(V1, 255);
    digitalWrite(led, HIGH);
    digitalWrite(buzzer, HIGH);
    }
  else
      {
      digitalWrite(led, LOW);
      digitalWrite(buzzer, LOW);
```

```
  }
}
```

Esp8266 Receiver Code:

```
#include <ESP8266WiFi.h>
#include <espnow.h>

int led = 13; // Connected to D7 pin of NodeMCU
int flame_sensor = 12; //Connected to D6 pin of NodeMCU
int buzzer = 4; //Connected to D2 pin of NodeMCU

char ssid[] = "Tasnima"; //Enter the wifi name
char pass[] = "12345678";// Enter the wifi password

// Structure example to receive data
// Must match the sender structure
typedef struct struct_message {
    int id;
    int x;
} struct_message;

// Create a struct_message called myData
struct_message myData;

// Create a structure to hold the readings from each board
struct_message board1;
struct_message board2;

// Create an array with all the structures
struct_message boardsStruct[2] = {board1, board2};

// Callback function that will be executed when data is received
void OnDataRecv(uint8_t * mac_addr, uint8_t *incomingData, uint8_t len) {
  char macStr[18];
  Serial.print("Packet received from: ");
  snprintf(macStr, sizeof(macStr), "%02x:%02x:%02x:%02x:%02x:%02x",
           mac_addr[0], mac_addr[1], mac_addr[2], mac_addr[3], mac_addr[4],
mac_addr[5]);
  Serial.println(macStr);
  memcpy(&myData, incomingData, sizeof(myData));
  Serial.printf("Board ID %u: %u bytes\n", myData.id, len);
  // Update the structures with the new incoming data
  boardsStruct[myData.id-1].x = myData.x;
  Serial.printf("x value: %d \n", boardsStruct[myData.id-1].x);
  Serial.println();
```

```
  if(myData.x == 1)
  {
    digitalWrite(led, HIGH);
    digitalWrite(buzzer, HIGH);
    }
}

void setup() {
  // Initialize Serial Monitor
  Serial.begin(115200);
  WiFi.begin(ssid, pass);

  // Set device as a Wi-Fi Station
  WiFi.mode(WIFI_STA);

  // Init ESP-NOW
  if (esp_now_init() != 0) {
    Serial.println("Error initializing ESP-NOW");
    return;
  }

   pinMode(led, OUTPUT);
  pinMode(buzzer, OUTPUT);

  digitalWrite(led, LOW);
  digitalWrite(buzzer, LOW);

  // Once ESPNow is successfully Init, we will register for recv CB to
  // get recv packer info
  esp_now_set_self_role(ESP_NOW_ROLE_SLAVE);
  esp_now_register_recv_cb(OnDataRecv);
}

void loop(){
  // Access the variables for each board
  int board1X = boardsStruct[0].x;
  int board2X = boardsStruct[1].x;
}
```

--- END OF THE REPORT ---