Project B-Eagle : IOT-based gas leakage detection system

VSS Anirudh Sharma, Devadas Ramadarsan, Anugyan Das, Harsh Shrivastava

Abstract— As gas is going to become an important source of power as the world moves towards a cleaner future, more and more processes & systems will be powered by CNG, LPG, methane and even hydrogen. Along with this, a variety of gases are already in use in the industrial & production sector. As these gases are mostly hazardous or inflammable, in this paper we have suggested an IOT-based gas leakage detection system which monitors concentration of some target gas in a space, finds the source and reports it to prevent accidents. We have done a thorough analysis of the problem, and suggested multiple solutions for different cases & different scales.

1 Introduction

A gas leak refers to an unintended leak of natural gas or another gaseous product from a pipeline or other containment into any area where the gas should not be present. Gas leaks can be hazardous to health as well as the environment. While residential leaks pose risks to human safety, leakage in every piece of the natural gas industrial chain – wellheads, compressors, valves, pumps, gauges and pipe connectors – has serious implications for the climate. Oil and gas companies want to help reduce the quantity of lost gas due to the associated costs and liabilities. Global fugitive methane costs over $30 billion in lost revenue per year, according to a 2012 study. And just 5 percent of leaks in the production and transport system are responsible for more than half the methane emissions. Gas leak sources need to be detected accurately to control disasters. We are solving the problem of accurately locating sources of gas leaks even in environments inaccessible to humans.[4]

2 Objective

We propose a leak source search system mounted on small sized drones or rovers that will accurately reach the source of the gas leak or fire. Subsequently, the drone/rover might act as a beacon locating the source exactly or could manage the hazard if relevantly equipped. Here, we try:

- To create a Gas Leak Source Search System.
- To equip a drone with this system and make it reach the source.
- This drone would act like a beacon, informing the user about the exact location of the source.

3 Key Idea

3.1 Hypothesis

We hypothesize that the intensity of gas peaks at its source. This makes the problem of source detection into peak detection

3.2 Proposed Method

Thus, if the intensity of gas ppm be considered the cost function, we can perform gradient 'ascent' and reach the source.d reach the source.

3.3 The Algorithm

With the intensity of gas (ppm -> digital value) being considered the cost function, the objective of the algorithm is to maximize this cost function. The algorithm is as shown in Alg 1:

Algorithm 1: Gradient Ascent

$\alpha := Learning\ rate,\ \Delta = Distance\ of\ centre\ to\ sensor$

$for\ i = 1, 2, 3 \ldots$

$$I_x = \frac{\delta I}{\delta x};\ I_y = \frac{\delta I}{\delta y};\ I_z = \frac{\delta I}{\delta z}$$

$$x_{k+1} = x_k + \alpha I_x; y_{k+1} = y_k + \alpha I_y; z_{k+1} = z_k + \alpha I_z;$$

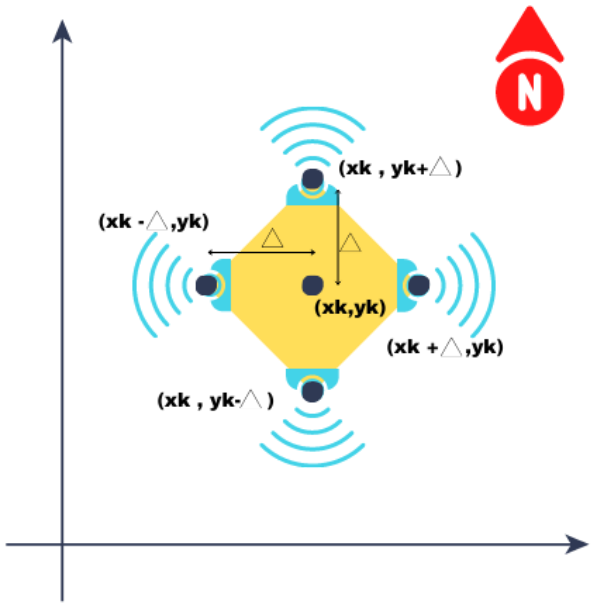
3.4 How to find gradients?

We don't know the function of the distribution of I. The next best way is to find an approximation to the gradients based on some sensor readings. To achieve this, we propose a sensor system as in fig 1

fig 1: Sensor System

6 gas sensors must be placed on the left, right, front, back, up and down of the frame. All equidistant from the center of the frame. Placing this frame in a coordinate system, we get coordinates of the sensors as shown in fig 2.

fig 2: Sensor coordinate system



We can now approximate the gradients I_x, I_y, I_z as follows:

$$\frac{\delta I}{\delta x} \approx \frac{I(x+\Delta,y,z) - I(x-\Delta,y,z)}{2\Delta}$$

$$\frac{\delta I}{\delta y} \approx \frac{I(x,y+\Delta,z) - I(x,y-\Delta,z)}{2\Delta}$$

$$\frac{\delta I}{\delta z} \approx \frac{I(x,y,z+\Delta) - I(x,y,z-\Delta)}{2\Delta}$$

To standardize the distance travelled by the system in every iteration of the algorithm, we propose a modified gradient ascent algorithm (Alg 2). This will help is define the step size to be constant and bigger than the Δ so that the gradient approximations closely mimic actual values and let define an intuitive and practical breaking condition.

Algorithm 2: Normalized Gradient Ascent

$\alpha := Step\ Size,\ \Delta = Distance\ of\ centre\ to\ sensor$

$for\ i = 1, 2, 3 \ldots$

$$I_x = \frac{\delta I}{\delta x};\ I_y = \frac{\delta I}{\delta y};\ I_z = \frac{\delta I}{\delta z}$$

$$R = \sqrt{I_x^{\,2} + I_y^{\,2} + I_z^{\,2}}$$

$$J_x = \frac{I_x}{R};\ J_y = \frac{I_y}{R};\ J_z = \frac{I_z}{R}$$

$$x_{k+1} = x_k + \alpha J_x;\ y_{k+1} = y_k + \alpha J_y;\ z_{k+1} = z_k + \alpha J_z;$$

The loop would break if

$$if\ \sqrt{(x_k - x_{k-2})^2 + (y_k - y_{k-2})^2 + (z_k - z_{k-2})^2} < \alpha$$

3.5 Simulations

We tried to simulate the alg 2 in 2 D for different gas distributions, static and flowing. A 400x400 grid of zeros was filled with values of 1000 inside circles with different radii at different centers. This grid is then blurred using gaussian blur to create an effect of distribution of gas. Gaussian blur has been chosen for 2 reasons:

1. The centralized nature of blurring, intuitively similar to gas leaks.
2. The sum total of intensities sums the same before and after the blur, ensuring conservation of particles as observed in real life.

In the flowing case, the rate of incrementing this initial intensity of 1000 in the circle is proportional to the radius, which appears to be a reasonable approach. The gaussian blur is run in each iteration after adding intensity to the circles to simulate flow.

Figures 3, 4 and 5 are simulated runs of static gas distributions. Figures 6 and 7 are simulations of flowing gas.

fig 3: Single point simulation : No flow

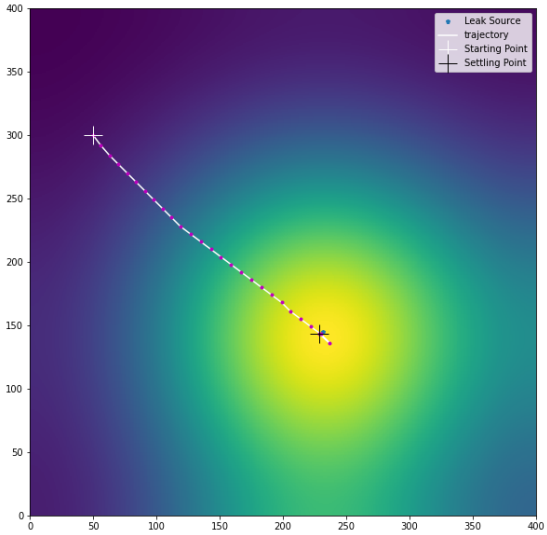


fig 4: 3 centered leak: No flow

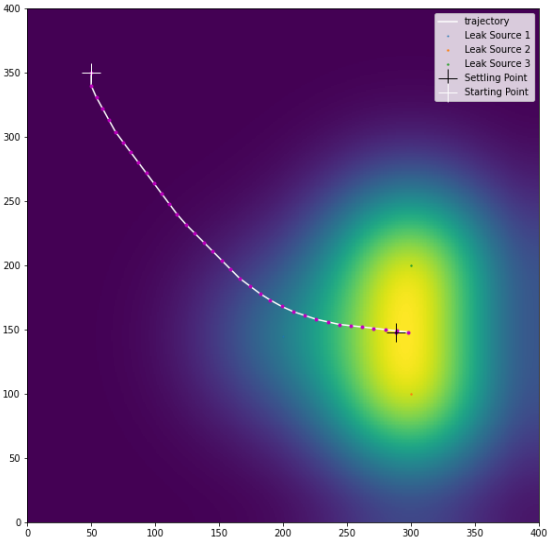


fig 5: 3 sources and 1 distant source: No flow

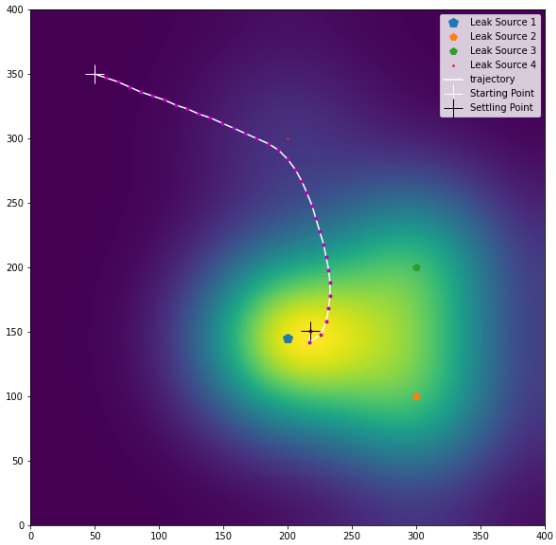


fig 6: single point source with flow

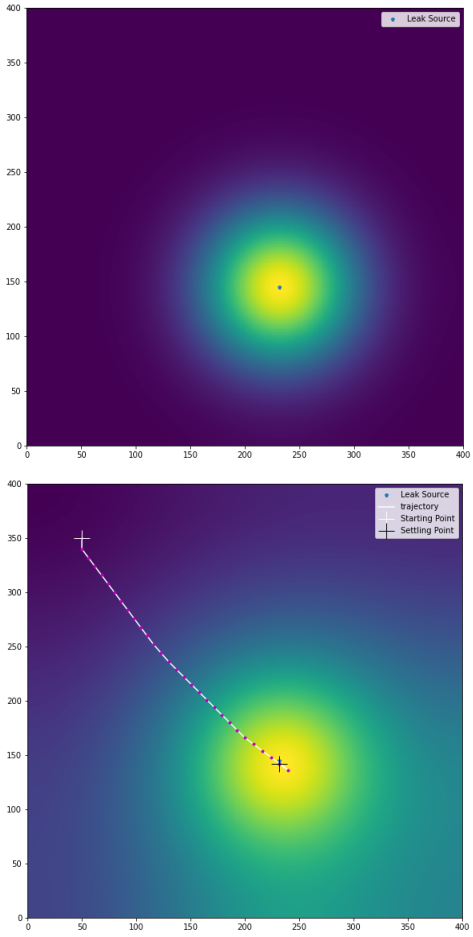


fig 7: 3 sources: With flow

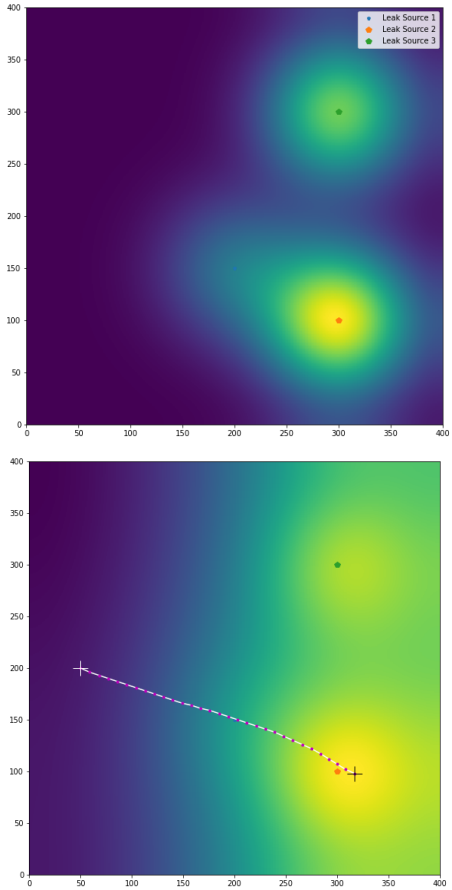
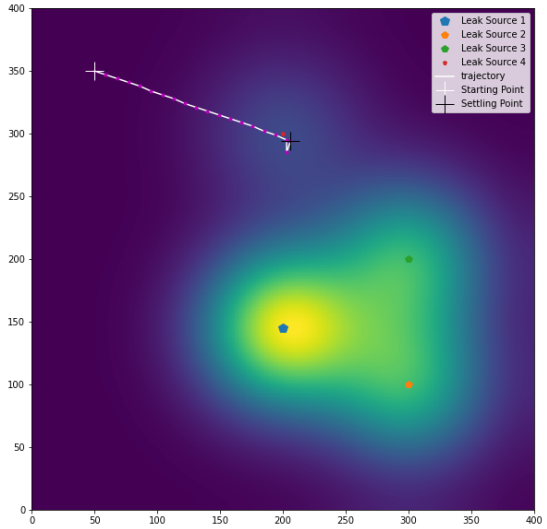


fig 8: 3 sources and 1 distant source (lesser blur): No flow



3.6 Observations

In all the single point source simulations, the system perfectly locates the source. In figures 5 and 7, where different sources have different rates and radii, the system settles at the highest peaks.

The important observation occurs in figures 5 and 8. Both the

leak locations and rates are exactly the same. The only difference is in the blur, which is a proxy for time. More the time the gases were left to mix up, more the blur in representation. In fig 5 the system settles at the global maxima (luckily) but in fig 8, the system settles at a local maxima. This gives us an important insight:

Use multiple drones all at different initial points to successfully detect all peaks.

3.7 Vision of execution

The system can be made physically mobile by mounting it on a drone or a rover. A drone would move in 3 dimensions and require 6 gas sensors (Up, Down, Right, Left, Front, Back) as in fig 9 and a rover would move in 2 dimensions only and would require only 4 gas sensors (Left, Right, Front, Back) as in fig 10.

fig 9: Drone Based Gas Leak Detection Concept

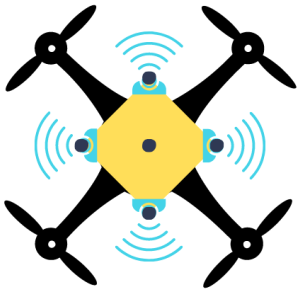
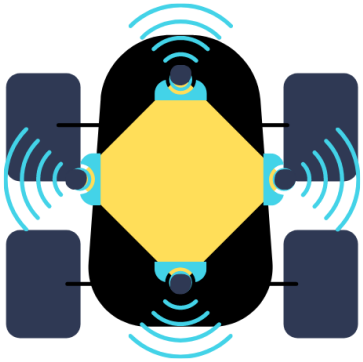


fig 10: Rover Based Gas Leak Detection concept



The drone can communicate its location to the control centre once it settles at the source so that relevant remedy can be deployed. This is achieved with the help of RF Rx and Tx in this project.

3 Evolution of the Idea

3.1 Project SafeHouse

Initially, the project was called Project Safe house, where a stationary gas detection system (fig 11) was achieved. This system is fixed in a room and connected to WiFi. On the gas intensity in the room crossing a threshold, the device would ring an alarm, send an alert to the users' smartphones and using a custom application made through Blynk, the users could read the gas levels of this room (fig 12). Here's the link to the Mark III demonstration.

fig 11: Project SafeHouse, Mark III

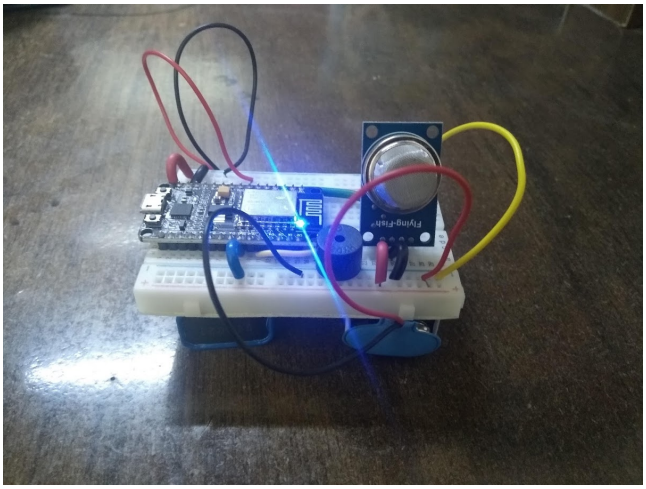


fig 12: Blynk Gas Leak Detector App

Fig 1: Normal

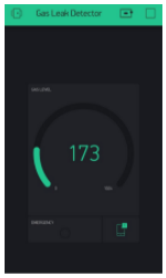


Fig 2: Gas Leak Immediate

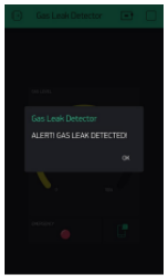


Fig 3: Alert after 5 seconds of leak detection

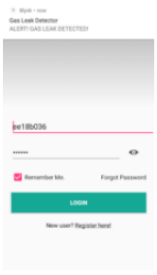


Fig 4: Alert notification when not on the surveillance app

3.2 Hand-held device

The user will manually hold the sensor frame and move about as directed by the modified gradient ascent computations. However there are few limitations to this hand-held method. The path calculated by the drone may not be feasible for the human to follow. Also,the handheld device will force to adjust the drone according to human mobility convenience and this may cause the drone to enter the region of zero gradient and hence the gradient ascent may fail.

4 Electronics and sensors

4.1 GAS Sensor : AS-MLV-P2

The ams AS-MLV-P2 is a MOS (metal oxide semiconductor) based gas sensor component. It was specifically designed for a broad detection of reducing gases such as VOCs (volatile organic compounds) and CO (carbon monoxide) associated with bad air quality. The AS-MLV-P2 sensor component is MEMS (micro electromechanical system) device using silicon wafer technology.

fig 13: MLV-P2

Specifications

- Power: 2.5V ~ 5.0V
- Dimension: 40.0mm * 21.0mm
- Mounting holes size: 2.0mm
- Applications: Gas leakage detector

Benefits

- Sensitive for LPG, natural gas, coal gas
- Output voltage boosts along with the concentration of the measured gases increases
- Fast response and recovery
- Adjustable sensitivity
- Signal output indicator

4.2 Flight controller : PIXHAWK 2.4.8

Pixhawk is a flight controller providing readily-available, low-cost, and high-end, autopilot hardware designs. It can be easily combined with a companion microcontroller (raspberry pi 3 in our case) to do the computation part for the path calculation.

Features

- Sensors: Gyrometer, Accelerometer, Barometer Magnetometer
- 6-10V supply voltage range supported.
- Micro SD card to record flight data

fig 14: PixHawk

4.3 Raspberry Pi 3 Model B+

We will do our flight trajectory calculations and send the control signals from raspberry pi to the flight controller.

Features

- 5V supply, 40-pin GPIO Header
- 2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN
- Bluetooth 4.2
- Four USB 2.0 ports

fig 15: Raspberry pi 3

4.4 LiPo 3000mAh 11.1V battery

Features

- Rechargeable Power Supply for RC Cars and Quadcopter
- 11.1 V, 40/80 C , 3000mAh
- Light weight, weighing only about 215grams
- Can supply upto 240A max current.

fig 16: LiPo Battery. 11.1V

4.5 Voltage regulator ICs

We need two voltage regulators, one for 5V output and another for 8V for raspberry pi and pixhawk respectively.
We will use 7805 IC and 7808 IC for the task.

4.6 433MHZ RF Transmitter Receiver Wireless Module With upto 100 meters Range

This hybrid RF Transceiver Module provides a complete RF transmitter and receiver module solution which can be used to transmit data at up to 3KHz from any standard CMOS/TTL source.

The transmitter module is very simple to operate and offers low current consumption (typical. 11mA). Data can be supplied directly from a microprocessor or encoding device, thus keeping the component count down and ensuring a low hardware cost.

Applications 433MHz RF Transmitter Receiver Wireless Module:-

- Remote Controls
- Automation System
- Wireless Security System
- Sensor Reporting
- Car Security System
- Remote Keyless Entry

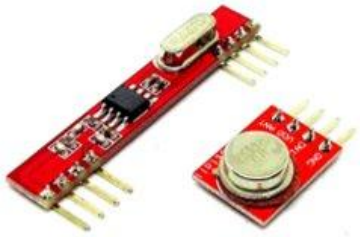
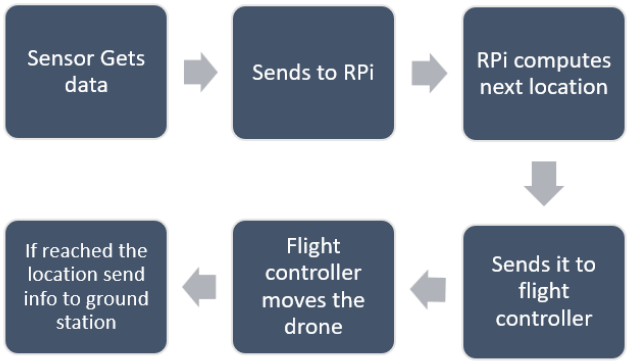


fig 17: RF 433 MHz Rx and Tx

5 The drone

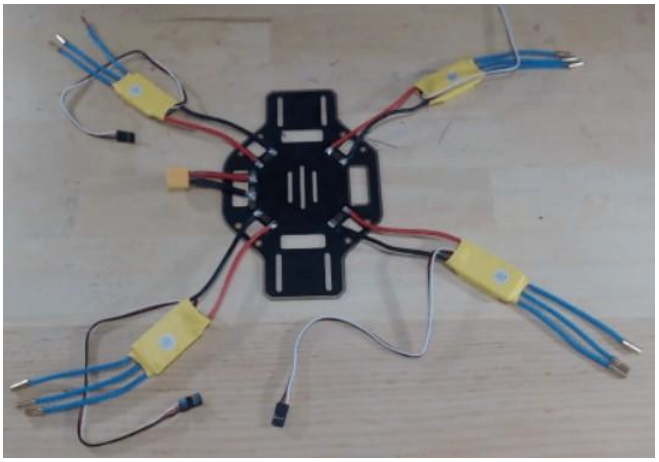
fig 18: Overall Flowchart



5.1 Drone : Assembling

Connect the LiPo battery to the power distribution board(PDB) and attach it in the middle of the board to provide parallel supply of 11V to the ESCs.

fig 19: ESC + Frame + PDB



Fix your ESCs to the drone's arms: one on every arm. The ESCs can be either on the top or bottom of the arm.
Screw the motors to the end of each arm. The motors should come with mounting screws. Be sure not to over-tighten. We will have two CW and two CCW motors, they should be

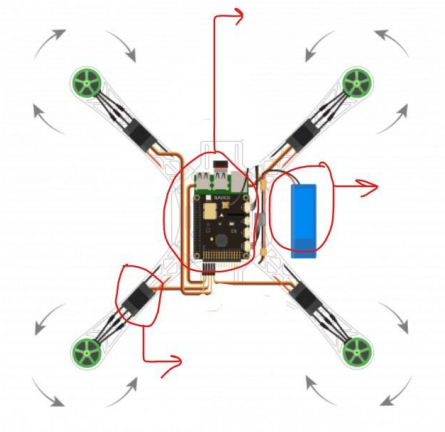


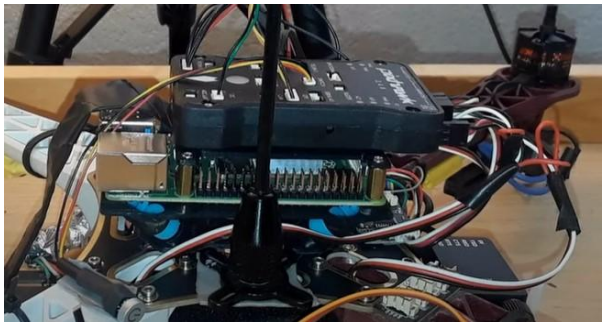
fig 20: Propeller direction

placed on the correct arms according to this diagram.

Complete instructions can be found in this link for assembling the drone : https://dojofordrones.com/build-a-drone/

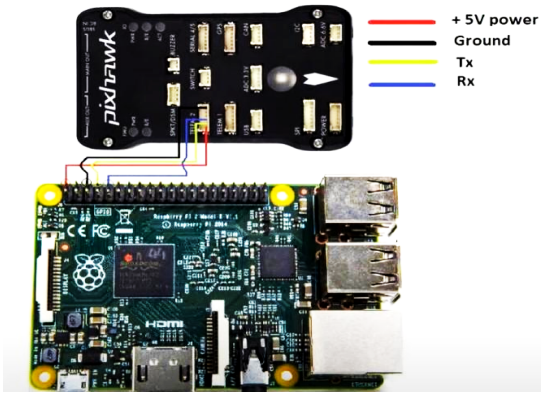
After the drone is assembled, we will move onto the connection between the raspberry pi and pixhawk.

fig 21: PIXHAWK and raspberry pi on frame



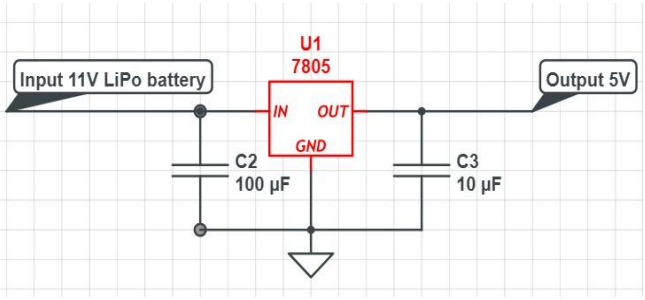
The arrow on pixhawk should point in the USB ports direction of raspberry pi. The pin connection diagram is shown below.

fig 22: PIXHAWK - Raspberry pi connection



Now since the raspberry pi requires a 5V supply and PixHawk requires 8V, we will move onto the LiPo 11V to the required voltage supply circuit.

fig 23 : Voltage regulator



Similar will be the circuit for 8V output where instead of 7805 we will use the 7808 IC.

5.2 Calculations

- Power wastage in voltage regulator :

$$Heat\ Dissipated\ P\ =\ |(V_{out} - V_{in})|\ *\ I_{out}$$

For Raspberry Pi, output voltage is 5V and input is 11V, current drawn in idle state is 400mA, thus power wastage in raspberry pi,

$$P_{raspberry}\ =\ 6\ *\ 400/1000\ =\ 2.4W$$

For PIXHAWK flight controller, output voltage is 8V and input is 11V, current drawn is 280mA, thus power wasted in pixhawk,

$$P_{pixhawk}\ =\ 3\ *\ 280/1000\ =\ 0.84W$$

- Max theoretical current that can be supplied by the LiPo 3000mAh battery can be given by multiplying the C-rating of the battery and the capacity of the battery.
  Max current = 3000/1000 * 60 = 180 A.

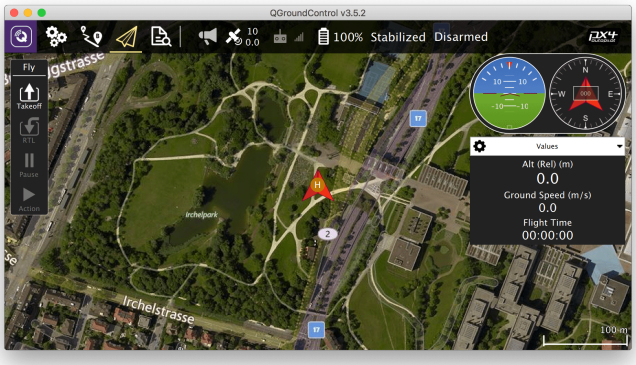
5.3 Drone: Setup

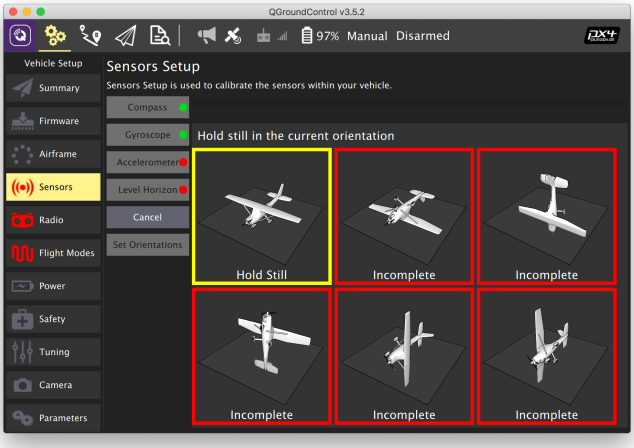
We aim to fly the drone autonomously. Hence, we need to program our on-board computer (Raspberry Pi 3) and our flight controller (Pixhawk). To achieve this, we plan to use Clover. It is an open-source ROS-based framework, providing user-friendly tools to control PX4-powered drones. Clover is available as an ROS package but mainly the preconfigured image for Raspberry Pi is shipped and used.
This pre-configured image for Raspberry Pi comes with complete set of required software for working with peripheral devices and programming autonomous flights. It is compatible with our RaspberryPi 3 and Pixhawk.

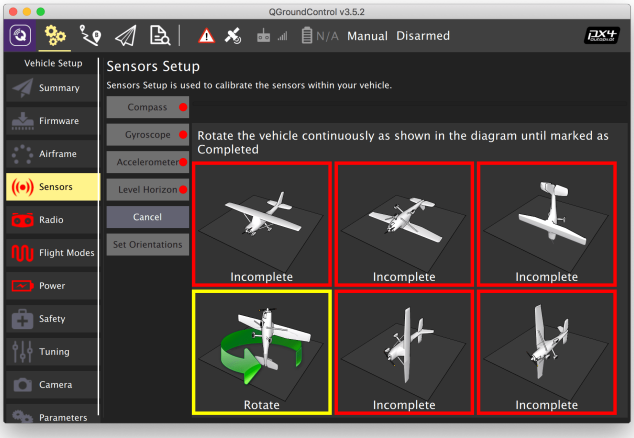
Configuring flight controller

We will also use a software called QGroundControl to flash, configure and calibrate the flight controller. First, we will have to download flash the latest stable version of clover (https://github.com/CopterExpress/Firmware/releases/download/v1.8.2-clover.13/px4fmu-v4_default.px4) to the flight controller using QGC. Then we will need to configure the flight controller. The following parameters will need setup: Airframe, Radio, Sensors, Flight Modes. Once this is done, we would have to select our airframe in QGC. Then we can proceed to calibrating our drone sensors (gyro-meter, accelerometer, compass etc.). After this, we would also have to do the power setup and ESC calibration. All this can be easily done using the QGC software.

fig 24: QGroundControl application







Configuring RPi

The Clover RPi image mentioned before contains all the necessary software for working with and programming autonomous flights. The platform is based on Raspbian OS and robotics framework ROS. The complete source code of the image builder and all the packages is available on GitHub (https://github.com/CopterExpress/clover). All we have to do is flash the image on the SD card (using any relevant software in PC) and put the card into RPi. Once this is done, we can connect to RPi using Wi-Fi, and gain access to it using SSH. RPi image provides a pre-configured Wi-Fi access point. Once we connect to this WiFi, we can use SSH to access the RPi and run commands. We can also use QGroundControl via WiFi and we can monitor, control, calibrate and configure the flight controller.

5.4 Drone: Programming

Once the drone is assembled and setup ready for use all we need is to program it so that it can fly autonomously as we want it to. Once all the above steps are done, we need to import rospy and we can move the drone 2 mts in x-direction as simply as writing the below code and running it:

```
navigate(x=2, y=0, z=0, frame_id='body')
```

Hence, we can implement our algorithm and simply after each iteration navigate to x = x and y = y (x, y are the difference of next and current coordinates). Our final code would look something as in Fig:24:

```
def changeLoc(I,delta,stepSize):
  dIx = (I['E']-I['W'])/(2*delta)
  dIy = (I['N']-I['S'])/(2*delta)
  dIz = (I['U']-I['D'])/(2*delta)
  lenI = sqrt(dIx**2+dIy**2+dIz**2)
  dx = stepSize*dIx/lenI;
  dy = stepSize*dIy/lenI;
  dz = stepSize*dIz/lenI;
  return (dx),(dy),(dz)

def find_source(delta = 0.2,stepSize = 0.5 ,x=0, y=0, z=1, yaw=float('nan'), yaw_rate=0, speed=0.5, frame_id='body', tolerance=0.2, auto_arm=False):

    I = {'E':pi.read(E),'W':pi.read(W),'N':pi.read(N),'S':pi.read(S),'U':pi.read(U),'D':pi.read(D)}
    trajectory = [[x,y,z]]
    error = stepSize+1
    while error>=stepSize:
      dx,dy,dz = nextLoc(I,delta,stepSize)
      res = navigate(x=dx, y=dy, z=dz, yaw=yaw, yaw_rate=yaw_rate, speed=speed, frame_id=frame_id, auto_arm=auto_arm)
      if not res.success:
        return None


      trajectory.append([dx+trajectory[-1][0],dy + trajectory[-1][1],dz + trajectory[-1][2]])
      try:
        #error is the distance between current point and previous-to-previous point
        error = sqrt((trajectory[-1][0]-trajectory[-3][0])**2+(trajectory[-1][1]-trajectory[-3][1])**2+(trajectory[-1][2]-trajectory[-3][2])**2)
      except:
        pass
      I = {'E':pi.read(E),'W':pi.read(W),'N':pi.read(N),'S':pi.read(S),'U':pi.read(U),'D':pi.read(D)}

    land()
    return trajectory
```

fig 25: Code

Final Steps

After all this, all that is left to do is to SSH to RPi on the drone and run the python script. The drone should fly autonomously and eventually land at the source of gas leak.

6 Further Work

The solutions discussed till now are what was achievable by us in terms of a readily usable prototype. But when we thought about scaling our solution for industrial usage, we developed a modified design. While dealing with gas detection and monitoring on an industrial scale, we realised that a advanced set of sensors & techniques can be used to generate real-time 3D mappings of the gas concentrations and that too with a high level of accuracy. The following section covers our analysis of this solution, which is more on a qualitative basis rather than a quantitative basis as this technology is a bit out of our scope at this point of time.

6.1 LiDAR-based gas mapping system

Light detection and ranging (**LiDAR**) techniques use **lasers** to create 3D (**topographic**) or gas concentration (**atmospheric**) imagery of the surveyed environment. Both uses for LiDAR can be performed using either pulses of laser light (pulsed lasers) or laser light that stays on all the time (continuous-wave lasers). LiDAR is a remote sensing method that uses light in the form of a laser to measure ranges (variable distances) to the scene. This light, after it interacts with the scene and combined with other data recorded by other sensing mechanisms in the system, generates precise, three-dimensional information about the scene's topography i.e. surface characteristics, and gas concentrations by interacting with the gases in the medium.

6.2 Why is LiDAR needed ?

Although the prototype suggested by us does a good job in small scales, and might as well work well in industrial scales, we figured out that the latter case brings in some new challenges:

(1) In a factory, we are expected to always deal with average background levels of some gases and will be interested in detecting anomalous levels of these gases.

(2) In this case, we would like to have a continuous three -dimensional feed in real-time of the gas concentrations in different parts of the factory as the gases are dangerous & people are working in close proximity to them.

(3) Not only we are interested in the sources of gas leakages, but also in their intensity & spread after the leakage. And we expect to do all this keeping in mind that there might be obstacles such as large machines & parts blocking the view of our sensors.

The use case we are targeting is primarily industries under a roof which deal with gases (particularly, light heteromolecular games like CO2 & CH4). We have designed our system for the industry (factories, production units, warehouses) where the gases are used in an indoor setting. In this case, we want to provide our customers dealing with gases (both hazardous & non- hazardous) with real-time 3D mapping of a large area enabling them with:

- gas leakage **source detection.**
- gas leakage **rate quantification.**
- gas **concentration & spread analysis.**

6.3 Our proposed solution

We are planning on using LiDAR technology based on **frequency-modulated continuous-wave (FMCW) ranging**, for high-resolution 3D imagery and simultaneous path integrated gas concentration measurements via **wavelength modulation spectroscopy (WMS)** integrated with an **efficient system of lightweight & mobile** LiDAR sensors.

Explanation of the entire system:

(1) Our **mobile LiDAR sensors** are fitted on rails or hanging cables on the roof of the factory. Using the motion of the sensors allows us to create a **synthetic aperture** and thus, we can cut cost on the size & number of sensors required.

(2) Continuous real-time LiDAR measurements are performed by transmitting laser beams on the factory grounds and collecting light backscattered from the objects on the topography.

(3) Imagery is generated by spatially scanning the transmitted laser beam. Encoder measurements of the transmitted beam angle are collected to enable reconstruction of the point measurements into images.

(4) The **combination of range and path-integrated gas concentration measurements** allows computation of the **average gas concentration** between the sensors and each point on the factory grounds.

(5) The **average concentration** can be used to estimate the **quantity of anomalous gas**, even in the cases where distinct plumes of the gases are not clearly visible.

(6) The wavelength of the laser has to be chosen corresponding to the gas to be detected as the laser interacts with the molecules of the gas. Our system allows us to deploy **multiple LiDAR sensors**, each monitoring **a different gas**.

The most novel element of our solution is the data collection through mobile sensors. When we bring the motion of our mobile LiDAR sensors into the picture, the following things become accessible:

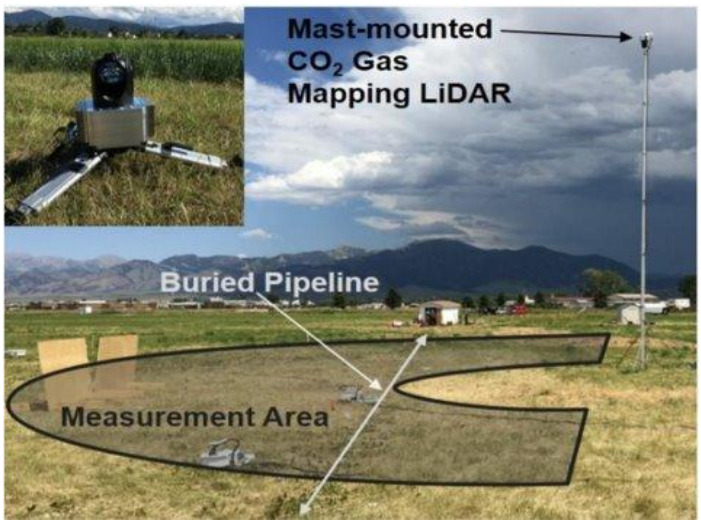
- It allows us to cut down on cost by reducing the **number of sensors**, the **quality of the sensor involved**, and hence also **power** as the sensors use a lot of power.
- When combined with sensor motion, the type of data we are receiving in the previous section also enables **3D gas plume reconstruction** via **tomography** for **volumetric localization** and **3D concentration estimates.** This allows us to see where the gas has spread after the leakage. Tomography is **imaging by sections or sectioning through the use of any kind of penetrating wave**.
- Combining this with positional elements of the sensors like **location, orientation, angle & range of the transmitted beam** allows us to bring into the picture the **objects like machines & parts** on the factory grounds. Our 3D model will be able to capture the gas leaks & spreads even if they occur behind the **obstacles**.

Two important parts in which we have to do some due diligence are:

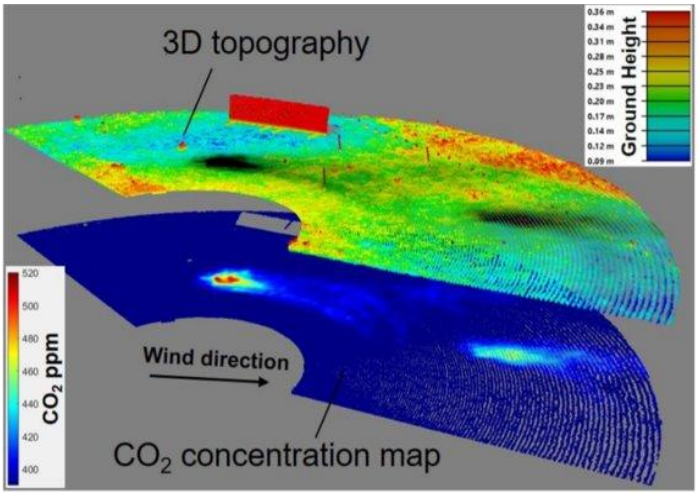
- exact trajectory, motion parameters and number of our overhead mobile LiDAR sensors. Parameters such as position & orientation of the sensors at different times to ensure optimal & accurate sensing will have to be modelled by doing experiments and seeing the data.
- application of analytics to stitch the incoming data from all the sensors and generate the 3D plots of the scenes. All the methods of processing the data have to be explored.

Research on the above two domains have already been done and the solutions are available. But, we have not spent enough time on this due to lack of corresponding data.

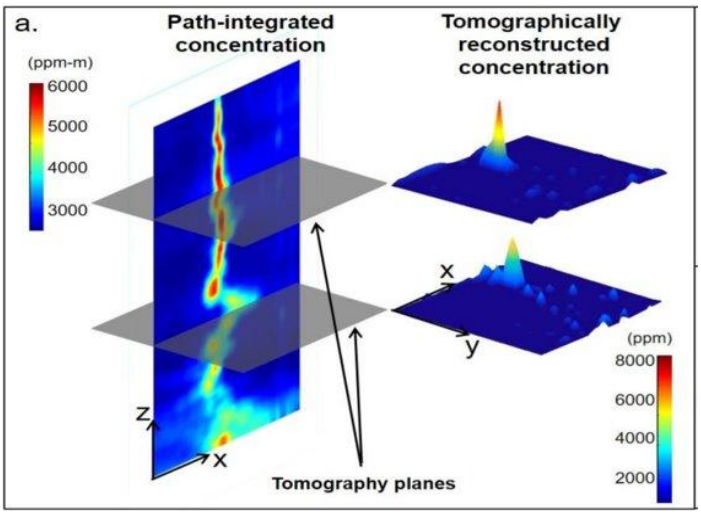
Here is a similar example implemented by Bridger Photonics where a fixed LiDAR sensor has been used to monitor CO2 levels coming from a buried gas pipeline used to supply CO2 to crops.



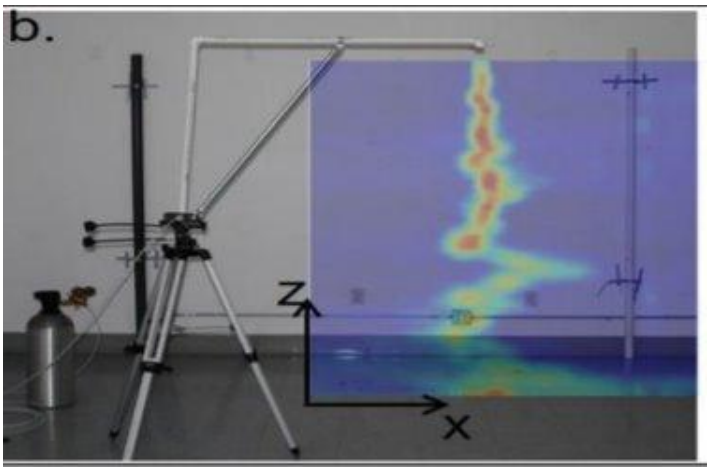
We can see the two distinct plumes through which the gas is coming out in the image on the right. Both, the topography & CO2 concentration plots are made available.



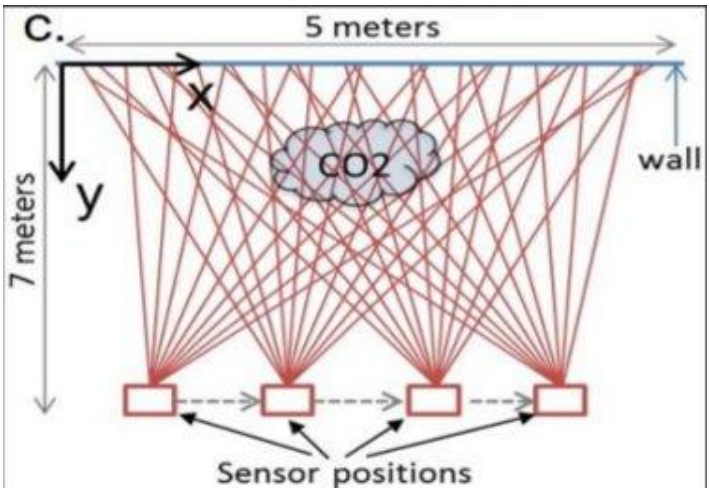
Here is an example of a plume of CO2 gas captured with multiple LiDAR sensors placed at slightly different locations. Using tomographical methods, the plume is reconstructed in 3-D setting, with the concentration of the gas clearly visible.



In this image, it is clearly visible that the gas is coming out of the nozzle and the plumes & the spread of the gas release is mapped in the image.



There is a diagram which shows the sensors used by Bridger Photonics to capture this gas plume. It shows the orientation in which all the sensors were kept.



10. Work Distribution

10.1 V S S Anirudh Sharma, EE18B036

- Prototyped Mark I, II and III of Project Safe House
- Created the Gas Leak Detector App using Blynk.
- Devised Algorithms 1 and 2 (Gradient Ascent and Normalized Gradient Ascent algorithms)
- Conceptualized the gas peak detection sensor system
- Generated all the simulations and made observations.
- Made all the codes for the drone operations with Ramadarsan.

10.2 Anugyan Das, EE18B039

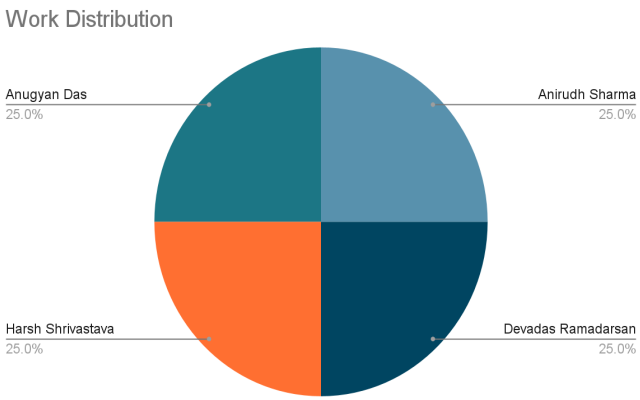
- Conceptualized the design of our prototype.
- Designed the entire pipeline of our small scale prototype & our proposed large scale solution
- Studied and conceptualized the future work of the project, looking into how it can be evolved and suited further to industrial settings.
- Did a detailed study of all the sensor mechanisms which can be used for gas detection.
- Presented analysis of LiDAR systems & set of moving sensors working together.

10.3 Devadas Ramadarsan, EE18B045

- Researched on setting up the drone. Including, configuring and calibrating flight control.
- Researched on required softwares and packages for Pixhawk and RPi.
- Researched and studied on how to move the drone based on the trajectory obtained from gradient ascent algorithm
- Worked with Harsh on how to connect Rpi and Pixhawk.
- Made all the codes for the drone operations with Anirudh

10.4 Harsh Shrivastava, EE18B049

- Researched and selected the required electronic equipment.
- Studied and documented the systematic assembling of the drone structure along with the other electronics equipment.
- Made a power distribution circuit so that only one battery can satisfy multiple requirements.
- Worked with Ramadarsan on how to connect Rpi and Pixhawk.



11. References

1. https://github.com/BirchJD/RPi_433MHz
2. https://clover.coex.tech/en/
3. https://pypi.org/project/rpi-rf/
4. https://news.stanford.edu/2019/06/07/trying-stop-natural-gas-leaks-wake-destruction/
5. http://wiki.ros.org/ROSberryPi/Installing%20ROS%20Indigo%20on%20Raspberry%20Pi
6. http://wiki.ros.org/mavros
7. https://dojofordrones.com/build-a-drone/
8. https://thepihut.com/blogs/raspberry-pi-tutorials/how-to-use-voltage-regulators-in-a-circuit
9. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8082872
10. https://github.com/CopterExpress/clover
11. https://www.researchgate.net/publication/283690136_Trace_Gas_Detection_with_Lidar_from_Space
12. https://www.bridgerphotonics.com/blog/gas-mapping-lidartm-explained
13. https://energyfactor.exxonmobil.com/reducing-emissions/methane/lidar-methane-leak-detection/