PHASE 3

AIR QUALITY MONITORING:

The purpose of this IOT project is to monitor the Air Quality by measuring the AQI(Air Quality Index) value based only on particulate matter.

DEVICE REQUIREMENT:

1. Arduino Board - UNO
2. PM2.5 Sensor (Particulate Matter)
3. ESP8266 Wifi Chip
4. Bread Board
5. Arduino - UNO Upload Cable

STEPS TO BUILD THE IOT DEVICE:

1. Integrating Arduino UNO with IDE.
2. Integrating PM2.5 Sensor with Arduino UNO
3. Integrating ESP8266 Sensor with Arduino UNO
4. Testing the device.

CODES USED:

Integrating PM2.5 Sensor with Arduino UNO:

```
#include <SoftwareSerial.h>
SoftwareSerial pmsSerial(2, 3);

void setup() {
  // our debugging output
  Serial.begin(115200);

  // sensor baud rate is 9600
  pmsSerial.begin(9600);
}

struct pms5003data {
```

```
  uint16_t framelen;
  uint16_t pm10_standard, pm25_standard, pm100_standard;
  uint16_t pm10_env, pm25_env, pm100_env;
  uint16_t particles_03um, particles_05um, particles_10um, particles_25um, particles_50um,
particles_100um;
  uint16_t unused;
  uint16_t checksum;
};

struct pms5003data data;

void loop() {
  if (readPMSdata(&pmsSerial)) {
    // reading data was successful!
    Serial.println();
    Serial.println("---------------------------------------");
    Serial.println("Concentration Units (standard)");
    Serial.print("PM 1.0: "); Serial.print(data.pm10_standard);
    Serial.print("\t\tPM 2.5: "); Serial.print(data.pm25_standard);
    Serial.print("\t\tPM 10: "); Serial.println(data.pm100_standard);
    Serial.println("---------------------------------------");
    Serial.println("Concentration Units (environmental)");
    Serial.print("PM 1.0: "); Serial.print(data.pm10_env);
    Serial.print("\t\tPM 2.5: "); Serial.print(data.pm25_env);
    Serial.print("\t\tPM 10: "); Serial.println(data.pm100_env);
    Serial.println("---------------------------------------");
    Serial.print("Particles > 0.3um / 0.1L air:"); Serial.println(data.particles_03um);
    Serial.print("Particles > 0.5um / 0.1L air:"); Serial.println(data.particles_05um);
    Serial.print("Particles > 1.0um / 0.1L air:"); Serial.println(data.particles_10um);
    Serial.print("Particles > 2.5um / 0.1L air:"); Serial.println(data.particles_25um);
    Serial.print("Particles > 5.0um / 0.1L air:"); Serial.println(data.particles_50um);
    Serial.print("Particles > 10.0 um / 0.1L air:"); Serial.println(data.particles_100um);
    Serial.println("---------------------------------------");
  }
}

boolean readPMSdata(Stream *s) {
  if (! s->available()) {
    return false;
  }
```

```
// Read a byte at a time until we get to the special '0x42' start-byte
if (s->peek() != 0x42) {
  s->read();
  return false;
}

// Now read all 32 bytes
if (s->available() < 32) {
  return false;
}

uint8_t buffer[32];
uint16_t sum = 0;
s->readBytes(buffer, 32);

// get checksum ready
for (uint8_t i=0; i<30; i++) {
  sum += buffer[i];
}

/* debugging
for (uint8_t i=2; i<32; i++) {
  Serial.print("0x"); Serial.print(buffer[i], HEX); Serial.print(", ");
}
Serial.println();
*/

// The data comes in endian'd, this solves it so it works on all platforms
uint16_t buffer_u16[15];
for (uint8_t i=0; i<15; i++) {
  buffer_u16[i] = buffer[2 + i*2 + 1];
  buffer_u16[i] += (buffer[2 + i*2] << 8);
}

// put it into a nice struct :)
memcpy((void *)&data, (void *)buffer_u16, 30);

if (sum != data.checksum) {
  Serial.println("Checksum failure");
```

```
   return false;
  }
  // success!
  return true;
}
```

Connecting ESP8266 Sensor to ThinkSpeak:

```
#include “ThinkSpeak.h
#include <ESP8266WiFi.h>
char networkname[] = “”; // your network name
char passcode[] = “”; // your passcode
WiFiClient client;
unsigned long tsChannelID = ; // ThingSpeak Channel ID
const char * tsWriteAPIKey = “”; //ThingSpeak Write API Key
String airQuaility = “”;
const int fieldOne = 1;
void setup()
{
Serial.begin(115200);
WiFi.mode(WIFI_STA);
ThingSpeak.begin(client);
thingSpeak();
}
void loop()
{
thingSpeak();
if (Serial.available() > 0)
{
while (Serial.available() > 0)
{
int inChar = Serial.read();
airQuaility += (char)inChar;
}
}
pushData();
}
void thingSpeak()
{
if (WiFi.status() != WL_CONNECTED)
```

```
{
while (WiFi.status() != WL_CONNECTED)
{
WiFi.begin(networkname, passcode);
delay(5000);
}
}
}
void pushData()
{
int getData = ThingSpeak.writeField(tsChannelID, fieldOne, airQuaility, tsWriteAPIKey);
if (getData != 200)
{
delay(15000);
pushData();
}
airQuaility = “”;
}
```