Kenith.R
IBM18CS045
IoT
5A.

① Advantages of WIFI:

- Convenience – The wireless nature of such networks allows users to access network resources from nearly any convenient location within their primary networking environment.

- Mobility – With the emergence of public wireless networks, users can access the internet even outside their normal work environment. Most coffee shops, for example offer their customers a wireless connection to the internet at little or no cost.

- Productivity – Users connected to a wireless network can maintain a nearly constant affiliation with their desired network as they move from place to place.

- Expandability – Wireless networks can serve a sudden increased number of clients with the existing equipment. In a wired network, additional clients would require additional wiring.

Keniah.R
1BM18CS045

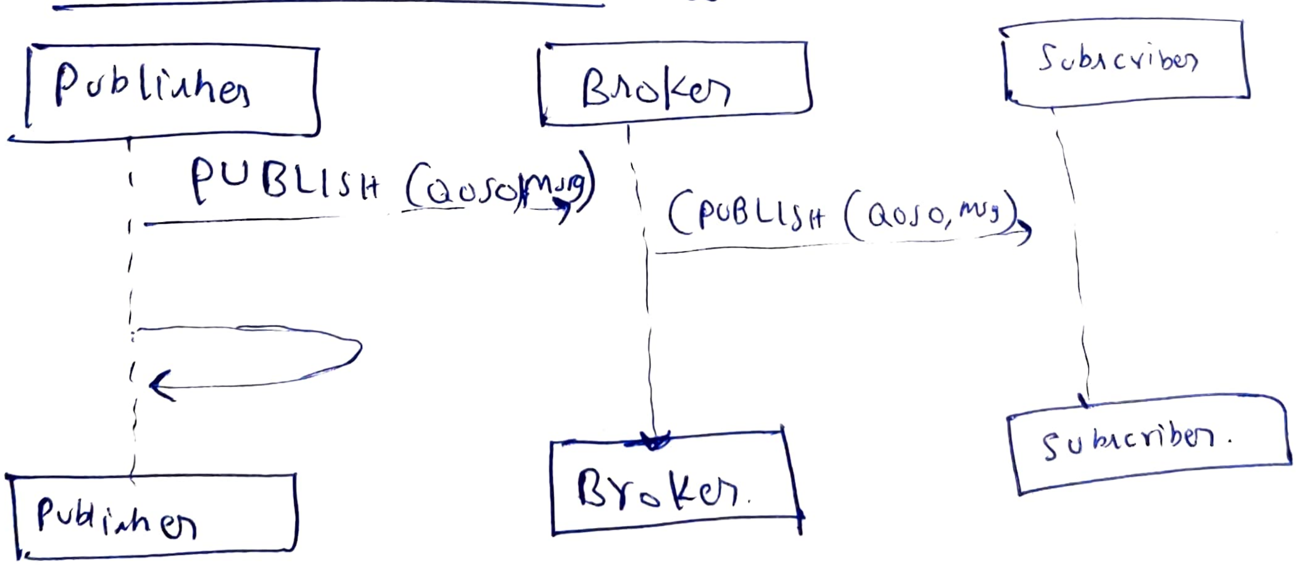
In IoT enterprise:

we may see many IoT application and use cases requiring high-throughput and low-latency wireless communication. Network communications, such as WIFI-6 gateways, CPE and devices requiring high throughput and an efficient management of connections.

② ⓑ Quality of Service (QoS) in MQTT messaging is an agreement between sender and receiver on guarantee of delivering a message.

There are three levels of QoS:

(i) QoS 0 – publish once at most:

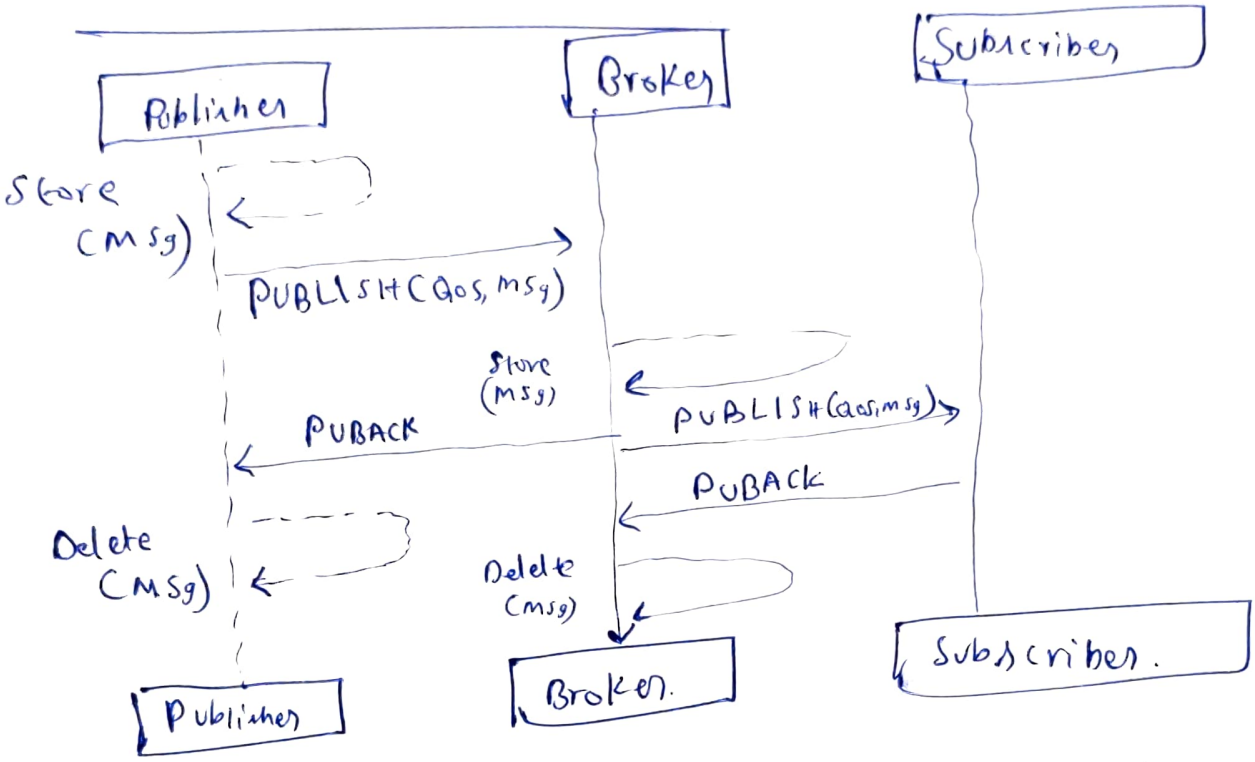


~~Publish of messages depends on the capabilities of the under~~

This is the simplest, lowest-overhead method of sending a message. The client simply publishes the message and there is No acknowledgement by broker.

Kenith R
1BM18CS045

QOS 1 – Publish once at least:



This method guarantees that the message will be transferred succefully to the broker. The broker sends an acknowledgment back to the sender, but in the event that the acknowledgement is lost the sender won't realise the message has got through, so will send the msg again.

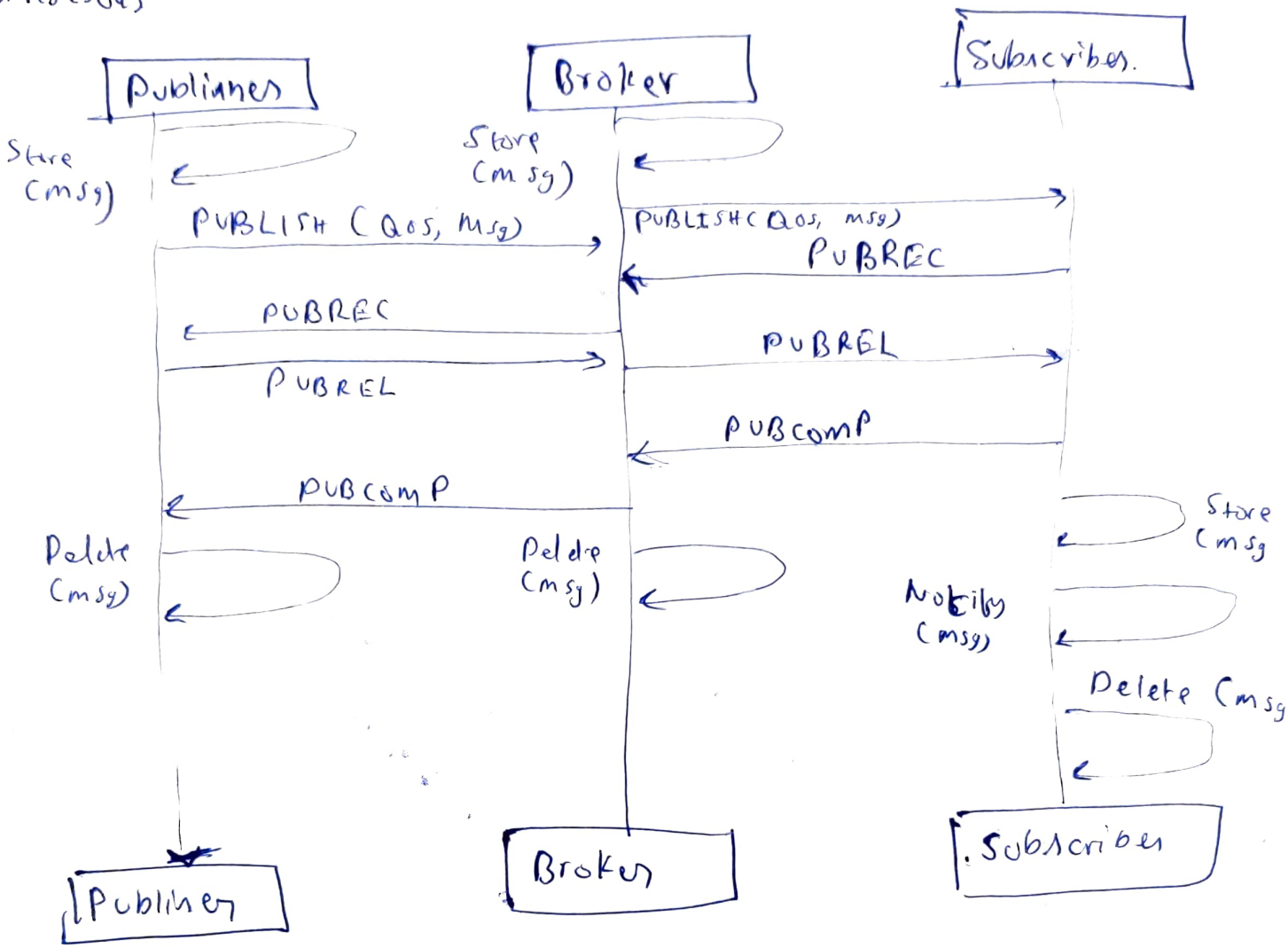
QOS 2 – Publish only once:

This is the highest level of service, in which there is a sequence of four message between the sender and receiver, a kind of handshake to confirm that the main message has been sent and the acknowledgment has been recieved.

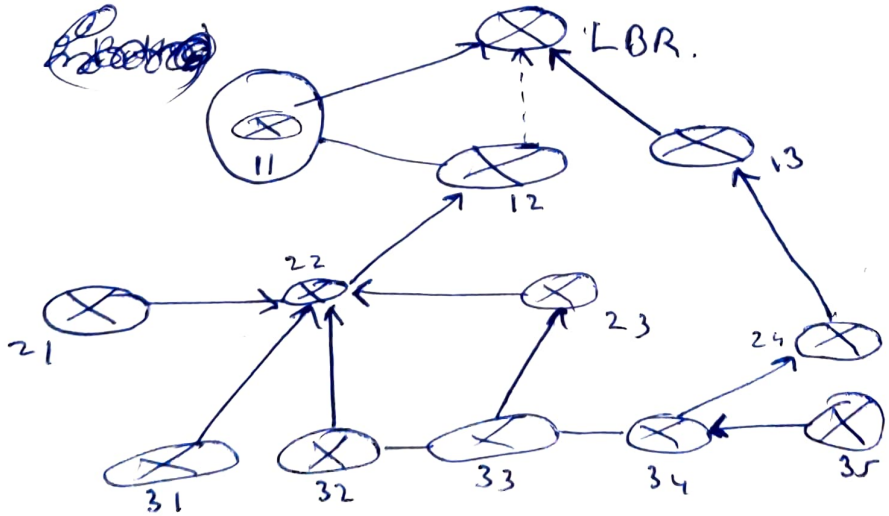
(P. T. O for Diagram).

Kenith.R

1BM18CS045

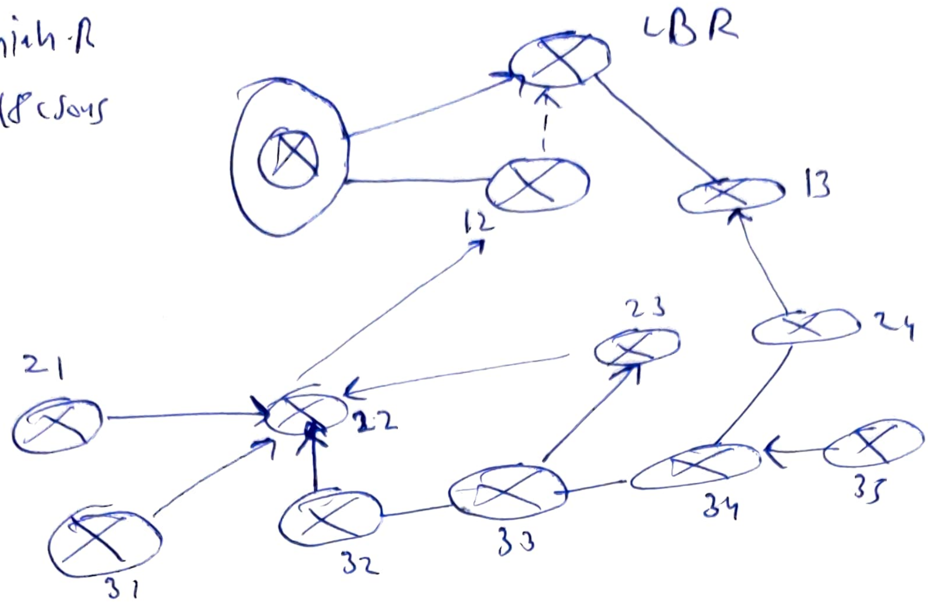


(2) @. The instances of DODAG are:



DAG Instance 1.

Kenish R
1BM18CS045



DAG Instance 2.

Note:

- - - - - : Poor quality

——— : Fair quality

——— : Good quality.

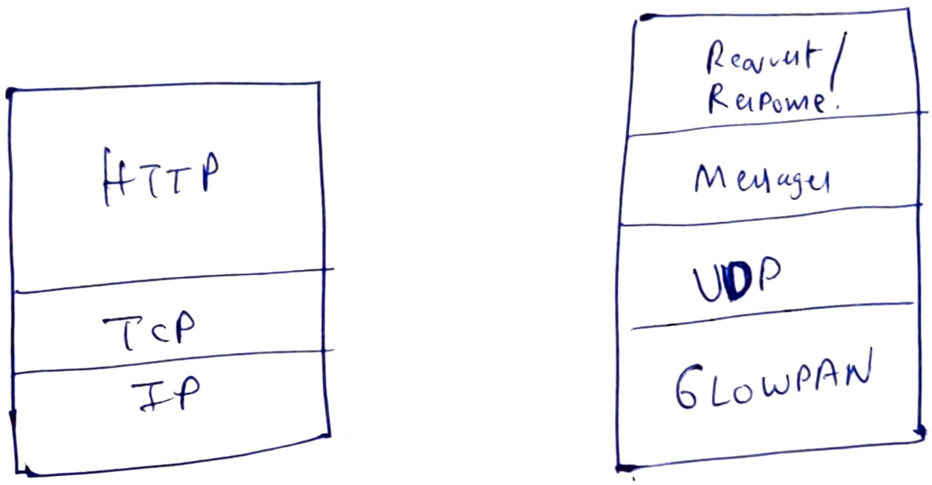
The Above shown diagram, DAG Instance 1 is High quality - no battery operated nodes. and DAG Instance 2 is Low latency.

~~Information Object (DIO)~~

(2) (c) unlike HTTP based Protocols, CoAP operates over UDP instead of using Complex congestion Control as in TCP. It is based on REST architecture. CoAP Provides URI, REST, Method such as GET, POST, PUT and DELETE. It also allows IP multicast, which satisfies group Communication for IOT.

Kanishk.R
1BM19CS045

To compensate the unreliability of UDP protocol, CoAP defines a retransmission mechanism. The diagram below shows the HTTP and CoAP Protocol Stack.



HTTP Protocol | CoAP Protocol

CoAP is not just a simply compression of HTTP protocol. Considering low processing capabilities and low power consumption demand of restrained resource, CoAP redesigned some features of HTTP to accomodate these limitations. CoAP used under constrained network.

Keniah.R

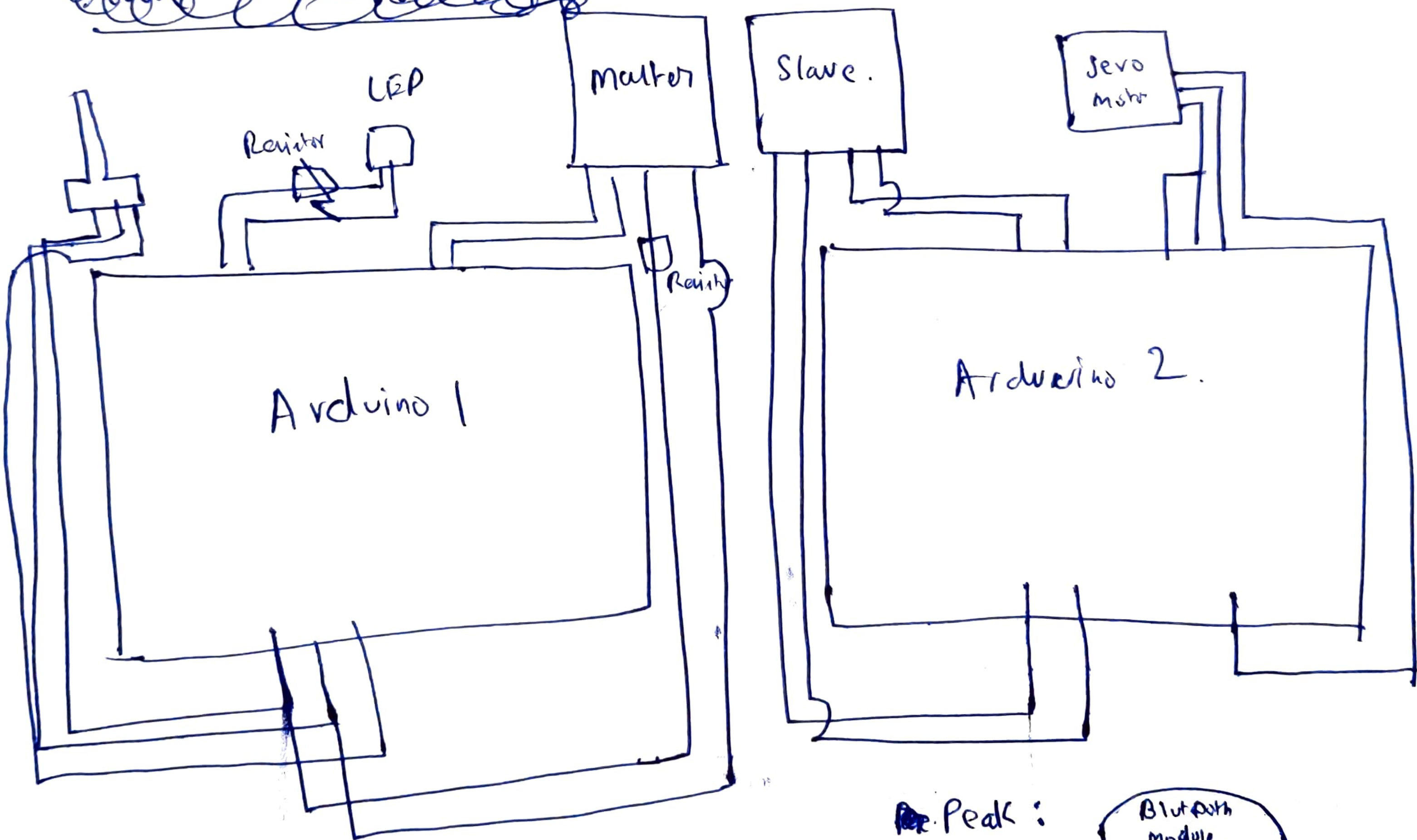
1BM18CS045.

(3)(a). Hardware req :

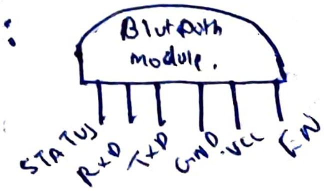
- HC-05 Bluetooth module
- Arduino Board
- Servo motor
- Potentiometer
- 3x 220 Ohm resistors
- Bread Board and jump Wires.

This is a master - slave Configuration:

~~Slave Configuration~~ Diagram:



Peak :



Kanishk
[illegible]

Master Code:

```
#define ledPin 9
int state = 0;
int PotValue = 0;

Void setup() {
    pinMode (ledPin, OUTPUT)
    digitalWrite (ledPin, Low),
    Serial.begin (38400);
}

Void loop() {
  if (Serial.available() >0){
      State = Serial.read();
  }
  if (State == '1') {
     digital write( ledPin, HIGH),
     State = 0;
  }
  else if (State == '0'){
     digital Write (ledPin, Low),
     State = 0;
  }
PotValue = analogRead (A0)
 int PotValueMap = map (PotValue,
                        0, 1023, 0, 255);
Serial.Write (PotValueMAP);
 delay (10);
}
```

Slave Code:

```
#include <servo.h>
#define buttonPin 8
Servo myServo;
int State = 20;
int buttonState = 0;

Void setup () {  pinMode (button,
                          INPUT);

 myServo.attach (9);
 Serial.begin (38400);
Void loop () {
 if ( Serial.available () >0) {
      State = Serial.read();
 }
myServo.write (State);
  delay (10);

buttonState = digitalRead (button);
  if (buttonState == HIGH)
  { Serial.write ('1');
  }
else { Serial.write ('0'); }
}
```

Jenish. R
IBM Ideas

Detailed Description:

first we need to define the pins and some variables needed for the Program. In the setup section, at the master, we set the LED PIN. as output and set it low right away, as well as, start the serial Communication at 38400 baud rate. Similarly, at the slave, we set the button pin as input, define the servo to which pin it's connected and start the serial Communication with the same baud rate.

(4) (6) Components required:

1. Arduino uno
2. GSM module
3. bread board
4. gas sensor
5. sim Card
6. wires

PIN Configuration:

GSM Tx to Arduino Pin 2
GSM Rx to Arduino Pin 3
Gas sensor output A0.

Kenith.R
IBM Cloud

Code:

```
#include <SoftwareSerial.h>
SoftwareSerial Cell (2,3);
int threshold = 200;
Void setup () {
  Cell.begin (9600);
delay (500);
Serial.begin (9600);
}
Void loop () {
float Sen_val = analogRead (A0);
 if (Sen_val > threshold ) {
    Serial.Println ("calling");
    Cell.Println ("ATD + 91 8934500789;");
     delay (10000);
    Cell.Println ("ATH");

  Serial.Print ("Sending SMS");
   Cell.Println ("AT+CMGF=1");
   delay (1000);
   Cell.Println ("AT+CMGF =\"+918934500789\"");
   delay (500);
   Cell.Println ("GAS ALERT");
   Cell.Write ((char)26);
    delay (1000);
    }
  else { Serial.Println ("NO Gas Leakage"); }
    }
```