

# EGH455

## Group 2

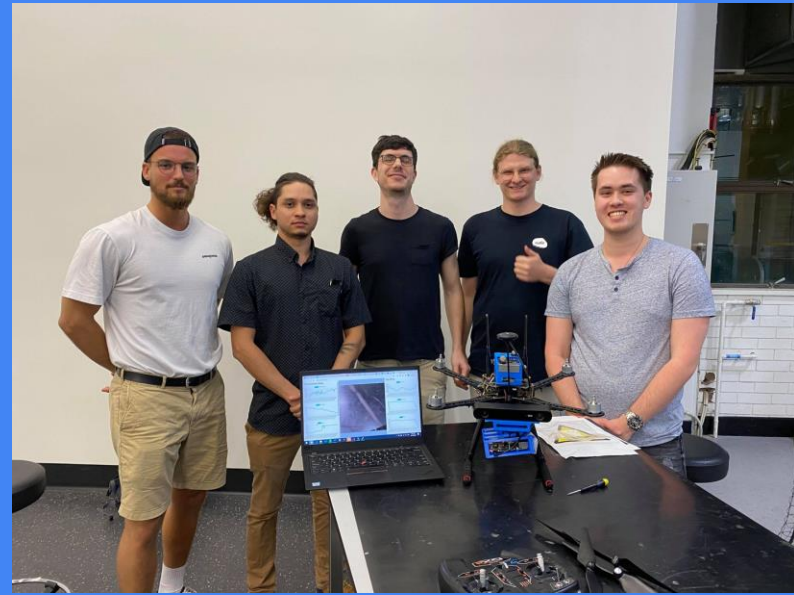
Alexander Iftene - Project Manager/Enclosure

Adrian Hiltunen - Sensor Data

James Arnold - Image Detection and Processing

Oliver Campbell - Image Detection and Processing

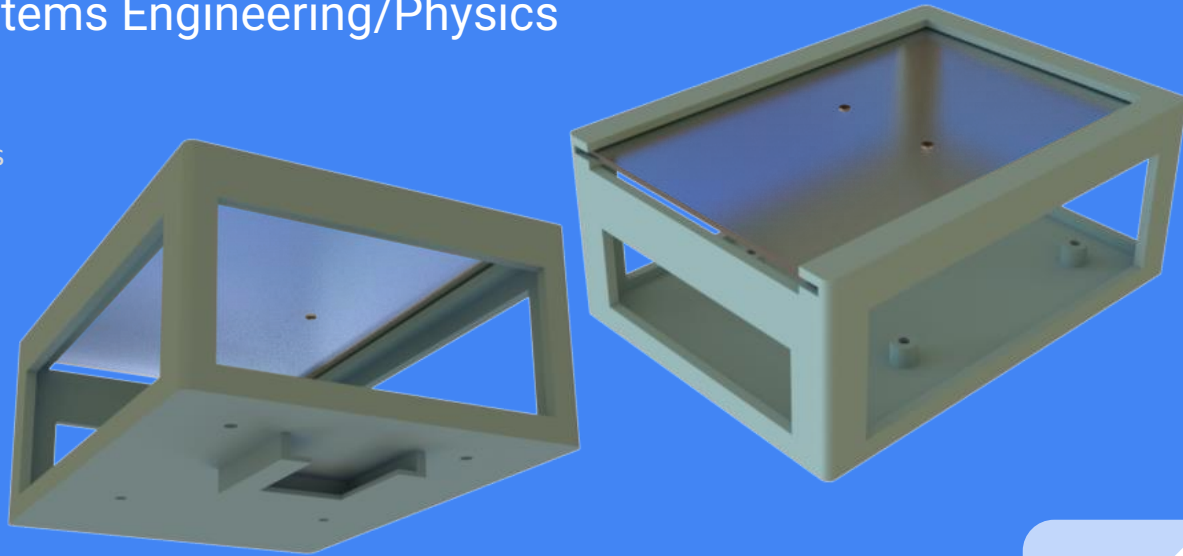
Brian Sivertsen - Web Visualisation



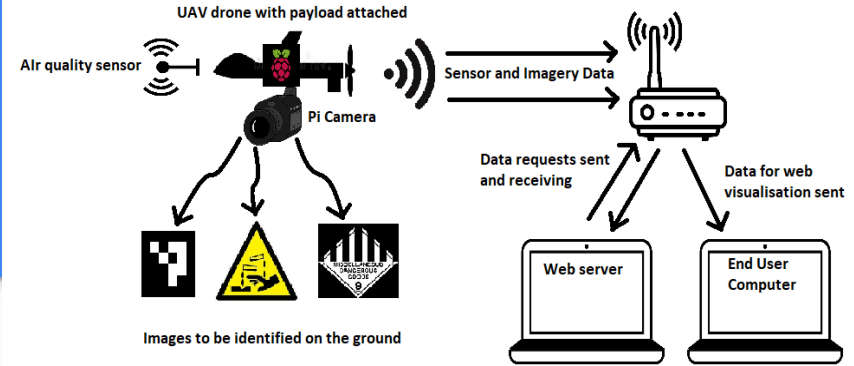
# Alexander Iftene

Project Manager/Enclosure Design  
Computer and Software Systems Engineering/Physics  
Final year of study

- Oversee team members responsibilities and subsystems from a high level system perspective, ensuring deadlines are met.
- Assist team members where required, especially for dependent tasks.
- Organise meetings (including booking rooms and organising locations) and maintain meeting minutes.
- Design and create 3D printed container to mount RPi and Camera to drone.
- Manage budget (money, payload weight, power, etc.).



# Introduction and Conops



**REQ-M-01: Description:** The UAVPayload<sup>TAQ</sup> shall remain under the maximum weight of 250 g.

**Rationale:** The UAV that will carry the payload is required to remain under the maximum take-off weight of 2.2 kg.

**REQ-M-03: Description:** The UAVPayload<sup>TAQ</sup> shall communicate with a ground station computer to transmit video, target detection and air quality data.

**Rationale:** The UAVPayload<sup>TAQ</sup> is a standalone embedded system that should be adaptable to different UAV platforms.

**REQ-M-8: Description:** Preliminary designs shall be completed by week 7

**Rationale:** There are time constraints. In order to know what parts to purchase, the preliminary designs must be completed before week 7.

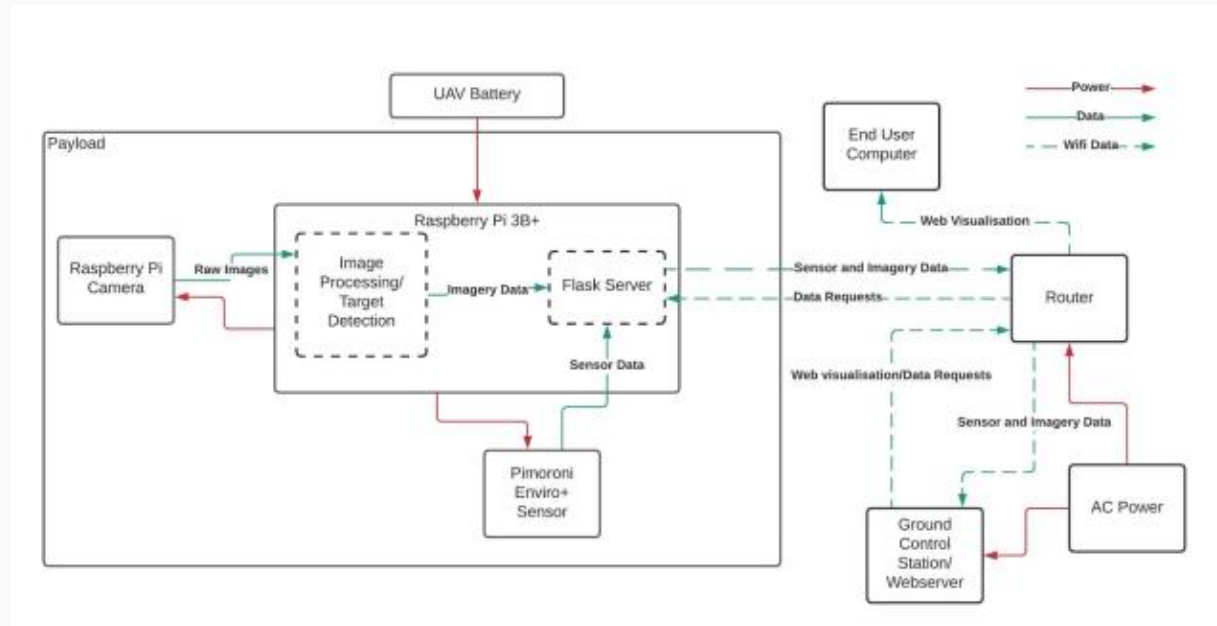
**REQ-M-9: Description:** Additional purchase orders for the UAVPayload<sup>TAQ</sup> components shall be placed before week 7.

**Rationale:** There are time constraints. In order to start the build, the UAV parts must be ordered to allow time for delivery.

**REQ-M-10: Description:** Developed solution shall conform to the systems engineering approach.

**Rationale:** This is a requested requirement from the customer to ensure that the UAV is designed and built to a specific standard.

# System Architecture



# Work Breakdown Structure (WBS) And Work Packets (WP)

## Work Packets:

WP-UAVPLD: Work Packet for the UAV Payload Project Team

WP-ENC: Work Packet for the Enclosure subsystem

WP-DAT: Work Packet for the Data acquisition and sensing subsystem

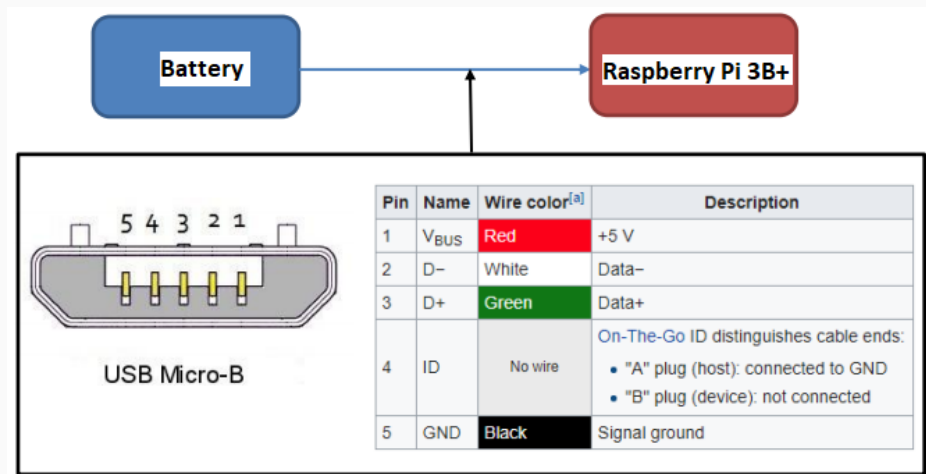
WP-IMG: Work Packet for the Image Processing subsystem

WP-WEB: Work Packet for the Web Visualisation subsystem



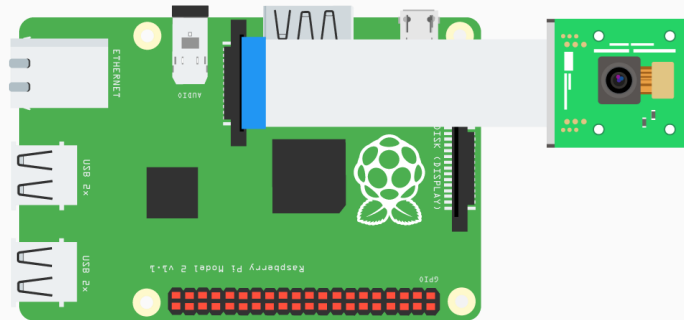
# Interface Control Document (ICD) - Raspberry Pi 3B+

- Interface is required to supply power from external rechargeable battery.
- Minimum of 2 amps with 5V.
- Supplied with Micro USB type B.
- Pins 1 and 5 are utilised with VCC at 5V.



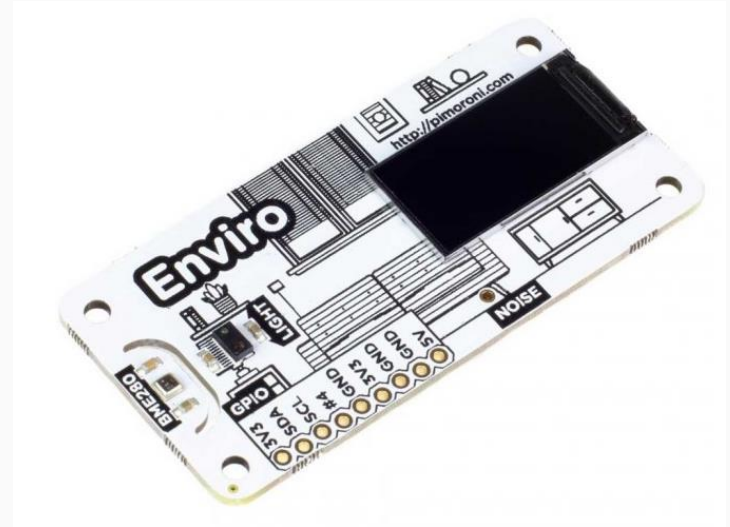
# Interface Control Document (ICD) - Raspberry Pi Camera

- Proprietary interface developed by Raspberry Pi.
- Both power and data interface system.
- Connector is a flat ribbon cable with push pin on both ends.



# Interface Control Document (ICD) - Pimoroni Enviro+ Sensor Hat

- Interfaces power from the Raspberry Pi board.
- Uses the male GPIO header from board to connect.
- Powered by 5V VCC through male GPIO header.





# Budget

- Budget was quite low cost.
- No components were broken/replaced during the development phase.
- Budget was easy to manage and additional purchases were not necessary to complete project on time.

Component	Value
Raspberry Pi 3B+	70
Raspberry Pi 3B+ Case	12.5
Raspberry Pi 3B+ Power Supply	18.95
Raspberry Pi Camera module	38.95
Pimoroni Enviro+ Air Quality Sensor	99.95
16GB Micro SD Card	18.7
SD Card Reader	20.95
<b>Total</b>	<b>\$280</b>

# Timeline

Weeks 1 - 6 (Planning)	Weeks 6 - 8 (Implementation)	Weeks 8 - 10 (Testing)	Weeks 10 - 12 (Optimising)
Determining roles and responsibilities.	Web Visualisation system underway and being developed.	Testing of web system integration with sensory data and image data.	Tweaking all the subsystems so they operate optimally in flight with payload.
Designating workloads and objectives.	Sensory system underway and developing.	Checking to see if basic image processing requirements were met.	Printed final model of enclosure for payload.
Familiarizing with subsystems and customer requirements for the project.	Image Processing team developing system.	Sensory system debugging and integration.	Ensuring payload is within weight requirements of customer needs.
Developing PMP report for customer.	Designing of enclosure for payload underway.	Enclosure printing and re-printing.	Present and submit final PMP.

# Adrian Hiltunen

Sensor Data Acquisition - Electrical & Software Systems - Final Year

# Sensor Data Subsystem: Introduction & System Requirements

## System Requirements

- i. **REQ-M-02/HLO-M-1:** The UAV Payload device must measure and return readings of hazardous gases, air pressure, humidity, temperature, light, and noise levels via sensors onboard the device.
- ii. **REQ-M-03, HLO-M-1:** The UAV Payload must communicate wirelessly to a GCS via a standalone embedded system.
- iii. **REQ-M-05, HLO-M-3:** Real time air sampling data must be recorded and dynamically updated throughout the UAV's flight through a Web Interface.

## Work Packets

WP-DAT-1: Sensor Research

WP-DAT-2: Hardware Implementation

WP-DAT-3: Communication Design

WP-DAT-4: Integration Development

WP-DAT-5: Integration Testing

# Sensor Data Subsystem: Architecture

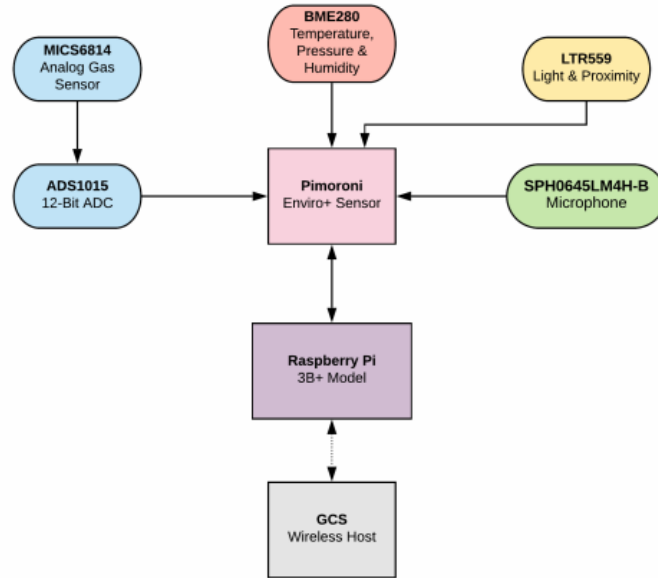


Image courtesy of Pimoroni:  
<https://shop.pimoroni.com/products/enviro?variant=31155658457171>

# Sensor Data Subsystem: Design

## Hardware:

- Pimoroni Enviro+ Sensor Hat for Raspberry Pi

- All-in-one solution

- Provided by design brief

- Datasheet compilation & investigation

## Software:

- Primary foundation sourced from existing driver library

- Scripted modules written in Python

- Additional data formatting required

# Sensor Data Subsystem: Design

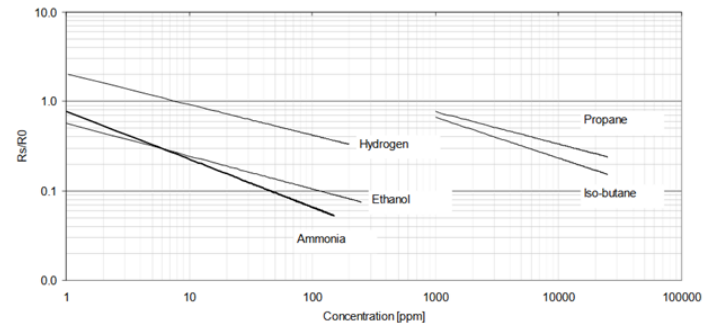
$$m = \frac{\log(F_2) - \log(F_1)}{\log(x_2) - \log(x_1)} = \frac{\log(F_2/F_1)}{\log(x_2/x_1)} \dots 1$$

AXIS FLIP'd	X1	X2	Y1	Y2	M	C
RED	3.5	0.01	1	1000	-1.179	4.382
OX	0.065	40	0.01	6	0.996	0.152
NH3	0.78	0.065	1	100	-1.853	0.631

$$F(x) = \text{constant} \cdot x^m \dots 2$$

$$\frac{F(x)}{C} = x^m \rightarrow \left(\frac{F(x)}{C}\right)^{\frac{1}{m}} = x \dots 3$$

$$C = \frac{F(x)}{x^m} \dots 4$$



NH3 sensor, continuous power ON, 25°C, 50% RH

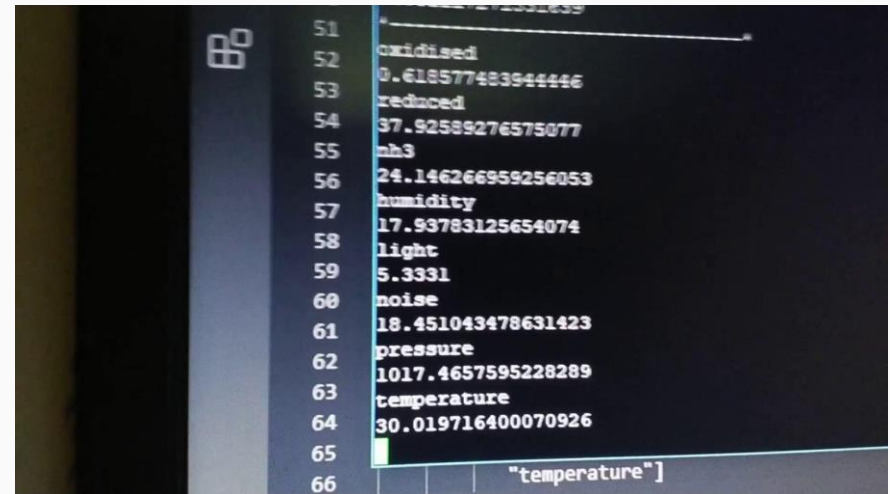
# Sensor Data Subsystem: Design

```
def updateSensors():  
  
    data = gas.read_all()  
    data_unraised = [56000.0/data.oxidising, 56000.0/data.reducing, 56000.0/data.nh3]  
    data_powers = [0.996, -1.179, -1.853]  
    data_raised = numpy.power(data_unraised, data_powers)  
  
    cpu_temp = get_cpu_temperature()  
    raw_temp = bme280.get_temperature()  
    compensated_temp = raw_temp - ((cpu_temp - raw_temp) / factor)  
  
    low, mid, high, amp = noise.get_noise_profile()  
  
    values["oxidised"] = data_raised[0]*0.152  
    values["reduced"] = data_raised[1]*4.382  
    values["nh3"] = data_raised[2]*0.631  
    values["humidity"] = bme280.get_humidity()  
    values["light"] = ltr559.get_lux()  
    values["noise"] = amp*64  
    values["pressure"] = bme280.get_pressure()  
    values["temperature"] = compensated_temp
```



# Sensor Data Subsystem: Test Report

- Functionality proven through proximity testing with known chemicals
- Temperature data offset by heatsink proximity to sensors
- Pressure and gas readouts cross referenced with known values around the world
- Light tested with torch and blackout
- Spoken audio test



51	oxidised
52	0.618577483944446
53	reduced
54	37.92589276575077
55	nh3
56	24.146266959256053
57	humidity
58	17.93783125654074
59	light
60	5.3331
61	noise
62	18.451043478631423
63	pressure
64	1017.4657595228289
65	temperature
66	30.019716400070926
	"temperature"]

# Sensor Data Subsystem: Conclusion

Lessons learned:

- Don't always rely on user sourced libraries
- Sometimes better to write your own drivers from scratch

Further work includes:

- Defining noise levels in dBu / dBFS
- Integration: module `__init.py__` placement (file hierarchy)



# Oliver Campbell

Image Processing and Target Detection - Mechatronics Engineering - Final Year

# James Arnold

Image Processing and Target Detection - Software and Computer Science  
Engineering - Final Year



## System Requirements

- **REQ-M-3/HLO-M-1:** The UAVPayload<sup>TAQ</sup> shall communicate with a ground station computer to transmit video, target detection and air quality data.
- **REQ-M-4/HLO-M-2/HLO-M-3:** The target identification system shall be capable of alerting the GCS of a targets type.
- **REQ-M-6/HLO-M-3:** The Web Interface is required to display the images of the targets that are taken directly from the UAVPayload<sup>TAQ</sup> and updated every time a new picture is taken.
- **REQ-M-12/HLO-M-2/HLO-M-3:** The UAVPayload<sup>TAQ</sup> shall process all imagery on-board via the on-board computer.
- **REQ-M-13/HLO-M-2/HLO-M-3:** The processing must be able to analyse all data from the flight path while the UAV moves at a maximum speed of 1m/s.
- **REQ-M-14/HLO-M-2/HLO-M-3:** The processing must be able to analyse all data from the flight path while the UAV operates at an altitude of 2m.

## Work Packet Structure

WP-IMG-1 - Image Capture and Research

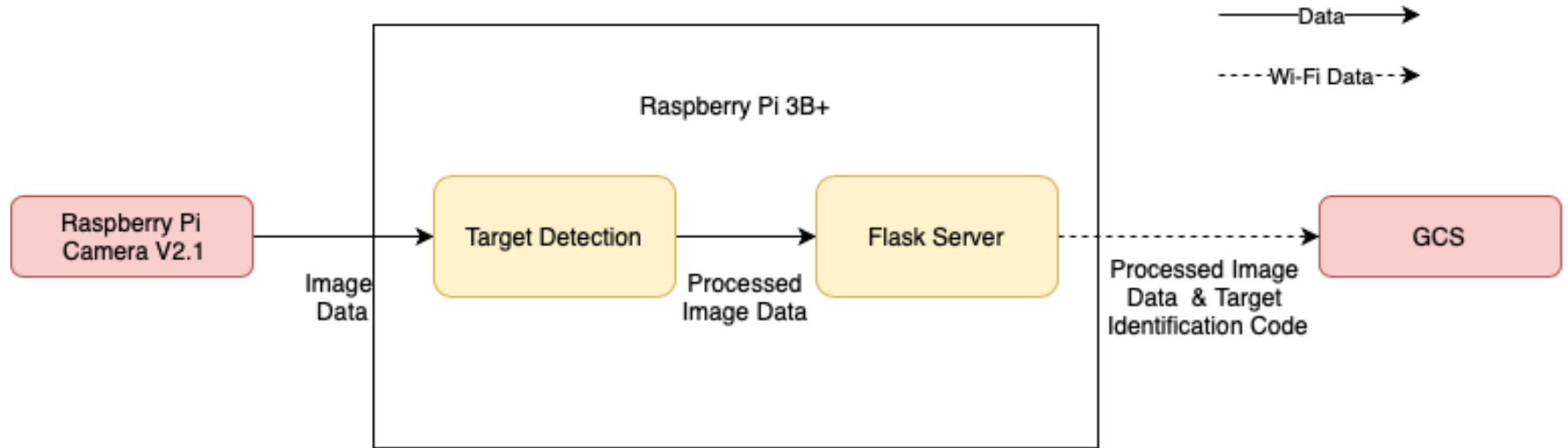
WP-IMG-2 - Define Training Dataset

WP-IMG-3 - Model Definition

WP-IMG-4 - Train Machine Learning Model

WP-IMG-5 - Unit Testing

# Subsystem Architecture



# Code Design

```
89 #Call Aruco, return 3 if detected
90 if (target_id == 0):
91     target_id = detectMarkers(image, dictionary, param, grayscale)
92
93 #Call mdg9 classifier, return 2 if detected
94 if (target_id == 0):
95     target_id = mdg9_classifier(image, grayscale)
96
97 #Call toxic classifier, return 1 if detected
98 if (target_id == 0):
99     target_id = toxic_classifier(image, grayscale)
100
101 #If target ID is still 0, save image anyway with no boxes and exit loop.
102 if(target_id == 0):
103     writeName = '/home/pi/photos/output/outputimage.jpg'
104     cv2.imwrite(writeName, image)
105     print('no classifiers detected')
```

```
23 def toxic_classifier(image, grayimg):
24     # load cascade classifier
25     cc = cv2.CascadeClassifier('cascadetoxnew.xml')
26     # run Cascade Classifier on image
27     results = cc.detectMultiScale(grayimg, 1.05, 3)
28     print (results)
29     if len(results)>0:
30         # draw rectangles around each positive result
31         for x,y,w,h in results:
32             cv2.rectangle(image, (x,y), (x+w, y+h), (0,255,0)
33             writeName = '/home/pi/photos/output/outputimage.jpg'
34             # save image to folder
35             cv2.imwrite(writeName, image)
36             return 1
37     else:
38         return 0
```

# Cascade Classifier Training

## Training Parameters:

- # of positive images, # of negative images
- # of stages
- minhitrate, maxfalsealarm

## Final Parameters:

- 850 pos images, 150 neg images, 6 stages, 0.3 hit rate, 0.1 false alarm
- Images at 100x100 resolution with target overlayed if positive
- Ensure samples contain target in every orientation

# Cascade Classifier Testing

- Print each target out to an A4 sheet
- Capture images at distances of 0.5, 1 and 2m.
- Orientate target in a variety of ways
- Record number of correct detections, false positives and missed detections
- Change one parameter at a time to minimise variables.



# Conclusion

- Could have had a more objective testing process (e.g. target >90% success rate and <10% false positive)
- Could have tested other machine learning with additional time.
- Overall, successfully met customer requirements and satisfied with performance.

# Brian Sivertsen

Web Visualisation - Software and Computer Science Engineering - Final Year



# Web Visualisation Introduction

## System Requirements:

- **REQ-M-3:** The UAVPayload<sup>TAQ</sup> shall communicate with a ground station computer to transmit video, target detection and air quality data.
- **REQ-M-4:** The target identification system shall be capable of alerting the GCS of a targets type.
- **REQ-M-5:** The Web Interface is required to display real time air sampling data that is recorded directly from the UAVPayload<sup>TAQ</sup> and updated dynamically throughout the duration of the flight.
- **REQ-M-6:** The Web Interface is required to display the images of the targets that are taken directly from the UAVPayload<sup>TAQ</sup> and updated every time a new picture is taken.
- **REQ-M-7:** The Web Interface shall be designed and run as a web server, which is to be accessible by any computers on the local network.
- **REQ-M-15:** Live data from the UAV must be made available through the web server within 10 seconds of capture.

## Work Packets:

WP-WEB-1: Feasibility Research

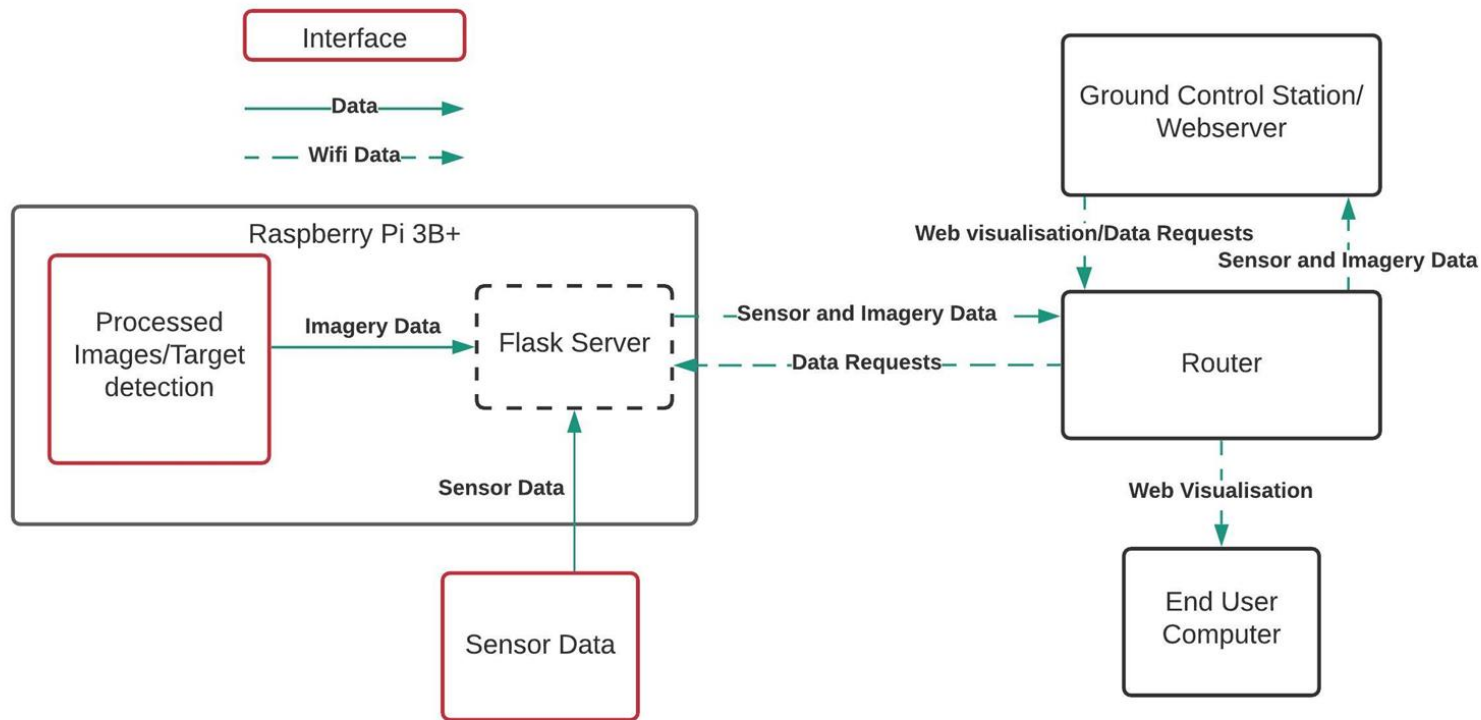
WP-WEB-2: Backend Preliminary Design

WP-WEB-3: Front End Preliminary Design

WP-WEB-4: Subsystem Development

WP-WEB-5: Integration Testing

# Web Subsystem Architecture



# Back End Server Design

```
#Import packages above
import sys
sys.path.insert(1, '/home/pi/enviropius/examples')
import dataSense

app = Flask(__name__)
@app.route('/data')
def get_current_data():
    d = dataSense.returnSensors()
    id = open("/home/pi/pythonscripts/detectionFile.txt", "r")
    d["id"] = id.read()
    return jsonify(d)

@app.route('/get_image')
def get_image():
    return send_file('/home/pi/photos/output/outputimage.jpg')
```

← → ↺ 🏠 ⚠ Not secure 192.168.43.113:5000/data

```
{
  humidity: 36.312575027605625,
  id: "0",
  light: 45.2082,
  nh3: 0.023396810545597963,
  noise: 15.996364231394265,
  oxidised: 0.24942645135306174,
  pressure: 1013.8826293833631,
  reduced: 1.0039478116889995,
  temperature: 6.132050825639343,
  time: 1603106876.655459
}
```

⚠ Not secure 192.168.43.114:5000/get\_image



# Front End Design

```
useEffect(() => {  
  fetch('/data').then(res => res.json()).then(data => {  
  
    data.time2=Date.now();  
    setData(data);  
    let newGraphs = graphs;  
    if (graphs.length < 50)  
      newGraphs.push(data);  
    else {  
      newGraphs.shift()  
      newGraphs.push(data);  
    }  
    if (data.id == "1"){  
      let toxic = new Audio("./toxic.ogg")  
      toxic.play();  
    } else if ( data.id == "2"){  
      let dangerous = new Audio("./dangerous.ogg")  
      dangerous.play();  
    } else if ( data.id == "3"){  
      let aruco = new Audio("./aruco.ogg")  
      aruco.play();  
    }  
    setGraphs(newGraphs);  
  });  
});
```

Environment Data



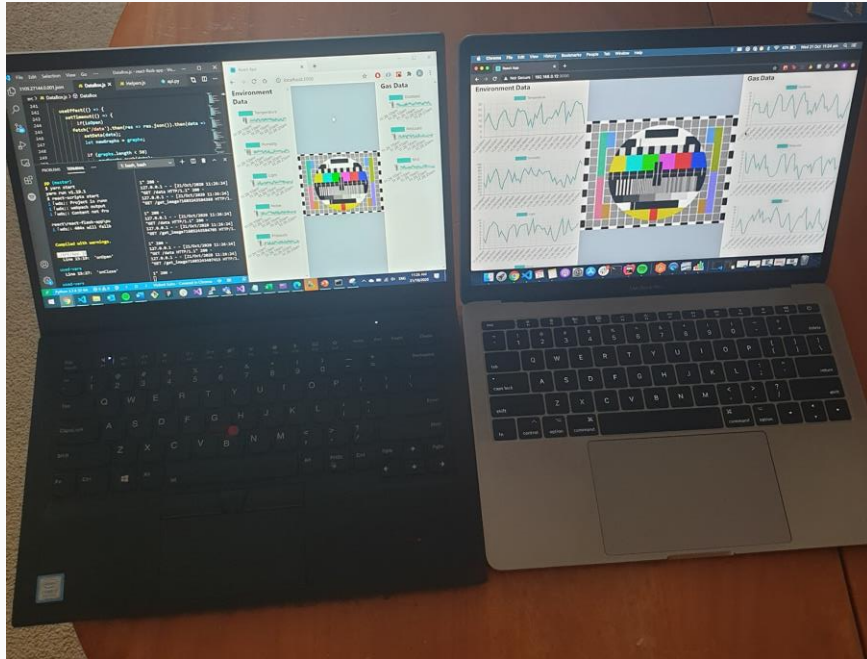
Gas Data



```
<Image width="40%" height="100vh" src={` /get_image?${data.time2}`} />
```

# Pre-Integration Testing

REQ-M-7: WEB subsystem runs as web server and is available over LAN:



REQ-M-4: Target identification should be capable of alerting GCS of targets type:



# Post-Integration Testing

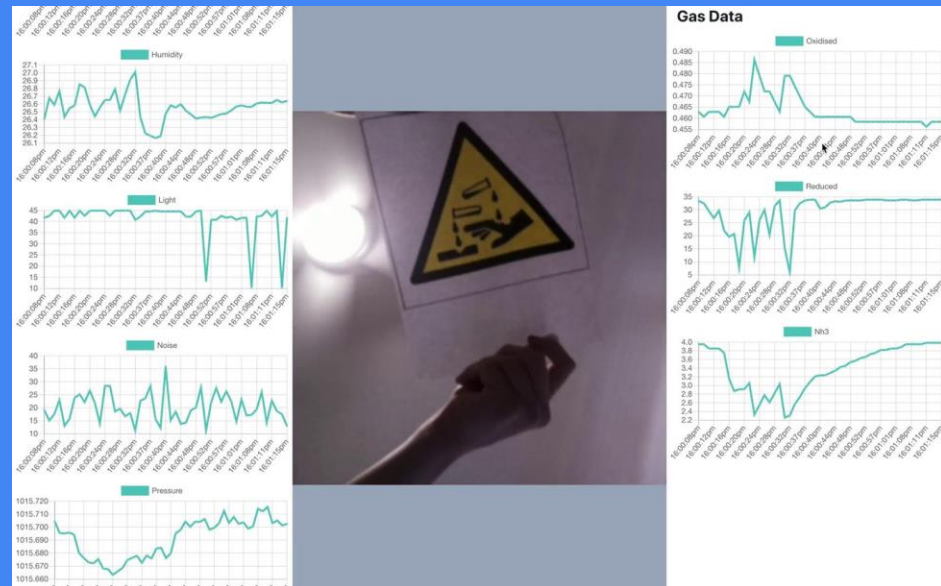
REQ-M-3: The UAVPayload<sup>TAQ</sup> shall communicate with a ground station computer to transmit video, target detection and air quality data:



REQ-M-5: The Web Interface is required to display real time air sampling data that is recorded directly from the UAVPayload<sup>TAQ</sup> and updated dynamically throughout the duration of the flight: Video

REQ-M-3: The Web Interface is required to display the images of the targets that are taken directly from the UAVPayload<sup>TAQ</sup> and updated every time a new picture is taken: Video

REQ-M-15: Live data from the UAV must be made available through the web server within 10 seconds of capture: Video





# WEB Subsystem Conclusion

# Verification and Validation Table

- The physical, functional and performance requirements were met.
- User requirements were partially met as enclosure had a design fault which complicated the mounting process to UAV.

Requirement	Description	Verification	Status
REQ – M – 01	Payload under 250g	Measurement	PASS
REQ – M – 02	Payload measures CO2, humidity etc	Demonstration	PASS
REQ – M – 03	Payload utilises ROS software as base control system	Inspection	PASS
REQ – M – 04	Sub-systems are compatible with ROS Kinetic.	Inspection	PASS
REQ – M – 05	Target identification alerts GCS of target type	Demonstration	PASS
REQ – M – 06	Web interface visualises all sensory system data	Demonstration	PASS
REQ – M – 07	Web interface visualises real time images of flight	Demonstration	PASS
REQ – M – 08	Web interface run as web server accessible to any devices on local	Inspection	PASS
REQ – M – 09	Preliminary designs completed by Week 7	Demonstration	PASS
REQ – M – 10	Additional purchase orders for payload placed before Week 7	Demonstration	PASS
REQ – M – 11	Developed solution conforms to systems engineering approach	Inspection	PASS (Enclosure design fault)
REQ – M – 12	System logged operation 10 mins prior to test	Demonstration	PASS
REQ – M – 13	Payload process all imagery on-board	Demonstration	PASS
REQ – M – 14	Processing must analyse all data from flight while UAV moves at max speed of 1m/s	Demonstration	PASS
REQ – M – 15	Processing must be able to analyse all data while UAV operates at altitude of 2m	Demonstration	PASS
REQ – M – 16	Live data from UAV must be available through web server within 10 seconds of capture	Demonstration	PASS

# Conclusions/Lessons Learned

- The project/customer requirements were met and even exceeded in some areas.
- However, improvements can be made to all respective areas of project to optimise results.
- The enclosure could have been more robustly and intuitively designed for the demonstration.
- The group performed well together as a collaborative unit.
- Every member was resourceful and incredibly reliable.
- Overall progress was achieved through meeting deadlines, great communication, frequent meetings and discussions.

Questions?