T.C.
GEBZE TEKNİK ÜNİVERSİTESİ

Elektronik Mühendisliği Bölümü

ELM 491
BİTİRME PROJESİ
II

HAVA KALİTE KONTROL KİTİ

Yusuf Hamdi Saylam 151024069
Ömer Konan 171024085

Danışman
Dr. Öğr. Üyesi Atilla Uygur

Haziran 2020
Gebze, KOCAELİ

ÖNSÖZ

Bu kılavuzun ilk taslaklarının hazırlanmasında emeği geçenlere, projemizin geliştirilmesinde yol gösterici olan Sayın Dr. Ört. Üyesi Atilla Uygur hocamıza içten teşekkürlerimizi sunarız.

Haziran, 2020

Ömer Konan
Yusuf Hamdi Saylam

İÇİNDEKİLER

ŞEKİL LİSTESİ

ÖZET

Günlük yaşamda bina içinde gaz kaçakları, yangın ve iç ortamın hava kalitesi bina işleticileri tarafından ele alınan problemler arasındadır. Bu sorunun çözümü olarak düş güç tüketen, bulut sistemine bağlı, sensörlerden alınan verilerin bulut sistemine kaydedilerek merkezi sistemden hava kalitesinin kontrol edildiği bir kit geliştirilmeye karar verdik.

İki aşama olarak ayrılan projede birinci aşamada sensörlerin ve kullanılacak kitlerin seçilmesi, bu kitlerle sensörlerden alınan verilerin okunması hedeflenmiştir. Hedeflenen verilerin buluta sisteme aktarılması ve sonuçlar çıkarılması projenin ikinci kısmında gerçeklenecektir.

Proje kapsamında kullanılan sensörlerden CO, CO2, Sıcaklık, Nem, Duman verileri sayısal olarak toplanacak ve bu veriler ile hava kalitesine dair sonuçlar çıkarılacaktır. Yapılan bu kit sayesinde toplanan veriler ile daha sonrasında makine öğrenimi algoritmaları kullanılarak, oluşabilecek sorunlar önceden tespit edilebilir ve erken müdahale mümkün olabilir. Bu proje makine öğrenimi algoritmaları çalışmaları için veri ihtiyacını gidermede kullanılabilecek. Fakat proje kapsamında makine öğrenimi algoritmaları test edilmeyecektir.

,

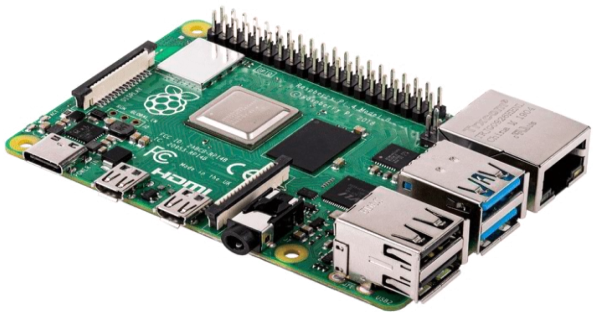
GİRİŞ

Bu proje GTÜ ELM 491 Bitirme Çalışması I dersi kapsamında hazırlanmaktadır. Projede uzaktan bulut tabanlı hava kalite kontrol kiti geliştirilecektir.. Proje sonucunda bulunulan ortamın hava kalitesini gösteren verileri toplanacak ve bu veriler bulut sistemine aktarılacaktır.

1. MALZEMELER

Bu bölümde projede kullanılan malzemeler tanıtılmış ve kullanım nedenleri açıklanmıştır.

1.1 - Raspberry PI Bilgisayar



Şekil 1

Özellikler

- 1.5 GHz dört çekirdekli ARM Cortex-A72 CPU
- 4GB LPDDR4 RAM SKU | 2GB Modeli
- VideoCore VI Grafikleri
- 4kp60 HEVC video
- Gerçek Gigabit Ethernet
- 2 × USB 3.0 ve 2 × USB 2.0 bağlantı noktalar
- 2 × mikro HDMI bağlantı noktası (1 × 4K@60Hz veya 2 × 4K@30Hz)

Geliştirilmesi hedeflenen kitin bütün işlemlerini kontrol eden düşük güç düketen, güçlü ve uygun fiyatlı bir bilgisayar. En büyük özelliklerinden biri küçük hacimli olması ve digital girdi almak için yeterli pinlere sahip olması. Ayrıca içersinde barındırdığı UART, I2C gibi haberleşme protokolleri ile sensör bağlantılarını kolay gerçekleştirilebilmesi ve sıradan bir bilgisayar gibi bulut sistemlerine bağlanabilmesi için wifi ve işletim sistemi kolaylığı sağlaması proje için seçilmesine sebep olmuştur.

1.2 5V 3A Güç Kaynağı

Projenin güç isterlerinin karşılanabilmesi için gerekli gerilim ve akım değerlerini sağlayacak olan adaptör, Raspberry PI'ın kutu içeriğinde bulunan 5V 3A USB-C portlu adaptör kullanılmıştır.

Şekil 2

1.3 Mikrodenetleyici

Şekil 3

Arduino UNO

Çalışma Gerilimi: 5V

Dijital I/O Pinleri: 14 (6 tanesi PWM çıkışı)

Analog Giriş Pinleri: 6

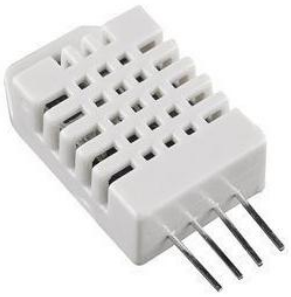
Her I/O için Akım: 40 mA

3.3V Çıkış için Akım: 50 mA

Boyut: 68.6*53.4 mm

Sensörler içerisinde analog çıkış veren sensörler olması ve kullandığımız ana bilgisayarımızın adc dönüştürücüye sahip olmaması, proje ekipman seçiminde adc dönüştürücü arayışına sebep oldu. Hali hazırda elimizde bulunan Arduino Uno'nun bu özelliğe sahip olması proje masraflarını kısmak ve proje geliştirme sürecinde dünyada yayılan COVID 19 nedeniyle aranan elektronik eşyaların temininde sıkıntılar yaşanması nedeniyle elimizde bulunan elemanlarla sorunları çözme kararı alındı. Mikrodenetleyicinin analog girişi ile okunan sensör verilerini ana işlemcimizi barındıran Raspberry PI'a iletmemiz gerekti. Bu problem de seri haberleşme yöntemi ile çözüldü. Böylelikle Arduino ile Raspberry PI arasındaki USB bağlantısı, hem mikrodenetleyicinin gücünün Raspberry PI üzerinden sağlanması hem de ikisi arasındaki haberleşmenin gerçekleştirilmesi sağlandı.

1.4 - DHT22 Sıcaklık Nem Sensörü



Özellikleri:

.

Dijital çıkış vermektedir.
Çalışma Gerilimi: 3.3-5 VDC
Nem: 0-100%RH
Sıcaklık: -40 - 80 °C
Hassasiyet:
Nem: +/- %3 (Max %5) RH
Sıcaklık: < +/- 1°C
Ölçüm Periyodu: 2 s
Ürün Ölçüleri: 22x28x5 mm

Şekil- 4

DHT22 sıcaklık ve nem algılayıcı kalibre edilmiş dijital sinyal çıkışı veren gelişmiş bir sensör birimidir. Yüksek güvenilirliktedir ve uzun dönem çalışmalarda dengelidir. 8 bit mikroişlemci içerir, hızlı ve kaliteli tepki verir.

-40 ile 80°C arasında +/-1°C hata payı ile sıcaklık ölçen birim, 0-100% RH arasında +/-5% RH hata payı ile nem ölçümü yapabilmektedir. Sensör ölçümü olarak sensörün data toplama periyodundan kaynaklı olarak 2 saniyelik periyotlarla ölçüm sonuçları alınabilmektedir.

1.5 - MH-Z19 Kızılötesi CO2 Ölçme Sensör Modülü

Özellikleri:

- Hedef Gaz: CO2 / Karbondioksit
- Çalışma voltajı: 3.6 ~ 5.5 V DC
- Ortalama akım: <18 mA
- Arayüz seviyesi: 3,3 V
- Ölçüm aralığı: 0 ~% 0,5 VOL isteğe bağlı
- Çıkış sinyali: UART - PWM
- Ön ısıtma süresi: 3 dakika
- Yanıt Süresi: T90 < 60 s
- Çalışma sıcaklığı: 0 ~ 50 ° C
- Çalışma nemi: 0 ~ 95% RH (Yoğunlaşma yok)
- Boyut: 33 mm × 20 mm × 9 mm (L × G × Y)
- Ağırlık: 21 g
- Yüksek hassasiyet, yüksek çözünürlük
- Düşük güç tüketimi
- Kararlı ölçümleme

Şekil - 5

1.6 – MQ-2 Yanıcı Gaz ve Duman Sensörü

MQ-2 gaz sensörü ortamdaki yanıcı ve sigara dumanını algılayabilen bir gaz sensörüdür. Ortamdaki gaz yoğunluğuna göre bir analog çıkış verir. Duman algılama aralığı 300 – 10000 ppm'dir.

Sensör kullanımı kolaylaştırmak adına bir kart üzerine monte edilip modül haline getirilmiştir. 5V ile beslenir. Sensörden hem dijital hem de analog çıkış alınabilir. Böylece hem gaz var-yok şeklinde yada gazın yoğunluğu tespitleri yapılabilir. Modül arkasında bulunan pot ile hassasiyet ayarı yapılabilir. Arduino, Raspberry Pi yada diğer mikrokontrolcülerle rahatlıkla kullanılabilir. Raspberry Pi ile kullanmak istediğinizde eğer analog pini kullanmak isterseniz arada Mcp3008 gibi bir çevirici kullanmanız gerekir.

Şekil -6

1.7 – MQ – 7 CO Gaz Sensörü

MQ-7 Karbonmonoksit gazı sensörü 10ppm ve 10.000ppm konsanstrasyonlarda Karbonmonoksit algılar. Diğer MQ sensörler gibi bu sensör de çıkış olarak gazın yoğunluğuna göre analog voltaj çıkışı verir. 10.000ppm ve 300ppm aralığında algılama yapabilmek gaz kaçağı için uygundur.

Özellikleri:

- Çalışma Sıcaklığı : -10°-50°C
- Çalışma Voltajı : 5V
- Çektiği Akım : 150 mA

Şekil – 7

1.8 – MQ-135

Çalışma voltajı: 5V DC

Özellikleri:

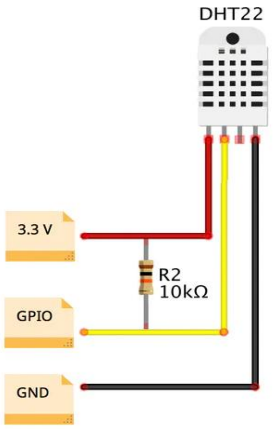
- Mikroprosesör uyumlu TTL çıkışı.
- Analog çıkış. Çıkış voltajı, havadaki gaz konsantrasyonuna orantısal olarak değişir.
- Yüksek hassasiyet. Sülfür, benzen, su buharı, duman ve diğer zararlı gazların(NH3, NOx, Alkol, CO2 vb) konsantrasyonunu hassas bir şekilde ölcer.
- Uzun bir çalışma ömrü ve kararlılığa sahiptir.
- Hızlı cevap süresi.
- Soketli problarla test için tak/çıkar kolaylığı**.**

Şekil – 8

2. ELEKTRONİK ELEMANLARIN ÇIKTI VE BAĞLANTILARI

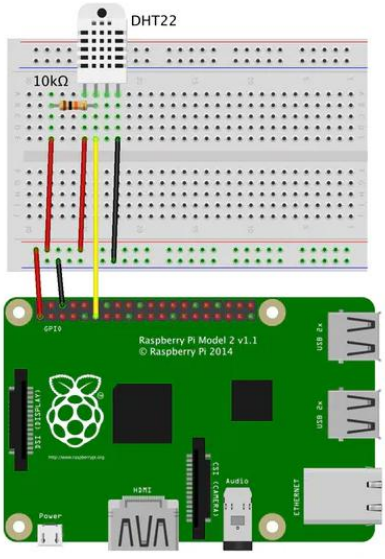
2.1 DHT-22

2.1.1-DHT-22 OUTPUT



Şekil -9

2.1.2- DHT-22 RASPBERRY PI BAĞLANTISI

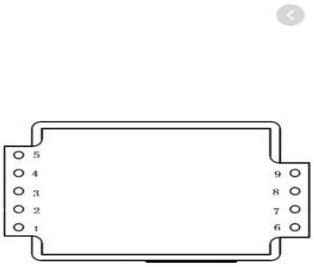


Şekil – 10

2.2 - MH-Z19

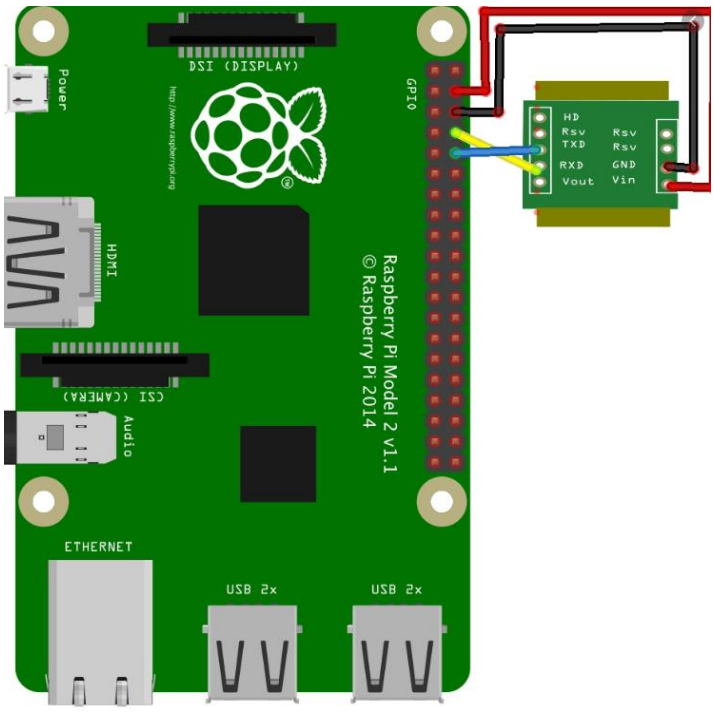
2.2.2-MH-Z19 OUTPUT

PIN	Description
Pin 6	Vin (voltage input)
Pin 7	GND
Pin 1	Vout (output voltage 3.3V, output current lower than 10mA)
Pin 9	PWM
Pin 5	HD (zero calibration, low level above 7 seconds) (Factory Reserved)
Pin 2	UART (RXD) 0~3.3V digital input
Pin 3	UART (TXD) 0~3.3V digital output
Pin 4	SR (Factory Reserved)
Pin 8	AOT (Factory Reserved)



Şekil – 11

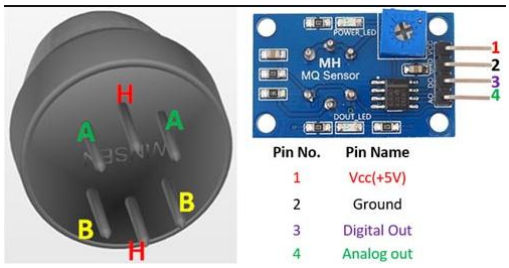
2.2.2- MH-Z19 RASPBERRY PI BAĞLANTISI



Şekil – 12

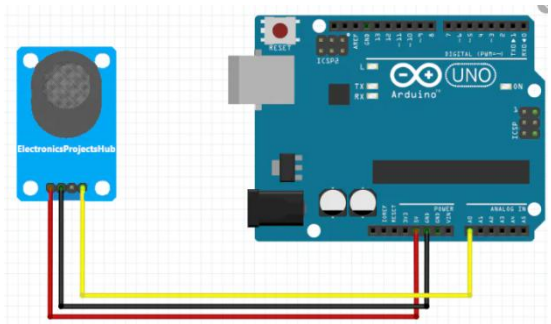
2.3 MQ-2, MQ-7, MQ-135

2.3.1 MQ-2, MQ-7, MQ-135 OUTPUT



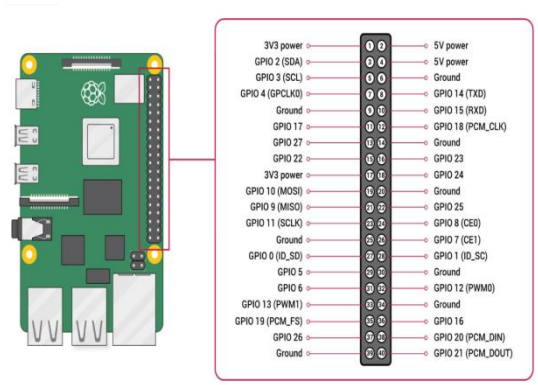
Şekil-13

2.3.2- MQ-2, MQ-7, MQ-135 ARDUİNO UNO BAĞLANTISI



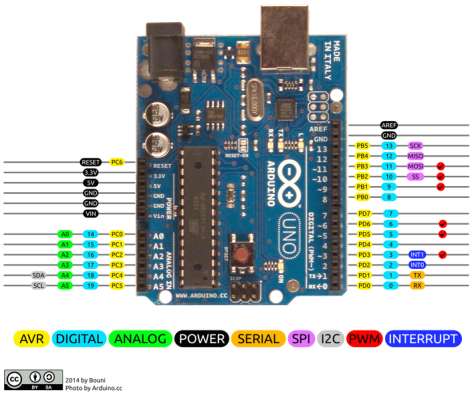
Şekil-14

2.4 RASPBERRY PI-4 OUTPUT



Şekil – 15

2.5- ARDUINO UNO OUTPUT



Şekil-16

3. YAZILIM

3.1 Arduino IDE

Arduino IDE, arduino kitleri için geliştirdiği; komutların yazılmasına, derleme işleminin yapılmasına ve son olarak da derlenen kodları doğrudan arduino kitine yüklenmesine olanak sağlayan yazılım geliştirme platformudur.

3.1.1 MQ2 Kodu

```
#define         MQ_PIN                      (0)     //define which analog input channel you are going to use
#define         RL_VALUE                    (5)     //define the load resistance on the board, in kilo ohms
#define         RO_CLEAN_AIR_FACTOR         (9.83)  //RO_CLEAR_AIR_FACTOR=(Sensor resistance in clean air)/RO,
                                                    //which is derived from the chart in datasheet

/***********************Software Related Macros************************************/
#define         CALIBARAION_SAMPLE_TIMES    (50)    //define how many samples you are going to take in the calibration phase
#define         CALIBRATION_SAMPLE_INTERVAL (500)   //define the time interal(in milisecond) between each samples in the
                                                    //cablibration phase
#define         READ_SAMPLE_INTERVAL        (50)    //define how many samples you are going to take in normal operation
#define         READ_SAMPLE_TIMES           (5)     //define the time interal(in milisecond) between each samples in
#include <LiquidCrystal.h>

const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);                                          //normal operation

/***********************Application Related Macros*********************************/
#define         GAS_LPG                     (0)
#define         GAS_CO                      (1)
#define         GAS_SMOKE                   (2)

/*************************Globals***********************************************/
float           LPGCurve[3]  =  {2.3,0.21,-0.47};   //two points are taken from the curve.
                                                    //with these two points, a line is formed which is "approximately equivalent"
                                                    //to the original curve.
                                                    //data format:{ x, y, slope}; point1: (lg200, 0.21), point2: (lg10000, -0.59)
float           COCurve[3]  =  {2.3,0.72,-0.34};    //two points are taken from the curve.
                                                    //with these two points, a line is formed which is "approximately equivalent"
                                                    //to the original curve.
                                                    //data format:{ x, y, slope}; point1: (lg200, 0.72), point2: (lg10000,  0.15)
float           SmokeCurve[3] ={2.3,0.53,-0.44};    //two points are taken from the curve.
                                                    //with these two points, a line is formed which is "approximately equivalent"
                                                    //to the original curve.
                                                    //data format:{ x, y, slope}; point1: (lg200, 0.53), point2: (lg10000,  -0.22)
float           Ro           =  10;                 //Ro is initialized to 10 kilo ohms

 void setup()
{
  Serial.begin(9600);                               //UART setup, baudrate = 9600bps
  Serial.print("Calibrating...\n");
  Ro = MQCalibration(MQ_PIN);                       //Calibrating the sensor. Please make sure the sensor is in clean air
   lcd.begin(16, 2);                                            //when you perform the calibration
  Serial.print("Calibration is done...\n");
  Serial.print("Ro=");
  Serial.print(Ro);
  Serial.print("kohm");
  Serial.print("\n");
 lcd.print("Calibration is done...\n");
 lcd.print("Ro=");
 lcd.print(Ro);
 lcd.print("kohm");
 lcd.print("\n");
 }
```

```
  void loop()
  {
     Serial.print("LPG:");
     Serial.print(MQGetGasPercentage(MQRead(MQ_PIN)/Ro,GAS_LPG) );
     Serial.print( "ppm" );
     Serial.print("    ");
     Serial.print("CO:");
     Serial.print(MQGetGasPercentage(MQRead(MQ_PIN)/Ro,GAS_CO) );
     Serial.print( "ppm" );
     Serial.print("    ");
     Serial.print("SMOKE:");
     Serial.print(MQGetGasPercentage(MQRead(MQ_PIN)/Ro,GAS_SMOKE) );
     Serial.pr
  int( "ppm" );
     Serial.print("\n");
         lcd.setCursor(0, 0);
         lcd.print("LPG:");
  lcd.print(MQGetGasPercentage(MQRead(MQ_PIN)/Ro,GAS_LPG) );
  //lcd.print( "ppm" );
  lcd.print("     ");
    lcd.setCursor(9, 0);
  lcd.print("CO:");
  lcd.print(MQGetGasPercentage(MQRead(MQ_PIN)/Ro,GAS_CO) );
  //lcd.print( "ppm" );
  lcd.print("       ");
   lcd.setCursor(0, 1);
  lcd.print("SMOKE:");
  lcd.print(MQGetGasPercentage(MQRead(MQ_PIN)/Ro,GAS_SMOKE) );
  //lcd.print( "ppm" );
  lcd.print("          ");
     delay(200);
  }
/*************** MQResistanceCalculation ********************************
Input:   raw_adc - raw value read from adc, which represents the voltage
Output:  the calculated sensor resistance
Remarks: The sensor and the load resistor forms a voltage divider. Given the voltage
         across the load resistor and its resistance, the resistance of the sensor
         could be derived.
************************************************************************************/
float MQResistanceCalculation(int raw_adc)
{
  return ( ((float)RL_VALUE*(1023-raw_adc)/raw_adc));
}

/************************* MQCalibration ********************************
Input:   mq_pin - analog channel
Output:  Ro of the sensor
Remarks: This function assumes that the sensor is in clean air. It use
         MQResistanceCalculation to calculates the sensor resistance in clean air
         and then divides it with RO_CLEAN_AIR_FACTOR. RO_CLEAN_AIR_FACTOR is about
         10, which differs slightly between different sensors.
************************************************************************************/
float MQCalibration(int mq_pin)
{
  int i;
  float val=0;

  for (i=0;i<CALIBARAION_SAMPLE_TIMES;i++) {            //take multiple samples
    val += MQResistanceCalculation(analogRead(mq_pin));
    delay(CALIBRATION_SAMPLE_INTERVAL);
  }
  val = val/CALIBARAION_SAMPLE_TIMES;                   //calculate the average value

  val = val/RO_CLEAN_AIR_FACTOR;                        //divided by RO_CLEAN_AIR_FACTOR yields the Ro
                                                        //according to the chart in the datasheet

  return val;
}
```

```
/*************************  MQRead *************************************
Input:   mq_pin - analog channel
Output:  Rs of the sensor
Remarks: This function use MQResistanceCalculation to caculate the sensor resistenc (Rs).
         The Rs changes as the sensor is in the different consentration of the target
         gas. The sample times and the time interval between samples could be configured
         by changing the definition of the macros.
************************************************************************************/
float MQRead(int mq_pin)
{
  int i;
  float rs=0;

  for (i=0;i<READ_SAMPLE_TIMES;i++) {
    rs += MQResistanceCalculation(analogRead(mq_pin));
    delay(READ_SAMPLE_INTERVAL);
  }

  rs = rs/READ_SAMPLE_TIMES;

  return rs;
}
/*************************  MQGetGasPercentage **************************
Input:   rs_ro_ratio - Rs divided by Ro
         gas_id      - target gas type
Output:  ppm of the target gas
Remarks: This function passes different curves to the MQGetPercentage function which
         calculates the ppm (parts per million) of the target gas.
************************************************************************************/
int MQGetGasPercentage(float rs_ro_ratio, int gas_id)
{
  if ( gas_id == GAS_LPG ) {
     return MQGetPercentage(rs_ro_ratio,LPGCurve);
  } else if ( gas_id == GAS_CO ) {
     return MQGetPercentage(rs_ro_ratio,COCurve);
  } else if ( gas_id == GAS_SMOKE ) {
     return MQGetPercentage(rs_ro_ratio,SmokeCurve);
  }

  return 0;
}

/*************************  MQGetPercentage ****************************
Input:   rs_ro_ratio - Rs divided by Ro
         pcurve      - pointer to the curve of the target gas
Output:  ppm of the target gas
Remarks: By using the slope and a point of the line. The x(logarithmic value of ppm)
         of the line could be derived if y(rs_ro_ratio) is provided. As it is a
logarithmic coordinate, power of 10 is used to convert the result to non-logarithmic
         value.
************************************************************************************/
int  MQGetPercentage(float rs_ro_ratio, float *pcurve)
{
  return (pow(10,( ((log(rs_ro_ratio)-pcurve[1])/pcurve[2]) + pcurve[0])));
}
```

3.2 Python Modülleri

3.2.1 MH-Z19

Python mh-z19 kütüphanesi kullanıldı. Kütüphaneyi indirmek için aşağıdaki komut satırını terminalde çalıştırmak yeterlidir.

- pip install mh-z19

3.2.2 DHT-22

Python'un Adafruit_DHT kütüphanesi ile digital pinden sıcaklık ve nem değerleri okundu. Kütüphaneyi indirmek için aşağıdaki komut satırını terminalde çalıştırmak yeterlidir.

- pip install Adafruit_DHT

```
#!/usr/bin/python

#pre requirement
#pip3 install Adafruit_DHT
import Adafruit_DHT

def read_dht(pin=4, sensor = Adafruit_DHT.DHT22):


    while(True):
        humidity, temperature = Adafruit_DHT.read_retry(sensor, pin)
        if humidity is not None and temperature is not None:
            print('Temp={0:0.1f}*C  Humidity={1:0.1f}%'.format(temperature, humidity))
        else:
            print('Failed to get reading. Try again!')



###Test###
"""
# Sensor should be set to Adafruit_DHT.DHT11,
# Adafruit_DHT.DHT22, or Adafruit_DHT.AM2302.
sensor = Adafruit_DHT.DHT22

# Example using a Beaglebone Black with DHT sensor
# connected to pin P8_11.
pin = '4'

read_dht()
"""
###Test####
```

4.ARDUINO İLE RASPBERRY PI HABERLEŞMESİ

Proje kapsamında anolog çıkış veren sensör verilerini okumak için adc dönüştürücüye ihtiyaç duyuldu. Raspberry'nin sadece digital input pinleri içermesi bu sensörlerin raspberry pi ile kullanılmasını engelleyen bir hal aldı. Bunun üzerine elimizde bulunan arduino uno ile analog output veren sensör verilerinin okunması kararı alındı. Arduino ile okunan veriler USB üzerinden seri haberleşme ile Raspberry PI' a iletildi. Böylelikle elde edilen bütün sensör verileri tek bir yerde toplanmış oldu ve bu verilerin bulut sistemine iletilmesi kolaylaştı. Seri haberleşmeyi gerçekleştirebilmek için Arduinda Serial kütüphanesini kullanıldı. Raspberry PI üzerinde yine serial kütüphanesi ile aşağıdaki modül oluşturuldu. Bu modül aracılığı ile arduinodan gönderilen bilgiler haberleşme kanalı üzerinde yakalandı ve kaydedildi.

```
import serial
import time

class Cummunication:

    def __init__(self):
        self.portpath = "/dev/ttyACM0"
        self.boudrate = 9600
        self.timeout = 1
        self. stopbits = 1.5

    def read_message(self):
        port = serial.Serial(self.portpath, self.boudrate, \
                             timeout = self.timeout, stopbits =
self.stopbits)
        start = time.time()
        while(True):
            end = time.time()
            timer = end-start
            if timer >= 00:
                print(timer)
                ValueStr = port.readline()
                print(ValueStr)
                start = time.time()

com = Cummunication()
com.read_message()
```

5. BULUT SİSTEMİ

Veritabanı ihtiyacımız olan veya daha sonraları ihtiyacımız olacak bütün verileri depoladığımız sistemdir. Günümüz uygulamalarının(web,mobil,masaüstü) neredeyse tamamında yerel veya uzak sunucuya bağlantılı veritabanları bulunur. Veritabanlarına depoladığımız bu bilgileri daha sonraları kullanabilir, üzerlerine düzenlemeler yapabilir veya bu verileri anlamlandırarak bilgiler elde edebiliriz.

5.1-NoSQL NEDİR?

Son yıllarda verinin inanılmaz boyutlara ulaşması ve katlanarak artması sonucunda mevcut olarak kullanılan ilişkisel veritabanı sistemleri yerine ortaya atılmış bir kavramdır. İlişkili veritabanı sistemleri ile arasındaki en büyük fark ilişkisel veritabanı sistemlerinde veriler tablo ve sütunlar ile ilişkili bir şekilde tutulurken NoSQL'de JSON bir yapıda tutulmasıdır.

NoSQL sistemlerin avantajlarına değinmek gerekirse ilk olarak performans gösterilebilir. Okuma ve yazma işlemleri ilişkisel veritabanlarına göre çok daha hızlı olmaktadır. İkinci olarak ise NoSQL sistemler yatay olarak genişletilebilirler. Binlerce sunucu bir arada çalışarak inanılmaz derecedeki veriler üzerinde işlemler yapabilir.

5.2-MongoDB NEDİR ?

MongoDB 2009 yılında geliştirilmiş açık kaynak kodlu bir NoSQL veritabanıdır. Günümüzde aktif olarak kullanılan pek çok programlama dili için driver desteği bulunması bakımından bugün NoSQL sistemler içerisinde en çok tercih edilenlerden biridir

5.3- BULUT SİSTEMİ İLE RASPBERRY PI BAĞLANTISI

Öncelikle Arduino ve Raspberry PI ile sensörlerden alınan verilerin bir araya toplanması gerekiyordu. Python üzerinde bütün verileri bir yere toplayıp JSON formatında kaydeden bir modül yazıldı. Daha sonra MongoDB'nin python API'ı ile geliştirilen başka bir modül ile oluşturulan JSON formatında kaydedilmiş olan veriler düzenli aralıklarla bulut sistemine kaydedildi.

6- BULUT SİSTEMİNDEKİ VERİLERE ERİŞİM VE DEĞERLENDİRME ü

MongoDB ile bulut sistemine yüklenen veriler, daha önceden oluşturulan üyelik ve şifrelerle erişelebilir hale getirildi. Daha sonra Python ile bu verilerin okunması ve değerlendirmesini gerçekleştiren bir modül oluşturuldu. Böylelikle proje başında amaçladığımız bir kit üzerinden alınan verileri merkezi bilgisayarlara iletme işlemi başarıldı.

7.SONUÇ

İki ayrı aşama ele alınan projenin ilk kısmı olan bu projede gerekli sensörlerin bulunması, gerekli işlemci ve mikrodenetleyicilerin kullanımı üzerine araştırmaların yapılması ve elde edilen bilgiler doğrultusunda proje için en uygun ekipmanların kullanımının tamamlanması ile ikinci aşamada bu sensörlerden gelen verileri bir dosyada toplayıp bunları bulut sistemine aktarma amaçlanmıştır. Dünya genelinde kargo süreçlerini ve ekipman temini konusundaki sıkıntılar nedeniyle yurtdışından temin edilmesi planlanan, hassasiyeti daha yüksek sensörlerin kullanımı gerçekleştirilemese de var olan sensörler ile ilk aşama tamamlandırılmıştır. İkinci aşama olan bulut sisteminin oluşturulması verilerin bu sisteme kaydedilmesi ve farklı cihazlardan bu bilgilere erişim işlemi tamamlanmıştır. Böylelikle 5V bir adaptör ile çalışabilen ve farklı noktalardan takip edilebilen, düşük enerji tüketime sahip, uygun fiyatlı hava kalite kontrol sistemi tamamlanmıştır