Detection and Identification of Chemical Warfare Agents and Explosives in Complex Matrices

Alexandra Harvey

PhD

University of York and Defence Science Technology Laboratory
Chemistry

March 2019

Abstract

This research explores the use of comprehensive gas chromatography (GCxGC) with different detectors for military applications. Two military applications are focused on in this thesis; firstly, the use of GCxGC time of flight mass spectrometry (TOF-MS) for the analysis of samples returned from operational environments and, secondly, for the analysis of operational samples in the field, exploring the possibility of a portable instrument.

Operational samples are highly complex and can contain organic matter such as blood, dirt, oils, diesel and other compounds that are found within that environment. Current methods require operators to send the sample back to the laboratory for analysis where the sample is cleaned using solid phase extraction (SPE) before analysis via analytical equipment. This research demonstrates that a sample containing explosive or chemical warfare agent (CWA) material can be analysed using GCxGC-TOF-MS without the need for SPE, allowing the sample to be untampered thus minimising the loss of sample as well as the possibility of contamination from other sources.

Similarly, in the field where a suspected deposited CWA or explosive is found, a sample is taken and analysed via ion mobility spectrometry or a spectroscopic technique. These techniques can detect the species of interest but struggle when environmental contamination is present which can lead to false alarms. This research demonstrates the use of a 6-port valve to perform GCxGC-FID analysis on explosives and CWAs in simulated operational samples and explores the possibility of creating a fully portable GCxGC system for use in the field.

List of Contents

List of Tables

List of Figures

List of Equations

Acknowledgements

Firstly, I would like to thank Dstl for sponsoring my PhD while I was working for them. Also, for all of those that helped me to learn how to write bids so that I could conduct the research. I would like to specifically thank the Chemical Sensing team for letting me use the laboratory space and the training they provided me with.

I am thankful to have had the support of my statistician guru Emily Matthews who helped me learn to code and improve my design of experiments as well as teaching me about machine learning. Also, for all the earl grey tea she supplied me with while I was analysing data.

Tim Ayers and others at the University of York were so welcoming and knowledgeable about engineering. I am extremely grateful for all of the help Tim provided me and for letting me meet Scotty!

I would like to thank my friends and family, specifically Jack, Phoebe, Sarah, Chris, Richard and Siobhan who supported me throughout with advice, food, walks and proof reading!

Finally, I would like to thank Agilent Technologies who have allowed me to finish my thesis while working for them. Without the education time provided to me it would have been much more difficult to achieve.

Declaration

I declare that this thesis is a presentation of original work and I am the sole author. This work has not previously been presented for an award at this, or any other, University. All sources are acknowledged as References. Where collaborative work has taken place, this is documented and referenced in the relevant chapter.

Chapter 1

Literature Review on Gas Chromatography and Comprehensive Gas Chromatography

1 Comprehensive Two Dimensional Gas Chromatography Introduction

1.1 Introduction

In this thesis comprehensive two dimensional gas chromatography (GCxGC) is explored for defence applications. Chapter 1 provides an overview of the history and theory of gas chromatography (GC) and GCxGC. This will provide a basis for understanding the analysis detailed in following chapters.

Chapter 2 details the history and use of explosives and chemical warfare agents (CWA), why there is an interest in detecting them before and after an event and the different ways of analytically detecting the species of interest. The chapter then explores the research undertaken in this thesis on the use of cryogenic GCxGC time of flight mass spectrometry (TOF-MS) to detect explosives and CWAs. This is, to the author's knowledge, the first comprehensive study on the detection of these chemicals in defence-relevant matrices using GCxGC-TOF-MS.

Chapter 3 explores the use of portable detection equipment to detect explosives and CWAs in both a military and civilian environment. The chapter then details the use of a "portable" GCxGC system that builds upon those developed at the University of York, to detect explosives and CWAs in defence relevant matrices. This provides the possibility of not only being able to analyse more complex samples in the field without the need for sending samples back to an analytical laboratory but also the possibility of creating a man portable system (a system that can be carried by one person and meets criteria set by the military) that could be used without the need for scientific personnel. This is, to the knowledge of the author, the first use of valve GCxGC to detect defence relevant compounds in complex matrices.

Chapter 4 begins by reviewing the use of man portable GCxGC across the literature before detailing the work undertaken to produce a prototype man portable GCxGC system. This research provides a possible approach to produce a system that could allow for detection of defence-related compounds in complex matrices in the field. This uses lessons learnt during chapter 2 and 3 to help produce a system that is related to defence requirements.

Chapter 5 details the conclusions and future research that could follow from this body of work.

Chapters 6 and 7 are held by the ministry of defence for security purposes.

1.2 Chromatography

Chromatography, as recognised today, began in the early 1900s and has developed into one of the most commonly used techniques in modern analytical chemistry.[1, 2] The first documented research related to modern chromatography was undertaken by Tswett where he focused on the separation of liquids using homemade columns filled with different absorbents.[2]

In 1941, Martin and Synge reported the theory of partition chromatography which in turn led to the development of the modern gas chromatography (GC) system in 1951 by Martin and James.[3, 4] Following this fundamental breakthrough, in 1952 Martin and Synge won the Nobel Prize for their work on the partition theory, as it is the basic theory behind all types of chromatography today from Thin Layer to GC.

Since 1952, chromatography has rapidly improved to become user friendly, more efficient and more effective. New column stationary phases are developed to improve separations and the understanding of chromatography has improved allowing new developments. However, the fundamental principles described by Martin and Synge have not changed and are discussed in detail in the following section.

1.2.1 Gas Chromatography

GC is a method used to separate gas phase molecules, the sample is first injected into a heated inlet and volatilises. A carrier gas (the mobile phase), typically nitrogen, helium or hydrogen, pushes the material from the injector and injects the material in a sharp band, onto a column. Columns can be packed or capillary: only capillary columns were used during this research. The column is a capillary tube with an inner coating known as the stationary phase, Figure 1.

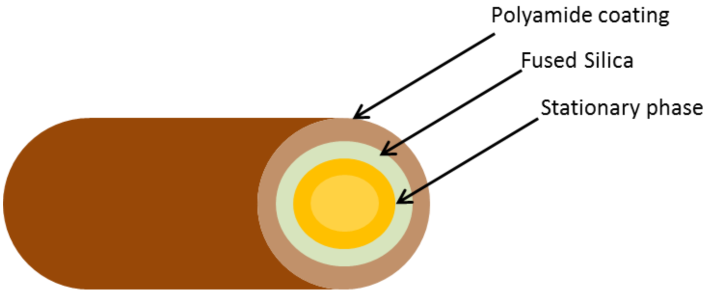


Figure 1 Cross section of a GC column showing the outer polyamide coating, the middle part is fused silica and the inner is the stationary phase.

Separation of the injected components is determined by the partitioning of each component between the mobile phase and the stationary phase (described by the partition ratio). In simple terms, if a compound has a high affinity for the stationary phase it will be retained and elute later than a compound with less affinity, thus they elute in time order of retention. The compounds then enter a detector, for example, a Flame Ionisation Detector (FID) or a Mass Spectrometer (MS). The detector allows for the detection and/or identification of the compounds.

1.2.1.1 Partition Ratio

The chromatographic separation can be defined as the baseline separation between two compounds. This can be described using the partition ratio defined as ‘the degree to which solutes are separated between the mobile and stationary phase’. To determine the partition ratio, we must view an equilibrium; if the solute species is A, we can define the equilibrium between the stationary and mobile phase by Equation 1.

$$A_{\mathrm{Mobile}} \overset{K}{\rightleftharpoons} A_{\mathrm{Stationary}} \qquad \textbf{Equation 1}$$

K is defined as the equilibrium constant between the two. K is the partition ratio which can be defined by Equation 2, where C_s and C_m are the molar analytical concentration of a solute in the stationary phase and mobile phase respectively.[5]

$$K = \frac{C_s}{C_m} \qquad \textbf{Equation 2}$$

In an ideal scenario the partition ratio is constant over a wide range of concentrations, thus making C_s and C_m directly proportional. However, the partition ratio is linked to several other physical factors; hence C_s and C_m are often not directly proportional. Figure 2 illustrates a simple chromatogram to aid the explanation of the following terms: t_r and t_m.

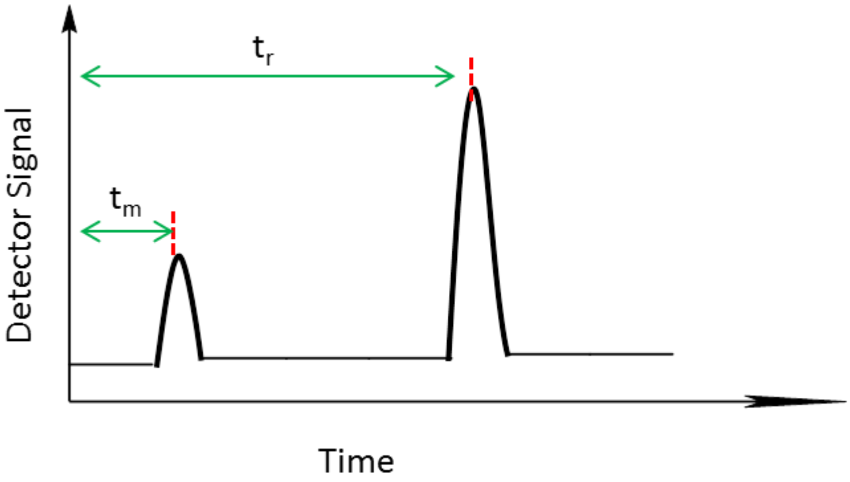


Figure 2 Example chromatogram with a non-retained and a retained species defined by t_m and t_r respectively.

In Figure 2, the peak on the left has not been retained by the stationary phase and thus elutes earlier than the larger peak on the right which interacts with the stationary phase (i.e. has a higher affinity for the stationary phase) and elutes later. The first peak is defined by t_m which is a measure of the average rate of migration of the mobile phase and can be described as dead time, i.e. time taken for an un-retained peak to travel through the column. The time required for a retained peak to reach the detector is known as t_r, otherwise known as the retention time.

The average linear rate of solute migration (V) can be defined using the length of the column (L) and the retention time (t_r) as shown in Equation 3.

$$V = \frac{L}{t_r}$$

Equation 3

The average linear velocity of the molecules in the mobile phase is described by μ, using t_m and the length of the column, Equation 4.

$$\mu = \frac{L}{t_m}$$

Equation 4

Combining the above, it is possible to relate the rate of migration of a solute to its partition ratio as a fraction of the velocity of the mobile phase, Equation 5.

$$V = \mu \text{ x fraction of the time solutes spends in the mobile phase}$$

Equation 5

The fraction described in Equation 5 can be explained in terms of the number of moles of solute in the mobile and stationary phase, Equation 6.

$$V = \mu \text{ x } \frac{\text{Moles of solute in the mobile phase}}{\text{Total number of moles in the solute}}$$

Equation 6

Therefore, if the total number of moles of the solute in the mobile phase is equal to the concentration (C_m) of the solute in that phase multiplied by the volume of the phase (V_m) and if the number of moles of solute in the stationary phase is described in a similar way (C_s x V_s) then Equation 7 allows a connection between the terms.

$$V = \mu \text{ x } \frac{C_mV_m}{C_mV_m + CsVs} = \mu \text{ x } \frac{1}{1 + C_sV_s/C_mV_m}$$ **Equation 7**

Consequently, the rate of solute migration (V) as a function of the partition ratio is given in Equation 8.

$$V = \mu \text{ x } \frac{1}{1 + KV_s/V_m}$$ **Equation 8**

1.2.1.2 Capacity Factor

The capacity factor is a measure of the retention of a peak. It considers the interaction of a compound with the stationary phase. The partition ratio can be combined with the capacity factor to describe the migration rates of solutes on different columns. For a solute A, the capacity factor (K_A') is defined by Equation 9.

$$K_A' = \frac{K_AV_s}{V_m}$$ **Equation 9**

Equation 9 can be substituted into Equation 8 assuming that K_A is equivalent to the partition ratio for A, giving Equation 10.

$$V = \mu \text{ x } \frac{1}{1 + K_A'} \qquad \textbf{Equation 10}$$

The capacity factor can be derived from a chromatogram using retention time (t_r) and the dead time (t_m), producing Equation 11.

$$K_A' = \frac{t_r - t_m}{t_m} \qquad \textbf{Equation 11}$$

The capacity factor is dependent upon temperature and the column packing. Thus, if a different capacity factor is required, it can be achieved by changing these variables.

1.2.1.3 Selectivity Factor

In order for separation to occur, two compounds must have different capacity factors. The selectivity factor (α) is the ratio between two compounds' capacity factors described in Equation 12.

$$\alpha = \frac{K'_B}{K'_A} \qquad \textbf{Equation 12}$$

1.2.1.4 Resolution

The resolution, Equation 13 is a measure of the degree of separation between two adjacent peaks taking into account the width of the peaks (W_h) and using the retention time (t_r). The value 1.18 in Equation 13 comes from the calculation of the peak widths. If we defined the peak width at the baseline of a Gaussian peak as 4 standard deviations, at the half height it would be 2.354 standard deviations. Therefore, the factor 1.18 is derived from (2 x 2.354/4).

$$R = 1.18\left(\frac{t_{r2} - t_{r1}}{W_{h1} + W_{h2}}\right) \qquad \textbf{Equation 13}$$

Peak separation and resolution are linked, as in some cases two broad peaks can have equal or better separation than two narrow peaks.

1.2.2 Rate Theory of Chromatography

The Rate Theory allows for a quantitative theoretical description of column efficiency. It considers the possibility of random movement by a molecule as it travels through a column. This theory is described in terms of plate height (H) and the number of theoretical plates (N). The terms are related by Equation 14.

$$N = \frac{L}{H}$$

Equation 14

As the plate height decreases and the number of plates increase, the column efficiency therefore increases.

The plate height can be found if we assume that each peak can be described as a Gaussian curve and that the width is described by the standard deviation (σ), Equation 15.

Equation 15

$$H = \frac{\sigma^2}{L}$$

Equation 15 allows for the efficiency of a column to be defined in terms of variance per unit length of column.

1.2.3 Van Deemter Curve

The average linear velocity and the column efficiency can be related by the Van Deemter equation. The Van Deemter, Equation 16, uses the following terms: height equivalent (h_{min}) to a theoretical plate, average linear velocity ($\bar{\mu}$), multi-path flow term (A), longitudinal diffusion term (B) and resistance to mass transfer term (C). Note that C in Equation 16 is defined as $C_m + C_{s.}$

$$h_{min} = A + \frac{B}{\bar{\mu}} + C\mu$$

Equation 16

If the efficiency is described by h_{min} and plotted against $\bar{\mu}$ for a given column and set of conditions then a Van Deemter plot is obtained, Figure 3.

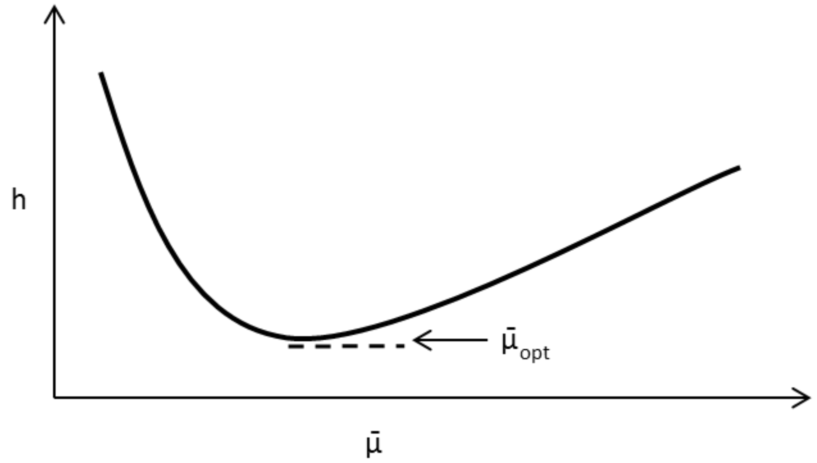


Figure 3 Van Deemter plot illustrating the optimum linear velocity where the efficiency of the column is at its greatest, defined by $\bar{\mu}$

The optimum linear velocity often relates to long GC run times. There are circumstances in which a loss of the column efficiency is outweighed by a large reduction in retention time.

The A term can be explained as eddy diffusion and arises due to movement of flow around particles in a packed column. In this work, no packed columns are used (only capillary), so the term needs to be modified. In a capillary column, there is no contribution from the A term to band broadening as there are no particles for the flow to move around. The equation that results from this modification is known as the Golay equation, Equation 17..

$$H = \frac{B}{\mu} + [C_m + C_s]\mu$$

Equation 17

Therefore, the only factors that affect plate height in a capillary column are diffusion coefficient, retention factor, the column dimensions and linear velocity.

The choice of carrier gas also affects the chromatography. If Equation 17 is plotted to produce a curve for different carrier gases, the best theoretical choice of carrier gas can be determined. Plots for helium, nitrogen and hydrogen are shown in Figure 4.

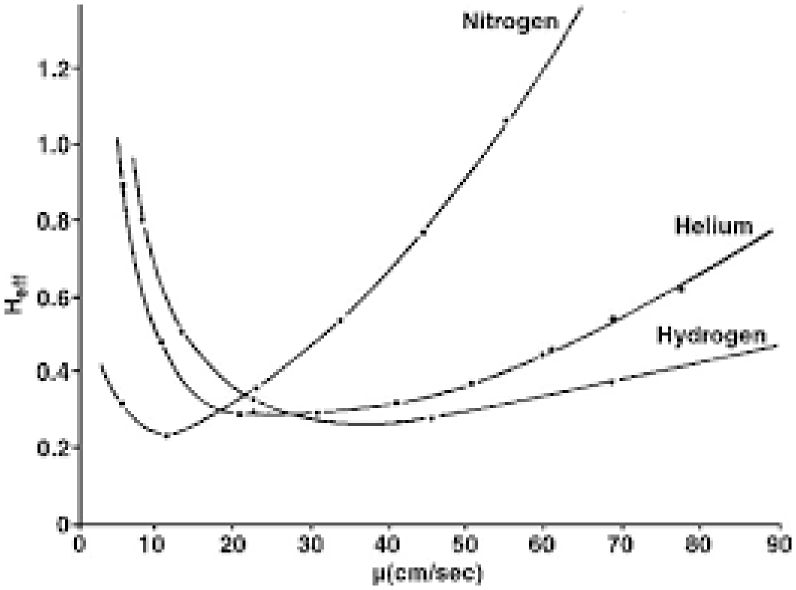


Figure 4 Typical Golay curves. Golay curves determine the best choice of carrier gas depending on the speed of the separation required. Image from Sigma aldrich[6]

Figure 4 illustrates the range in H for different possible carrier gases. If a fast method is required, theoretically hydrogen would be the best choice as it has the lowest H at high velocities. However, there are safety considerations to take into account when using hydrogen, which may outweigh the decrease in H.

1.3 Two Dimensional Gas Chromatography

1.3.1 Heart-cut separation

In 1958 Simmons et al. devised the first system that combined two or more capillary columns in a manner to allow fractions from the first column to be passed onto a second column for further chromatographic separation.[7] This is known as heart-cut separation and at the time allowed for separation that could not be achieved by one-dimensional chromatography.

Heart-cut separation involves taking small fractions from the first column and transferring this onto a secondary column using a switching valve, as shown in Figure 5. The switching valve most commonly used is a dean switch, which was invented in 1968, approximately 10 years after the first demonstration of heart-cut GC. This allows for fractionation of the primary column's separated analytes and then injection onto the second column.[8, 9] The secondary column will then separate out the fraction based on an orthogonal physical property; for example, the primary may separate out based on boiling point and the secondary based on polarity.

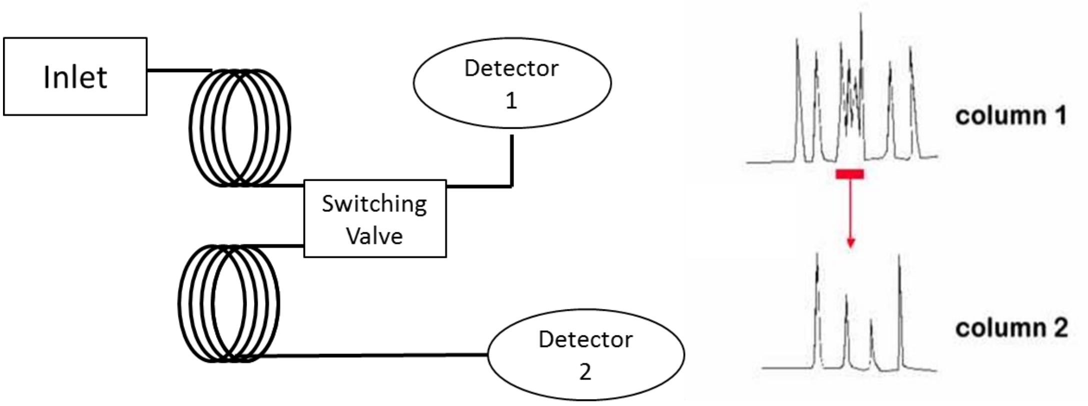


Figure 5 On the left is a simplistic schematic of how heart-cut separation using two columns can be undertaken, with the switching valve transferring the flow from the primary column to the secondary at a set time period. On the right is an example where there is a co-elution of peaks (highlighted by the red line) in the primary separation. This fraction (or time period) of the primary is then switched onto the secondary column where it is separated out to show the four peaks that have been deconvoluted.

A disadvantage with heart-cut separation is that it is time consuming, as only one fraction can be processed at a time. If there is an instance where several peaks are co-eluting, then several fractions would have to be switched onto the secondary column manually. This leads to partial analysis of the sample in the second dimension and an increase in analysis time. However, despite the disadvantage of slower analysis, the technique proved highly valuable in the analysis of petrochemical samples and is still used today for specific applications.[8, 10]

Naturally, the benefits of heart-cut GC, specifically in the petrochemical industry, led to the requirement for analysis to be continuous so that minimal information was lost. In 1991, Phillips et al. developed the first fully comprehensive GCxGC system.[11]

1.3.2 Comprehensive two-dimensional gas chromatography (GCxGC)

Comprehensive GCxGC was developed in 1991 by Phillips et al.[11] when the first fully comprehensive separation of an oil sample was completed. As the name suggests, for comprehensive GCxGC to be undertaken, the entire sample must be separated by the primary column and then the secondary column. The key component of a GCxGC instrument is the modulator.[12] The modulator fractionates the eluent from the primary column and injects these fractions onto the second column at regular time intervals (the modulation period).[13]

The secondary separation must be rapid, as the separation must be completed before the next modulation is performed. This separation is generally of the order of 2 – 10 seconds. The separation must be fast enough so that the modulation pulse has finished separating before the next pulse is injected. This helps to minimise the phenomenon called wrap-around but also ensures that sufficient modulations are performed across the first dimension to maintain the separation achieved on the primary column. Typically, a minimum of 4 modulations is performed across a typical primary column peak width to ensure minimal remixing within the modulator. Figure 6 demonstrates this as a schematic for comprehensive separation.

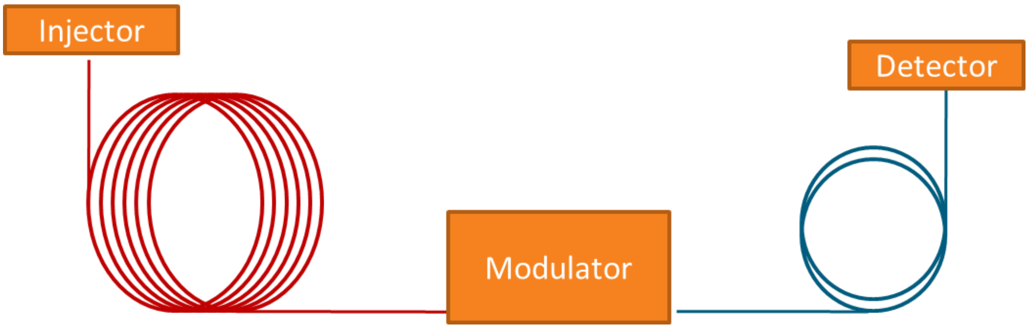


Figure 6 Schematic for comprehensive separation. The compounds are introduced by the injector and separated out in the primary column (red) before reaching the modulator at different points in time. The modulator splits the sample into small fractions and passes this onto the shorter secondary column (blue). The secondary column then completes its separation within 2 – 10 seconds (method dependent) whereupon the sample reaches the detector and the next fraction is then separated in the second dimension.

1.3.3 Peak Capacity

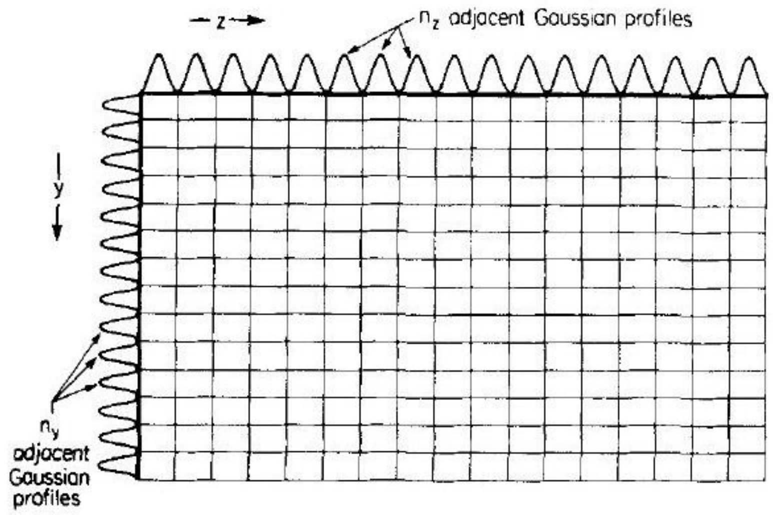
The peak capacity in one-dimensional GC is represented by η. If two columns are joined together the peak capacity is η_{tot} and is equivalent to the capacity of each column added together, Equation 18. J. Giddings described how to calculate the increase in resolution using the peak capacity.[14]

$$n_{tot} = n_1 + n_2 \qquad \textbf{Equation 18}$$

When observing GCxGC separations, Giddings determined that the peak capacities are multiplied thus allowing for a significant increase in the peak capacity, Equation 19.

$$n_{tot} = n_1 \text{ x } n_2 \qquad \textbf{Equation 19}$$

Giddings describes this pictorially in his paper which allows for a better visualisation of the increase in the peak capacity, Figure 7.[14] In heart-cut separation the image would differ to that observed in Figure 7 as it would not represent a square it would be one long line of cubes as the peak capacities are added together thus the peak capacity is lower and less space is available for separation.



z
n_z adjacent Gaussian profiles
y
n_y adjacent Gaussian profiles

Figure 7 The number of boxes represented in the image is equivalent to the product of the peak capacity of each column generated along two axes assuming adjacent Gaussian profiles. [14] This demonstrates a large separation space when using two columns in a comprehensive arrangement. Image from Giddings. [14]

1.3.4 Orthogonality

The use of GCxGC requires the two columns to be orthogonal to one another to achieve maximum separation. For orthogonality to be achieved, the separation mechanism in the second dimension must be independent of the primary, for example, the primary separation may be based on boiling point and the secondary based on polarity.

1.3.5 Data Analysis

In a typical GCxGC separation, a peak will be modulated into at least four separate modulated peaks to achieve good primary and secondary separation. Figure 8 provides a simple schematic of how the modulator fractionates the flow from the primary column and into the secondary allowing further separation and finally how this is then represented in a 2D contour plot in the software.

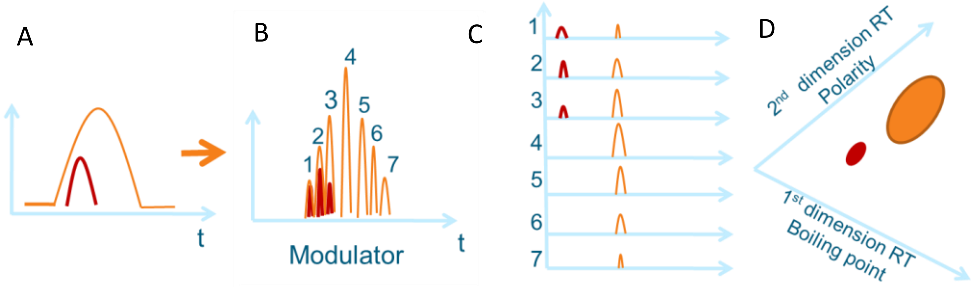


Figure 8 A: Co-eluting peak enters the modulator. B: The Modulator fractionates the flow. C: These are injected onto the second column at regular time intervals; second dimension separation is completed before the next fraction is injected. D: The 1D data is then displayed as 2D contour plot. Figure from R.Lidster.[15]

In Figure 8, the GCxGC software is able to build a 2D contour plot that can be made into a 3D plot when the volume of the peak is included. There are some issues associated with this analysis. Commercially available software works by creating a bounding box around the peak in two dimensions thus representing the peak as a rectangle and not as a gaussian peak. The integral is then worked out by using the height of the peak and the bounding box. This can lead to large errors in the volumes recorded but it is a simple approximation that allows for the calculation of the volume under the peak.

1.3.6 Modulator

The modulator can be described as the heart of the instrument. It allows for the trapping and releasing of compounds from the primary column onto the secondary allowing for comprehensive analysis to take place. Since the development of the first GCxGC, three strands of modulators have been developed; thermal, cryogenic and valve. These are discussed in further detail below.

1.3.6.1 Thermal Modulation

The first modulators for comprehensive GCxGC used dual stage thermal desorption.[9, 11] Phillips et al. developed the first system that utilised a metal capillary to retain the analytes.[12, 16] The metal was held at a constant voltage to produce heat and the system worked in two stages to retain and then desorb the analytes. More specifically, the modulation worked by having a temperature programmable box at the head of the second column this box remained at a temperature lower than that of the oven. In essence, the modulation could be described as fast thermal desorption.[12] The oven temperature is then used to allow for release onto the second column which could be held at a higher temperature.[9] Although the results from preliminary testing of these modulators appeared promising, the systems were not robust and often needed replacement. [17, 18] Geus at al. attempted to use a similar design as Phillips et al. but instead of using a constant voltage to provide current to heat the metal block, Geus et al. used copper wire coiled tightly round the outside of the modulator.[19] However, this had similar problems to Phillips et al.'s system, including the loss of volatile analytes.[13]

In an attempt to overcome some of the issues with the first design, Ledford and Phillips designed a moving modulator known as the sweeper.[20] The sweeper was the first thermal modulator that was commercially available; this used a separate heating element that was able to move across the capillary heating it at specific sites. A schematic of a sweeper is shown in Figure 9. The advantage of this system was that the heater block that controlled modulation was stable in temperature allowing efficient trapping.[13] However, the system allowed re-injection via desorbing at a large temperature differences so the column limited the applications that the system could be used for thus the application range of volatile compound analysis was limited.[13]

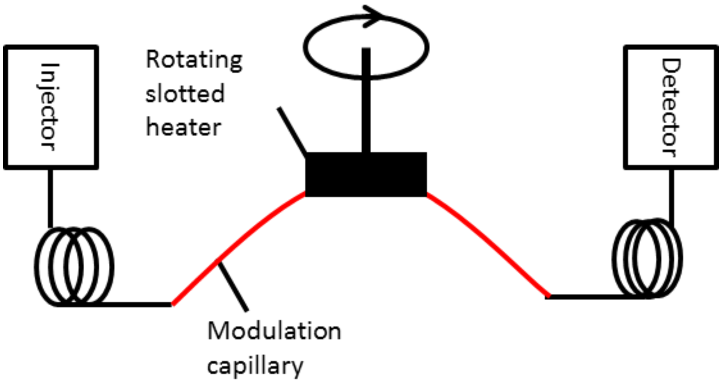


Figure 9 Schematic of a GCxGC using a sweeper. The rotating slotted heater allows for the modulation onto the secondary column which then reaches the detector.

1.3.6.2 Cryogenic Modulator

Marriott et al. advanced the GCxGC field and developed a cryogenic modulator.[21] This system still makes use of a moving modulator but instead of using heat, they utilised expanding liquid carbon dioxide (CO_2) to allow for analytes to be trapped at very low temperatures. This technique focused the analytes at the head of the second column and re-injection occurred by moving the modulator away from that site and allowing the oven to re-heat the column.[22] Although this system addressed some of the pitfalls of the traditional thermal modulators, the application range was still limited as there was breakthrough of the lower volatility analytes.[13]

Over the past few years, several systems have been designed to improve the cryogenic modulation either using liquid CO_2, expanding gaseous CO_2 or liquid nitrogen to trap the analytes.[23] Some of these systems have become commercially available and have reduced the number of moving parts by using hot and cold jets so that the systems are more robust. The liquid CO_2 or nitrogen is used to trap the analytes on the capillary and the cold jet is then switched off to allow injection onto the second column either by using the oven air to heat or more traditionally, a hot jet of air is pulsed onto the trapping region. These systems allow for highly sensitive detection and trapping of some volatile analytes but have some obvious drawbacks such as high consumable costs, freezing of water in samples, and not being portable.[24]

1.3.6.3 Valve Modulation

Using valve modulation means that the system can be made portable as it requires very few consumables and it is very small. The other advantage is that there is minimal breakthrough of the volatile species because using a valve allows the modulation to be independent of the trapping mechanism therefore limiting the breakthrough. Several research groups have used valve modulation for GCxGC, either using a rotary or diaphragm valve. Examples of each are explored briefly below.

Seely et al. first introduced diaphragm valve modulation in 2000.[25] Seely et al. used a 2-way 6-port diaphragm valve to allow for eluent to be modulated from the primary to the secondary column. Using this method allowed for removal of expensive consumables but had some limitations for sensitivity. The loss of sensitivity is due to the valve being held in the sampling position for only 80 % of the time thus 20 % of the eluent is flushed to waste. However, to maintain a refocusing capability similar to that of cryogenic modulation, the secondary column flow was at least 20 times higher than the primary thus making it unsuitable for use with a mass spectrometer unless the flow is split. [26]

Seely et al. also describe a total transfer modulator which uses a microfluidic dean switch to allow for modulation.[27] This can be housed outside the GC and is not temperature dependent but the switch limits the users modulation periods and flow ranges that can be used which can cause inconsistent transfer from the primary to the secondary column.[28] Agilent now sell a commercial microfluidic switch that can be incorporated into several systems.[29]

Lidster et al.[30] used a design based on Mohler et al.[31] and inserted a stopper into the diaphragm and rotary valves to prevent the primary column eluent from being vented to waste, allowing for 100 % mass transfer. This research explored the use of both valves for the analysis of volatile organic compounds (VOCs). The rotary valve did not perform as well as the diaphragm valve as it was less reproducible and was not as robust as the diaphragm.[30]

Until more recently, the use of a diaphragm valve has limited the temperature ranges and therefore the applications. However, Synovec et al. have described a high temperature diaphragm valve that can allow for separations to occur up to 325°C.[32] This valve uses Kalrez O-rings allowing for higher temperatures and therefore analysis of a greater range of volatile and semi-volatile materials. Synovec et al. demonstrated separation of vacuum pump oil and orange oil with high reproducibility using this valve thus extending the application range.[32]

Figure 10 demonstrates how a valve modulates the flow. The flow from column one fills a fixed sample loop. At a pre-defined time, determined by flows and the size of the sample loop, the flow will switch and use the auxiliary flow to push the sample in the loop onto the secondary column. This system does result in about 15 % of mass being lost and is a less sensitive technique than cryogenic modulation but it allows for portability of the technique to be achieved.

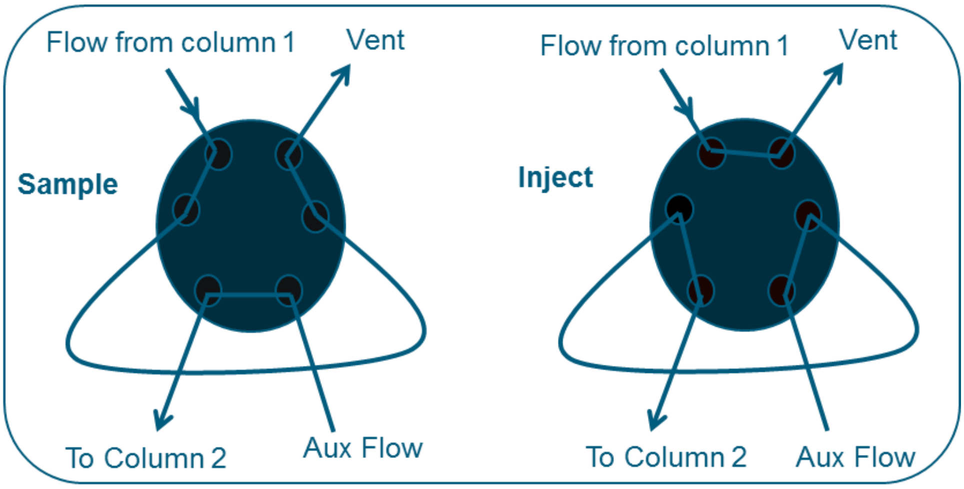


Figure 10 Schematic of how the valve modulation takes place, switching between flow from column one to pushing the fraction onto the secondary column. This takes place very rapidly, however, there is some mass lost during this process (ca. 15 %) leading to it being a less sensitive technique than cryogenic modulation.

The vent position can be filled with a stopper to stop the flow and improve the sensitivity such as in Lidster et al.’s and Mohler et al.’s research.

This chapter will conclude by reviewing the different detectors that GC and GCxGC can be combined with.

1.4 Detectors

Detectors can either provide information that allows for identification or detection. A detector that allows for identification is one that provides additional information about a sample, such as a mass spectrometer (MS) which provides a fragmentation pattern and a m/z allowing for identification. A detector that allows for detection is one that only provides a retention time based on elution from a column, such as a flame ionisation detector (FID) which is a non-selective detector that will provide a response when a material containing carbon is burnt and electrons are produce. Some detectors can provide selective detection such as a nitrogen chemiluminescent detector (NCD) which will only provide a response to a chemical containing nitrogen. The following sections 1.4.1, 1.4.2 and 1.4.3 detail the detectors used during this research.

1.4.1 Mass Spectrometry

Mass spectrometry (MS) is an analytical technique that allows for detection of chemicals based on their mass to charge ratio. In general, a compound will be introduced to a mass spectrometer and ionised. Different ionisation methods can be used depending on how much fragmentation of the molecule is desired. The ions are then sorted depending on their charge by different electronic and or magnetic plates and/or gates before hitting a detector plate. There are several different types of mass spectrometry only two of which we will focus on here; time of flight (TOF) MS and single quadrupole.

1.4.1.1 Quadrupole

A quadrupole, as the name suggests, is four parallel metal rods that are electrically charged in pairs such that opposite pairs have the same polarity (positive or negative polarity). This is achieved by the rods being connected to direct current (DC) and radio frequency (RF). When only the RF is applied to the rods, any m/z value ions can travel down the length of the rods and reach the detector plate. However, when a DC is applied as well, this allows for specification of the m/z ratio that the user wants to identify. Quadrupoles can be described as low sensitivity scanning detectors but do possess the ability to have some specification of mass to charge allowing for better signal to noise thus increasing the sensitivity of the technique. A schematic of a quadrupole is presented in Figure 11.

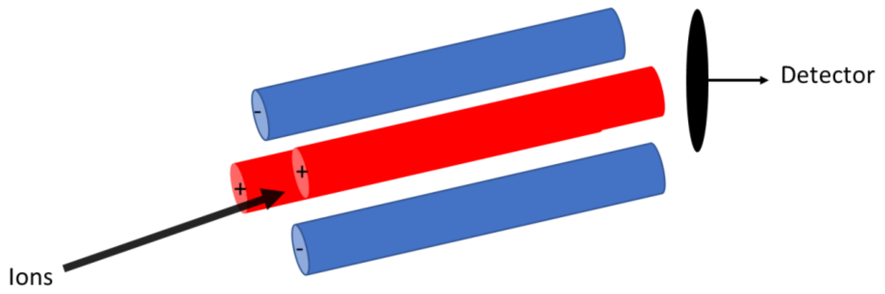


Figure 11 Schematic of a quadrupole

1.4.1.2 Time of Flight

A Time of Flight (TOF) MS has a limited dynamic range in comparison to other MS techniques but allows for detection of analytes in the low picogram to femtogram range. In a TOF the species of interest is ionised, and a pulse is applied that provides the ions with a specific kinetic energy. The ions then travel through a drift region where the ions are allowed to move based on their kinetic energy. This is followed by a reflectron which corrects the kinetic energy of ions of the same mass to charge ratio, the correction is needed as although in theory all ions of the same charge should be given the same kinetic energy by the pulse, they start at different potential energies and thus this effect has to be corrected for to avoid broad peaks. The reflectron correction achieves sharper and more gaussian peaks thus providing sensitive detection. A schematic of a TOF is provided in Figure 12.

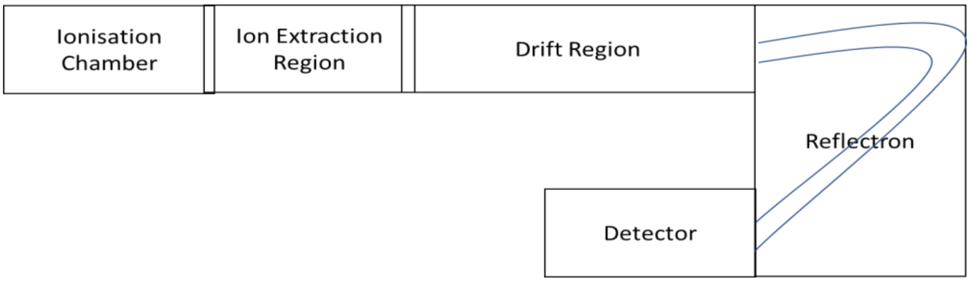


Figure 12 Schematic of TOF

1.4.2 FID

A flame ionisation detector (FID) is a universal detector, it will detect any species that contains a hydrocarbon element. Figure 13 shows a schematic of how a FID works. The compound (in gas phase) are separated by the GC column and reach the detector at different points in time depending on the separation method, the compound then meets the detector. The compound is burnt in a hydrogen flame and if it contains hydrocarbon species, these are ionised in the flame (at the anode). The ions then hit a conductive plate (the cathode) and that mechanism induces a current that can be measured and related back to concentration of the analyte.

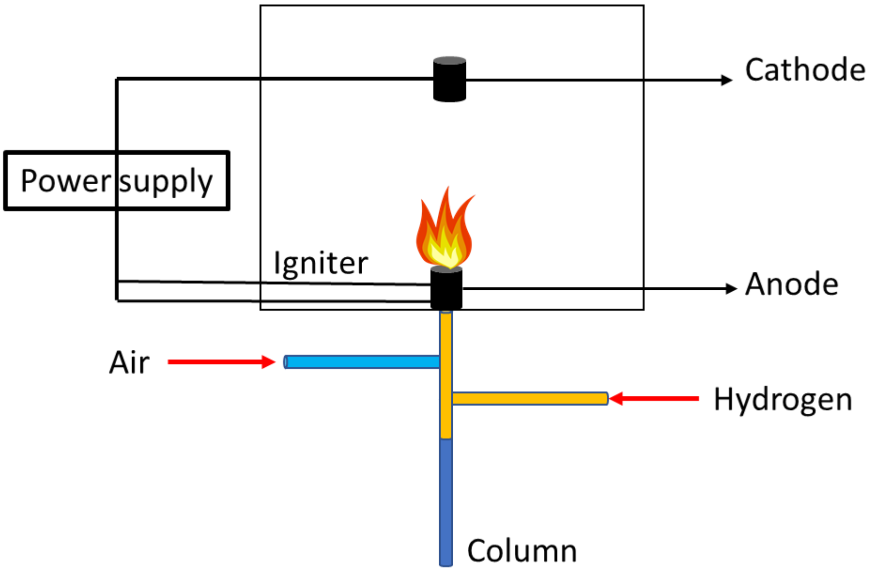


Figure 13 Schematic of a FID

1.4.3 NCD

Nitrogen chemiluminescent detector is a more selective detector than a FID that will only provide a response to a chemical that contains nitrogen. This is useful when observing a specific species in a complex matrix as the detection method allows for selectivity. An NCD works by burning the material presented to it by the GC column at a very high temperature to produce nitrous oxide (NO), this in turn is then reacted with ozone which produces a chemiluminescent signature (light signature) which can be detected by a photomultiplier. The response of the photomultiplier has a linear range which defines the dynamic range of the NCD.

Chapter 2

Comprehensive Gas Chromatography Time of Flight Mass Spectrometry for the Analysis of Threat Materials

2 Comprehensive Gas Chromatography Time of Flight Mass Spectrometry for the Analysis of Threat Materials

In the media today, we are faced with news articles of terrorists using explosives or chemical warfare agents (CWAs) to cause harm and disruption to society.[33-36] It is unsurprising that the use of explosives is commonplace in both military and civilian environments as they are becoming increasingly more available. Since terrorists can easily produce homemade explosives (HME) to cause disruption and, in areas of military conflict, it is not unusual to see them using HMEs with a military grade explosive. However, the use of CWAs is much more limited. It could be speculated that this is due to the agreement that use of CWAs cause widespread damage and potentially death and that their use is prohibited under the Organisation for the Prohibition of Chemical Weapons (OPCW) act.

Detection and identification of explosives and CWAs is highly important, not just in a military context but also for civilians.

In this chapter, the historical use and properties of explosives (2.1) and CWAs (2.2) will be explored. The chapter will then detail the method development and use of GCxGC-TOF-MS for detection of defence related compounds (2.4).

2.1 Explosives

2.1.1 Explosive History

Explosive use started back in the 15th century and the most common documented use is the use of black powder.[37] The use of black powder increased rapidly during industrial mining in the 1800s and since then, high explosives have started to be used for mining as they are easier to predict.[37] Explosives have also been commonplace in areas of conflict and in terrorist acts.

Terrorists frequently use improvised explosives devices (IEDs) to cause harm to civilians and military.[38] These are often of a simple construction using items that are easily purchased and accessed. However, more recently in areas of military conflict IEDs have begun to use components that are of military origin. The main component of an IED is the explosive, which can be formed from homemade explosives (HMEs) or from military explosive that has either been removed from its shell or stolen.[39] Components that form HMEs can be varied depending on what materials can be sourced locally making it very difficult to attribute them back to the source materials. However, the explosive used can be identified if it can be removed from the complex matrix it exists within.

In a civilian context IEDs are typically HMEs and ammonia nitrate fuel mixtures. Several events have taken place across the globe over the last 20 years. This includes organic peroxide attacks in: Finland in 2002, triacetone triperoxide (TATP) in London, Sweden and Denmark.[39] It should be noted that there have also been several occurrences of military explosives in IEDs, such as attacks by the IRA and the Lockerbie use of Semtex.

In terms of detection, this means that the components could be anything from acetone or peroxide to swimming pool cleaner and through to military explosives.[38] Steps are undertaken to attempt to stop the IED from causing harm in the first instance, which include gathering intelligence and searching at airports and venues.[40] Searches involves a number of stages, only one of which we will explore in this report – “trace search”. Trace search involves the use of portable detection equipment including fast GC, Ion Mobility Spectrometry (IMS), Differential Mobility Spectrometry

(DMS), Raman and IR spectroscopy along with detection dogs.[41-44] During a search operation, either the equipment is used to "sniff" areas and/or swabs are collected and taken back to the portable instrument for analysis. If an explosive is detected, a complementary technique will be used to confirm if it is present. However, if in the unfortunate case where there has been no prior knowledge that an attack is going to take place, an IED may function and cause harm.

If an IED functions it is the task, in the UK, for the forensic explosive laboratory (Dstl) to analyse the scene and identify the explosive that was used in the device.[45] This is required in order to determine how the device was constructed, i.e. what material was used, how it was made and to search for forensic evidence to link the IED to a suspect for the criminal justice system. In order to search for forensic evidence, the suspected trace explosive must be collected from the items in question. This can be done using swabs, a small vacuum system, solvent extraction or thermal desorption. Once the sample has been collected effective transfer of the explosive from sample medium has to take place, i.e. the sample must be "cleaned up" so that it can be easily analysed using analytical instrumentation. This process currently uses solid phase extraction to remove unwanted particulate.[46] However, this process is costly and can cause some mass loss of the analyte of interest. [46] This is explored further later in this report.

2.1.2 Explosive Properties

Understanding the properties of explosives helps to explain why they range in difficulty to detect. Here we will only explore vapour pressure, as it is the main property that affects detection for GC.

The vapour pressures of explosives is over orders of magnitude, for example highly volatile materials like peroxides to lower/nearly involatile materials such as inorganic salts.[47] Different papers supply different vapour pressures for certain explosives thus making it difficult to quote a pressure. The vapour pressure of explosives can vary depending on several properties including composition and temperature of the measurement. In the literature, there are reviews that demonstrate this variability in measurements and results. [43, 47, 48]

Direct vapour and particle detection has become the main focus for sampling and analysis, where the use of particle detection is more favourable due to the variable vapour pressures of explosives.[47] Additionally, if a terrorist does not want to be detected, they can contain the explosive within air tight containers making it difficult to detect the vapour.[39]

To give some perspective, trinitrotoluene (TNT) and ammonium nitrate (AN) have relatively low vapour pressures so the detection range for these needs to be around the ppb order of magnitude.[41] In comparison ethylene glycol dinitrate (EDGN), nitroglycerin (NG) and dinitrotoluene (DNT) have a high vapour pressure, so they are easier to detect in the vapour phase.[41, 47, 48] DNT is a common residue in TNT and as it has a higher vapour pressure it can be used as an indicator for further search.[48]   Materials such as cyclotrimethylenetrinitramine (RDX) and octogen (HMX) are extremely difficult to detect in the vapour phase due to low evaporation so detection of the particulate is generally favoured.[41]

For vapour detection, pre-concentration of the sample can be used to help achieve better sensitivity.[43, 49] Pre-concentrators work, by drawing in large volumes from the surroundings onto a filter to collect an increased volume of the organic and inorganic sampling material plus the suspected explosive.[49] The filter is then used as the sampling medium and the products are removed via solvent extraction or direct desorption to allow for sampling of the material.

2.2 Chemical Warfare Agents

2.2.1 CWA History

CWAs as we define them today, have been used since WWI where the Germans released chlorine gas in 1915 at Ypres.[50] The use of CWAs during WWI and the more limited use in WWII caused an estimated one million casualties from the release of approximately 124, 000 tonnes of CWA.[51] The research into the production and use of CWAs continued after WWII and it was only in 1997 that the Chemical Weapons Convention was drafted.[51-53] 197 countries are signatories to the chemical weapons convention that states that stockpile, manufacture or use of chemical weapons is prohibited.[54] The only permitted use of CWAs is for defence research. However, three countries are not signatories to the convention: Egypt, North Korea, and South Sudan.[55]

This has not stopped the use of CWAs over the last few decades and there have been several confirmed and unconfirmed uses of CWAs. For example, in 1991 during the Iran-Iraq war, Iraqi forces released sarin killing hundreds of people and in 1995 a terrorist group called Aum Shinrikyo released sarin in the underground system in Japan, killing seven people and injuring 500. [56-58] The most recent examples include; sarin release in 2013 in Syria, where approximately 1,300 people were killed and the use of the Novichok nerve agent in Salisbury in 2018.[56-58] These are but a few reported incidences of CWA use. There is still speculated use of CWA in Syria today making it a real threat to UK (and other) forces and Syrian civilians.[59]

The use of CWA in countries where UK troops are sent means that there is a requirement for portable detection equipment. This allows an early warning device in order to alert military forces to take protective measures, such as donning a respirator. However, there is also a requirement for the UK to act as an accredited laboratory under the OPCW and analyse samples from different countries in order to determine the agent used and the possibility that the sample could be attributed to its source, i.e. who manufactured or supplied the materials.

2.2.2 CWA Properties

CWAs can be classified in a number of different ways based on physical properties or, more commonly, by their medical effect.[60-62] The medical effects are listed below with examples of agents:

- Vesicant (blister agent) – Sulfur mustard and nitrogen mustard
- Choking agents –
- Blood agents – Hydrogen cyanide, phosgene
- Nerve agents – VX, VM and Sarin (GB)
- Irritants – 2-chlorobenzylidene malononitrile (CS) and dibenzoxazepine (CR) gas
- Incapacitants

However, it is the physical and chemical properties that will determine when and how an agent will be used and this can help to inform how it can be detected. In general CWAs are liquid and, in order for them to be effective, they must be disseminated effectively.[60, 61] The classifications above can be further broken down into persistent and non-persistent agents.

Non-persistent agents include sarin and some other G-series agents, which have a moderate to high vapour pressure so will vaporise easily on dissemination.[60] The agents can then form an airborne hazard (aerosol and vapour) which can deposit onto surfaces or stay in the atmosphere. If the environment is very cold, a different hazard state will be present.

Persistent agents have a low vapour pressure such as sulfur mustard and the V agents (VX and VM). These materials can be aerosolised for dispersion, otherwise they will be present in droplet form. These agents can be thickened with a polymer to help with dispersion and this can complicate detection.[63]

Solid agents do exist and in general will be dispersed as a fine power causing an aerosol and fine particulate hazard.[63]

In all cases the agents will cause disruption, harm to life and potentially death. Therefore, there is a need to be able to detect them before, during and after a possible attack. A forensic investigation would collect soil samples and swab surfaces such as clothing and skin. This presents a complicated matrix that must be separated from the analyte of interest to allow for detection and identification.

For both explosives and CWAs, detection in real world environments can be challenging for portable equipment and high-grade analytical equipment. The analytes of interest can be present in trace amounts in a large complex environmental matrix or they can saturate an area. The issue is that real world environments are a naturally complex matrix – for example, ambient air[64-66], water[67], and soil[68, 69] contain thousands of chemicals that can supress or interfere with the detection of an explosive or CWA. There are methods to allow detection in these instances, but they can be time consuming, costly and can have poor recovery rates. This research investigates the use of GCxGC to detect explosives and CWAs in complex matrices for defence investigations.

2.2.3 Detection

The most popular method for detecting CWAs since the 1960s has been GC.[70] However, there is no universal method to detect all CWAs and different techniques and methods need to be used to allow for detection and identification. Often, as described above, CWAs are found in a complex environmental mixture and expensive, time-consuming techniques are used to separate them. The University of Nebraska explored the use of GCxGC to separate a mixture of diesel and gasoline from a highly concentrated standard of sarin and soman.[70] The separation was successful but the chromatography is relatively ill defined and the separation needs to be improved.

Separately to this Gravett et al. used GCxGC to separate VX from soil following an accelerant-based fire and liquid decontamination. The authors claimed to be able to detect a break down product of VX but this was not a defined peak and wrapped around the chromatogram in the second dimension.[71]

There has been very limited research in the use of GCxGC for forensic defence applications. This work details the use of GCxGC to detect defence related chemicals starting with a cryogenic system and moving towards a portable valve system. The materials have been detected in matrices of relevance to defence and this is the first study to observe explosives in matrices and the first full study to represent the use of GCxGC in CWA detection.[72]

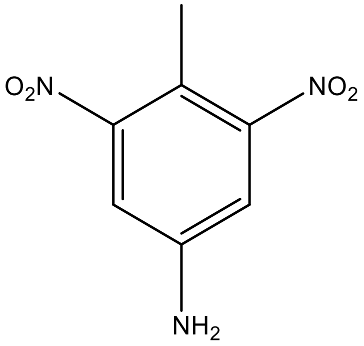
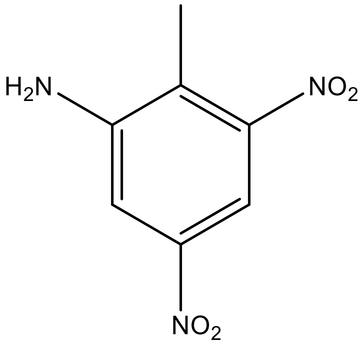
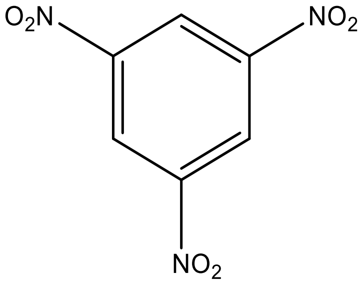
2.3 Experimental

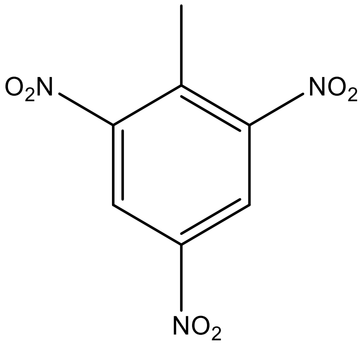
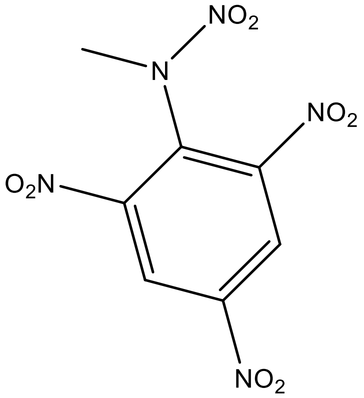
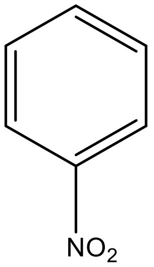
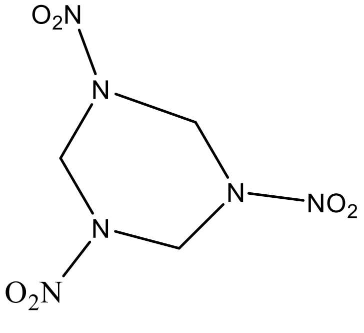
2.3.1 Explosives

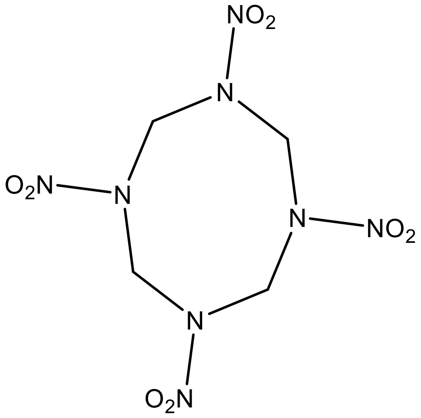
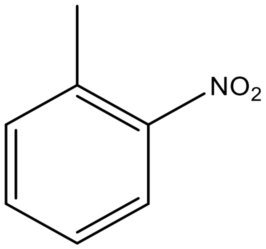
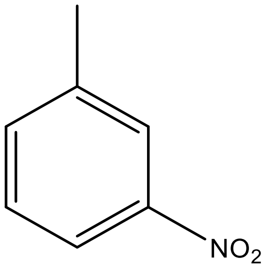
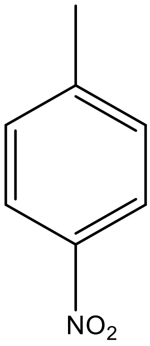
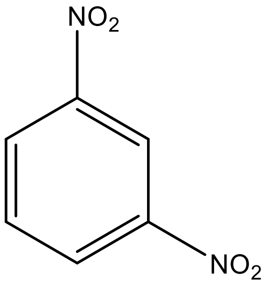
2.3.1.1 Sample Preparation

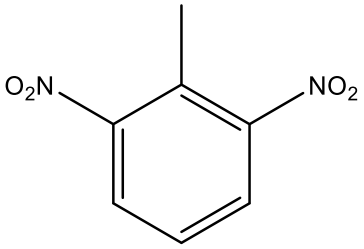
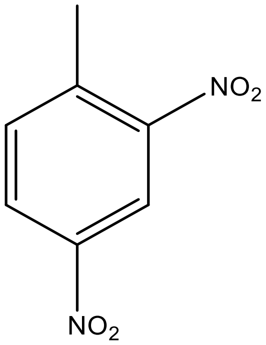
A 1000 ppm 8330 explosive standard (Restek) was diluted in ethyl acetate (Sigma Aldrich) to achieve a 10 ppm standard and a 1 ppm standard. Explosives present in 8330 standard can be seen in Table 1.

Table 1 8330 standard components

Explosive	Structure
4-amino-2,6-dinitrotoluene	
2-amino-4,6-dinitrotoluene	
1,3,5-trinitrobenzene	

TNT	
Tetryl	
Nitrobenzene	
RDX	

HMX	
2-nitrotoluene	
3-nitrotoluene	
4-nitrotoluene	
1,3-dinitrobenzene	

2,6-dinitrotoluene	
2,4-dinitrotoluene	

Diesel was provided by the University of York that had been purchased at a local garage and no sample preparation was undertaken. The 10 ppm standard was spiked into the solution to give a resulting 8330 standard concentration of 1 ppm.

Perfume – CK one - was purchased from a department store and was diluted by half with ethyl acetate. The 10 ppm standard was spiked into the solution to give a resulting 8330 standard concentration of 1 ppm.

An environmental organic aerosol sample in ethyl acetate, taken from North Kensington, London, was provided by the University of York. The 10 ppm standard was spiked into the solution to give a resulting 8330 standard concentration of 1 ppm.

2.3.1.2 GCxGC-TOF Parameters

GCxGC-TOF analysis was performed using a 6890B GC from Agilent Technologies fitted with a Leco thermal modulator system interfaced to an Agilent TOF-MS. The primary GC column was a BX5 (30 m x 0.25 mm x 0.25 µm). The secondary GC column was a BX50 (4 m x 0.2 mm x 0.2 µm). The primary oven was programmed from 40 °C (2 min) at 7 °C/min to 270 °C (5 min). The secondary oven was programmed from 70 °C (2 min) at 7 °C/min to 300 °C (5 min). A dual jet liquid nitrogen modulation system was used with a 5 second modulation period. Helium carrier gas was used at a flow rate of 1 mL/min. 1 µL of sample was injected using a Gerstal auto-sampler in splitless mode.

2.3.2 Chemical Warfare Agents

2.3.2.1 Sample Preparation

Two stock standards were produced for another study at Dstl Porton Down, from weapons grade (98 + %) CWAs and a small amount of the highly diluted material was used in this study. Standard 1 is a mixture of GA, GB, GD, GF, VX, and VM in isopropyl alcohol (IPA) (Optima Fisher) diluted to 10 µg/mL through to 0.2 µg/mL at even intervals. Standard 2 is a mixture of H, HN3 and T diluted in hexane (high purity Sigma Aldrich) to 10 µg/mL through to 0.2 µg/mL at even intervals.

2.3.2.2 Matrices

Replicate matrices to those found in operational samples have been produced. In some cases, the material has been acquired from military sources such as oil for vehicles or weapons, in other cases surfaces have been swabbed to replicate the samples that may return from operational environments. The swabbing procedure and list of matrices is detailed below.

Swabbing Procedure:

Use of lint free cotton swab soaked in IPA (Optima). The swab is used on no more than 10 surfaces and it is then extracted in 10 mL of IPA using a 30 minute extraction with agitation. The solvent is then vialed and the container and swab are disposed of after set a period of time.

Matrices:

- Hotel room
- Porton Down range aerosol sample
- Sahara Dust
- Diesel
- Aviation fuel (AV)
- Light weight oil (for weapon use) (LWO)
- Clothing that had been worn

2.3.2.3 GCxGC-TOFMS Method

GCxGC-TOFMS analysis was performed using a 7890A GC from Agilent Technologies fitted with a Zoex – Z1 thermal modulator system interfaced to a Markes International Time of Flight Mass Spectrometer. The primary GC column was a HP-5-MS (30 m x 0.25 mm x 0.25 µm). The secondary GC column was a BX50 (2 m x 0.1 mm x 0.1 µm). The primary oven was programmed from 60 °C (2 min) at 4 °C/min to 260 °C (2 min). The secondary oven was programmed from 80 °C (2 min) at 4 °C/min to 280 °C (5 min). A dual jet liquid nitrogen modulation system was used with a 3 second modulation period. Helium carrier gas was used at a flow rate of 1.4 mL/min. 1 µL of sample was injected using an Agilent PAL auto-sampler in splitless mode held at 280 °C.

When the matrices were injected, the inlet was used in split mode at 100:1 otherwise all parameters remained the same.

Different instrumentation has been used for the explosives and CWA analysis due to health and safety. CWA analysis can only be conducted at Porton Down.

2.3.2.4 GCxGC-FID Method

GCxGC-FID analysis was performed using a 7890A GC from Agilent Technologies fitted with a Zoex – Z1 thermal modulator system interfaced to a FID running at 200 Hz sampling collection. The primary GC column was a HP-5-MS (30 m x 0.25 mm x 0.25 µm). The secondary GC column was a BX50 (2 m x 0.1 mm x 0.1 µm). The primary oven was programmed from 50 °C (2 min) at 4 °C/min to 260 °C (5 min). The secondary oven was programmed from 70 °C (2 min) at 7 °C/min to 280 °C (5 min). A dual jet liquid nitrogen modulation system was used with a 5 second modulation period. Helium carrier gas was used at a flow rate of 1.4 mL/min. 1 µL of sample was injected using an Agilent PAL auto-sampler in splitless mode.

When the matrices were injected, the inlet was used in split mode at 100:1 otherwise all parameters remained the same.

2.4 Results and Discussion

2.4.1 Explosives

The contour plot from the GCxGC-TOFMS analysis of the 10 ppm standard of the 8330 standard explosive mix diluted in ethyl acetate is shown in Figure 14.

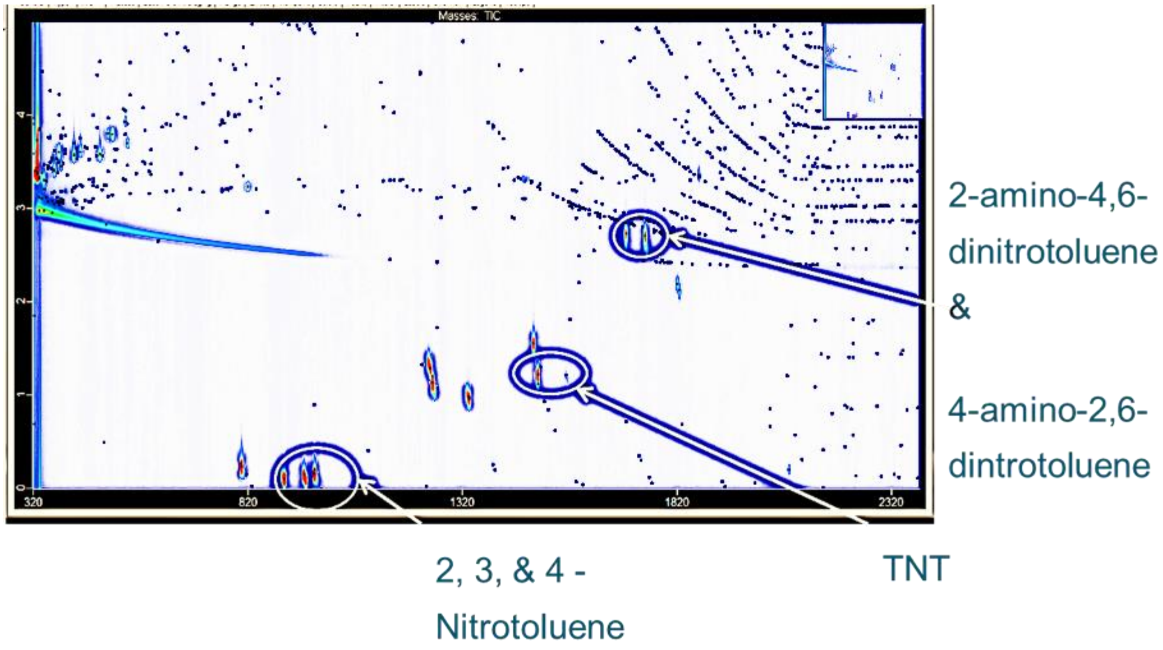


Figure 14 8330 standard with 6 of the explosives labelled, TNT, 2- nitrotoluene, 3-nitrotoluene, 4-nitrotolene, 4-amino-2,6-dinitrotoluene and 2-amino-4,6-dinitrotoluene for visual reference of separation. The x- axis shows the primary retention time in seconds and the y-axis shows the secondary retention time in seconds.

Figure 14 shows the separation of a standard explosive standard. The contour plot shows the GCxGC separation from the two columns used, with the non-polar column on the horizontal axis shows the elution order based on the boiling points. The retention on the vertical axis shows the elution based on the polarity. By using these two properties, volatility and polarity, to separate the standard, the isomers of DNT and nitrotoluene are separated out from one another.

RDX and HMX are relatively hard to detect using GC-MS due to their thermal instability and in this instance, HMX cannot be found but RDX has been tentatively identified using the mass spectra. Further method development would be required to enable us to detect these compounds; this could include decreasing column lengths, colder transfer lines and a higher column flow.

Limits of detection were explored using a 0.05 µg/mL standard and are listed in Table 2.*

Table 2 Limits of detection on column based on 0.05 µg/mL 8330 standard for explosives. n= 1, S/N of 3:1

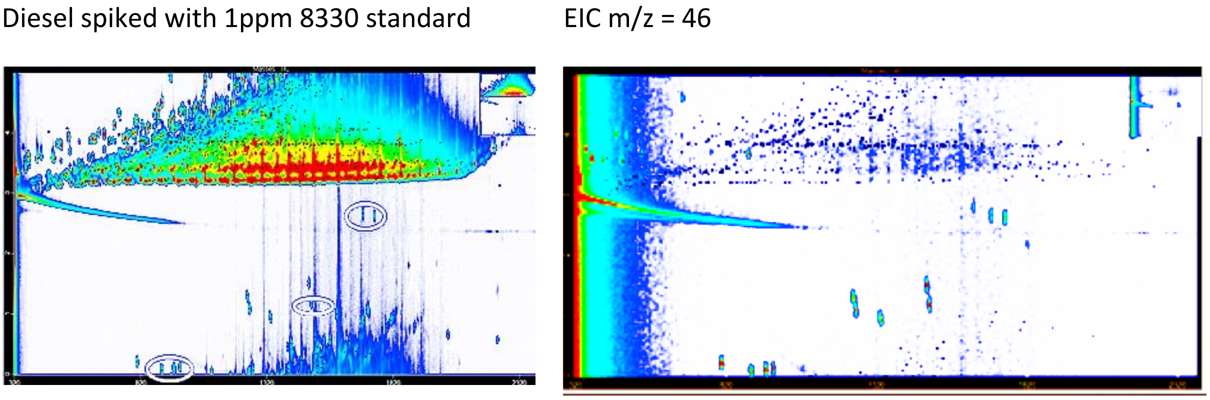
Explosive	**LoD based on S/N of 3:1 (µg/mL)**
4-amino-2,6-dinitrotoluene	0.083
2-amino-4,6-dinitrotoluene	0.073
1,3,5-trinitrobenzene	0.012
TNT	0.021
Tetryl	0.47
Nitrobenzene	0.24
RDX	n/a
HMX	n/a
2-nitrotoluene	0.0059
3-nitrotoluene	0.0081
4-nitrotoluene	0.006
1,3-dinitrobenzene	0.015
2,6-dinitrotoluene	n/a
2,4-dinitrotoluene	0.28

* LODs were calculated at 3:1 S/N linearly extrapolated from a 0.05 µg/mL standard.

2.4.2 Complex Mixture Analysis

When a person handles an explosive, material is transferred from them to the explosive (and vice versa), such as DNA and fingerprints. If the person, then touches other surfaces there is transfer of the explosive material to the surfaces. Transfer from the person to other objects can occur several times before there are undetectable amounts being transferred. Therefore, if an individual has been involved in the use or manufacturing of an explosive, they could have trace amounts of material present on their belongings or on themselves. This trace material is detectable, but it could be hidden in a complex matrix especially if they are trying to hide the material in the first place. Similarly, if an explosion has taken place, trace levels of the explosive could be present amongst the brick dust, body fluids, and debris. Essentially forensic samples from the field are highly unpredictable.

A small series of experiments were devised to observe if GCxGC is capable of separating out the explosives present in the 8330 standard from some common interferents, Figures 12-14.

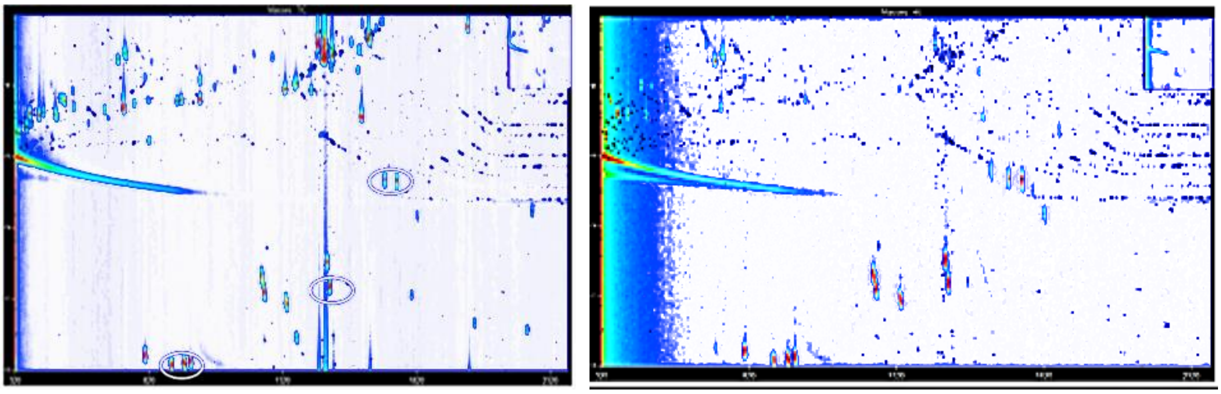


Diesel spiked with 1ppm 8330 standard

EIC m/z = 46

Figure 15 Diesel spiked with 1 ppm of the 8330 explosive standard. The left image shows the total ion chromatogram (TIC) and the image on the right shows the data with m/z of 46 extracted to highlight the presence of nitro groups in the sample. The white circles indicate the presence of the explosives defined in Figure 11. The x-axis shows the primary retention time and the y-axis shows the secondary retention time.

Figure 15 shows a common chromatogram of diesel with the separation of the alkanes, mono-aromatics up to the heavier tri-aromatics. The aim of this analysis was to separate the explosives from the diesel and it is clear from Figure 15 that they occupy a different separation space therefore they are easily detectable. The SIM ion of 46, a common ion for NO_2 containing species, was chosen to quickly view that explosives were being detected in a different separation space.

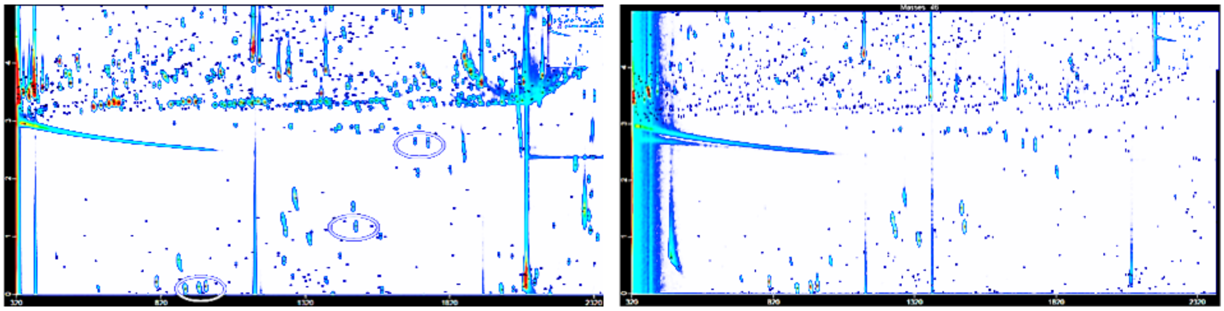


Perfume spiked with 1ppm 8330 standard

EIC m/z = 46

Figure 16 Perfume spiked with 1ppm of the 8330 explosive standard. The left image shows the total ion chromatogram (TIC) and the image on the right shows the data with m/z of 46 extracted to highlight the presence of nitro groups in the sample. The circled compounds are explosives of interest. The x- axis shows the primary retention time in seconds and the y-axis shows the secondary retention time in seconds.

Figure 16 is a good example of a cosmetic that may be present in an airport scenario or if a person is swabbed cosmetic material may also be present on the swab. A perfume contains volatile organic compounds (VOCs) of functionalised terpenoid compounds and muskes, which could cause false alarms in portable detection. Other functionalised fragrance compounds add to the complexity of the sample. In Figure 16 it is apparent that in most instances the explosives are occupying a different separation space to that of the VOCs.



London aerosol sample spiked with 1ppm 8330

EIC m/z = 46

Figure 17 London aerosol sample spiked with 1 ppm of the 8330 explosive standard. The left image shows the total ion chromatogram (TIC) and the image on the right shows the data with m/z of 46 extracted to highlight the presence of nitro groups in the sample. The x- axis shows the primary retention time in seconds and the y-axis shows the secondary retention time in seconds.

Secondary organic aerosols are formed though complex reactions of anthropogenic and biogenic VOCs with light and atmospheric oxidants which include radicals of OH and Cl, and ozone.[73] These reactions can vary at different temperatures and humidity. These particulates are typically less than 2.5 µm in size and can have adverse effects on human health owing to their ability to penetrate deep into the lung alveoli.[74] It is possible due to the low volatility of some explosives that they could exist in an aerosolised form post explosion; therefore it could become important to detect them in a complex matrix such as an aerosol sample.[75, 76] If released into the environment in an aerosolised form they could form complex particulates and/or aerosol with atmospheric oxidants present making them difficult to detect without some form of extraction. In Figure 17 it is clear that the explosives occupy a different separation space to that of the aerosol sample and that they are easy to detect at the low levels required for explosive exploitation.

It should be noted that there can be a “natural” background of explosives at some sites, therefore in forensic analysis of a scene it might be, for example, that above 5 ng detected is significant. The current UKAS accredited method for cleaning matrices that contains explosives is solid phase extraction (SPE). With some explosive materials, the stationary phase used in SPE can chemically interact with the explosive and decrease the amount of evidential material that is left in the sample. This is known and documented so that it can be taken into account.[77] The method described above using GCxGC-TOFMS allows for detection directly from the matrix with limited sample preparation thus it could improve evidence-based cases for detection of explosives if it is further validated and UKAS approved.

2.4.3 Chemical Warfare Agent Analysis

2.4.3.1 Limit of detection and Limit of Quantification using TOFMS

Standards were made as detailed in section 2.3.2 and were analysed using the method detailed in section 2.3.2.3. Table 3 details the limits of detection and quantification using the student T test. For each agent a 6-point calibration was undertaken and the data for this can be found in Annex 1.

Table 3 Limit of Detection and Limit of Quantification in ug/mL using student T value (N=3 for G & V and N=5 for mustard family)

		$RT_{1\,(min)}$, $RT_{2(sec)}$	LOD (ug/mL)	LOQ (ug/mL)	R^2
Compounds	GB	7.65, 2.5	0.026	0.085	0.9884
	GD1	13.80, 0.5	0.012	0.040	0.9459
	GD2	13.95, 1.0	0.039	0.131	0.9698
	GA	16.75, 2.02	0.033	0.109	0.8412
	GF	19.40, 2.5	0.005	0.016	0.9966
	VM	31.45, 0.0	0.028	0.093	0.9786
	VX	34.65, 0.0	0.010	0.034	0.9704
	H	18.40, 0.5	0.076	0.254	0.9979
	HN3	26.00, 0.5	0.045	0.150	0.9911
	T	41.60, 0.5	0.146	0.488	0.9965

As can be observed in Table 3 the limits of detection and quantification are in the sub ug/mL range, which is useful if low levels of agent detection are required, or if the main agent has started to degrade and low levels are left in the sample.

One of the aims of this research was to determine if, with little clean up, can the instrument detect CWAs in complex matrices. In order to test this, matrices of common interferents and replica samples that might return from operation were made (section 2.3.2.2) and spiked with chemical agents. The following details the results from the analysis.

2.4.3.2 CWA Detection in Complex Matrices

When a chemical agent attack has happened, depending on the agent and the method of attack, the agent will still be present within that environment. It is therefore important that samples are collected to determine what agent was used and if any evidence of whom co-ordinated the attack can be gathered.

A series of experiments were undertaken to determine if GCxGC-TOFMS could analyse samples from replicate environments with minimal to no sample cleaning. Table 4 details the results of what agents could be detected in the matrices tested. The chromatograms for the matrices can be found in Annex 1.

Table 4 Table detailing matrices spiked with chemical agent. Green indicates that the agent could be detected and red indicates that it could not be detected. If an extracted ion chromatogram was used a (*) is next to the tick.

		Agent						
		GB	GD1	GD2	GA	GF	VM	VX
Matrix	Aerosol sample 3	✓	✓	✓	✓	✓	✓	✓
	Clothing	✓	✓	✓	✓	✓	✓	✓
	Inside Oven	✓	✓	✓	✓	✓	✓	✓
	Hotel	✓	✓	✓	✓	✓	✓	✓
	Gasoline	✓	✓ *	✓ *	✓ *	✓ *	✓	✓
	LWO	×	✓	✓	✓	✓	✓	✓
	AV Fuel	✓ *	✓ *	✓ *	✓ *	✓ *	✓ *	✓ *
	Diesel	✓ *	✓ *	✓ *	✓ *	✓ *	✓ *	✓ *
		* using EIC						

The results in Table 4 indicate that almost every agent tested could be detected in a number of complex matrices without any sample preparation. Each of the matrices listed in the table represents a possible operational environment. For example, the AV fuel (aviation fuel) is from military samples at Porton Down. AV fuel contains a wide array of compounds from simple alkanes through to heavier aromatic species. These hydrocarbon species occupy the same separation space in one dimensional gas chromatography making it hard to deconvolute the resulting mass spectrums to identify the CWA present. Often clean-up of the sample is undertaken but if an unknown CWA is present it could easily be removed from the sample if the incorrect technique is used. This then requires a costly process using different clean-up methods and different analytical techniques to remove the matrix from the sample. In this research it has been demonstrated that the agent can easily be identified from the matrix, thus allowing a quick screening tool and with further development it should allow for an evidential tool.

Figure 18 provides an example chromatogram for reference. This research provides evidence that GCxGC-TOFMS can be used not only to detect low quantities of agent but also that agents can be detected in complex matrices with limited issues. The methods developed here can now be taken further and used to test real operational samples for orthogonal identification or as a screening tool for unknown samples.

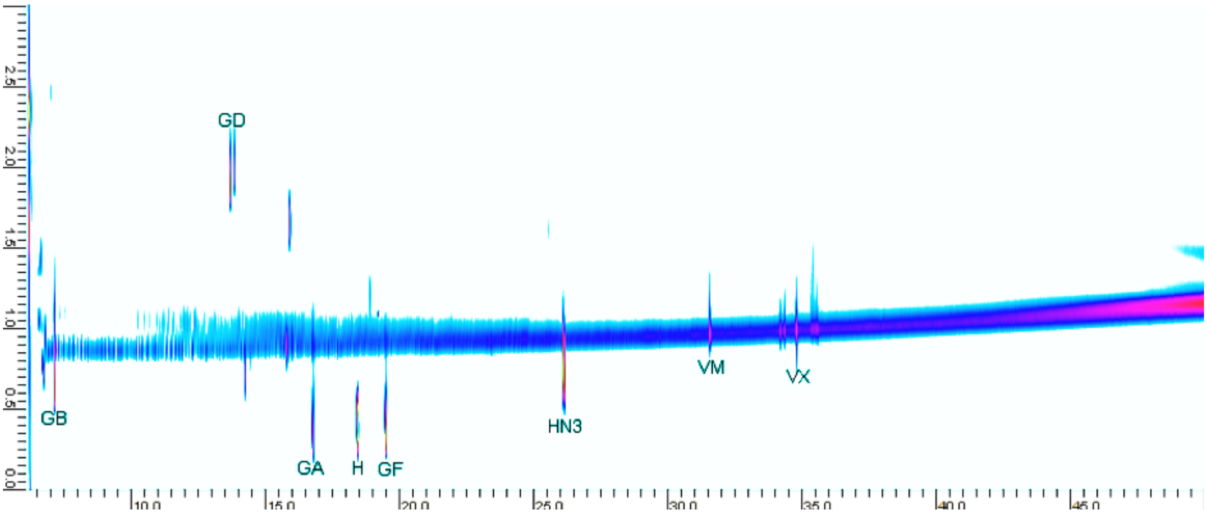


Figure 18 CWAs detected using GCxGC-TOFMS at 1 µg/mL. The agents are labelled on the spectrum and demonstrate that each agent occupies a different separation space in both the primary and secondary dimension. The GD peaks are also separated.

In Figure 18 there is 1D separation of all CWAs and some 2D separation can be observed, such as the two peaks for GD appearing at 2 seconds and the peak for GB at 1 second on the secondary axis.

There is a continuous column bleed across the chromatogram, which can make detection of any compound whose retention time appears in the area difficult. However, as this technique is time of flight mass spectrometry this can be easily corrected for using extracted ion chromatograms.

2.4.3.3 Conclusions of TOFMS Data

GCxGC-TOFMS has proven to be a highly effective tool for the analysis of both explosives and CWAs in complex matrices. The methods developed here can be further validated with real operational samples providing not only a screening tool but also orthogonal identification, which is often required.

The main issue with the instrumentation is the size. If the instrument could be made portable several more defence applications could be explored. In order to start exploring the possibility of a smaller system, the same experiments from the sections above were undertaken using a FID instead of a TOF to determine what differences in analysis would arise.

2.4.3.4 Limit of detection and Limit of Quantification using FID

As a cryogenic GCxGC-TOFMS performs excellently at detecting both explosives and CWAs in matrices of interest, it is of military interest to determine if a portable system could be made and used to detect these materials out in the field. This could provide evidential information in the field or it could act as a warning system that materials are present at a site or on a person. Further to this, the technique is classed as an orthogonal technique of high dimensionality, so it could provide information that could be of importance to defence intelligence.

A further study has taken place replacing the TOF-MS with a FID as it is a more portable detector and this provides the evidence required to move forwards to produce a fully portable system. Although it has no ability to identify, each compound has been run individually to build a retention time library for further analysis. Limit of detection (LOD) and limit of quantification (LOQ) have been studied for CWAs and are listed in Table 5. The LOD has been calculated using n=4 (nerve agent family) and n=5 (mustard family) to provide a T value multiplied by the standard deviation at the 97.5 % confidence interval. The LOQ has been calculated using 10 multiplied by the standard deviation. The values listed are expressed in µg/mL and all calibration data can be observed in Annex 1.

Table 5 Limit of Detection and Limit of Quantification in ug/mL using student T value (n=4 for G & V and n=5 for mustard family)

		$RT_{1\ (min)}$, $RT_{2(sec)}$	LOD (ug/mL)	LOQ (ug/mL)	R^2
Compounds	GB	9.91, 1.28	0.59	0.60	0.9957
	GD1	17.08, 2.45	0.59	0.63	0.9952
	GD2	17.17, 2.52	0.47	1.31	0.9987
	GA	20.25, 3.75	0.41	0.56	0.9895
	GF	23.00, 3.72	0.40	0.63	0.9783
	VM	32.25, 4.08	0.40	0.71	0.9799
	VX	38.50, 3.94	0.33	0.68	0.9540
	H	21.90, 3.97	0.70	3.13	0.9904
	HN3	29.75, 4.02	0.18	1.50	0.9929
	T	45.50, 1.04	0.75	0.87	0.9816

The LOD and LOQ in Table 5 are higher than that of the TOF, which is to be expected as FID is a less sensitive technique, but it has a larger dynamic range. This is beneficial as occasionally there is no option to pre-screen an unknown sample and therefore a detector that can take a larger concentration is of benefit and it is easier to clean from contamination. The other benefit of a FID, as mentioned before is that is it moving towards a portable system. However, the obvious disadvantage here is the loss of sensitivity.

2.4.3.5 CWA Matrix Analysis

The possible uses of GCxGC in defence include providing evidential value for samples of significant interest for defence and that further information can be gathered that might provide intelligence.

In this study, the use of GCxGC in the field for defence applications is tested. The possibility of being able to screen in the field not only with standard portable detectors but also GCxGC, will act to separate the sample from the matrix, increases the defence analysis capability in the field significantly.

Matrices were replicated to those that may be found during a military operation such as; a hotel room, oil spills, clothing from a person or aerosol samples. These were then spiked with CWA and the results for detection are detailed in Table 6.

Table 6 Agents detected in replicate matrices of operational environments. Green shows that the material is detected and red shows that it was not detected.

	Agent								
Matrix	GB	GD	GD	GA	H	GF	VM	VX	T
Diesel	✓	✓	✓	✓	✓	✓	✗	✗	✓
Gasoline	✓	✓	✓	✓	✓	✓	✓	✓	✓
Clothing	✓	✓	✓	✓	✓	✓	✓	✓	✓
Oven	✓	✓	✓	✓	✓	✓	✓	✓	✓
Aviation fuel	✓	✓	✓	✓	✓	✓	✗	✗	✓
Aerosol sample	✓	✓	✓	✓	✓	✓	✓	✓	✓
Hotel 2	✓	✓	✓	✓	✓	✓	✓	✓	✓
Light Weight Oil	✓	✓	✓	✓	✓	✓	✓	✓	✓

As can be observed in Table 6 almost all agents can be detected in each matrix when using FID. These peaks have been found using retention time and their visibility in the chromatogram. A few example chromatograms are shown in Figure 19 and Figure 20, all other figures can be found in Annex IV.

The possible reasons for why VX and VM cannot be easily detected in diesel or AV gas are that they contain heavy hydrocarbon fractions that co-elute with the agents of interest. The agents nor the hydrocarbons could be easily moved away from one another, as there is no separation space left in the chromatogram. Another method or a non-universal detector would be required for detection of these compounds in this instance.

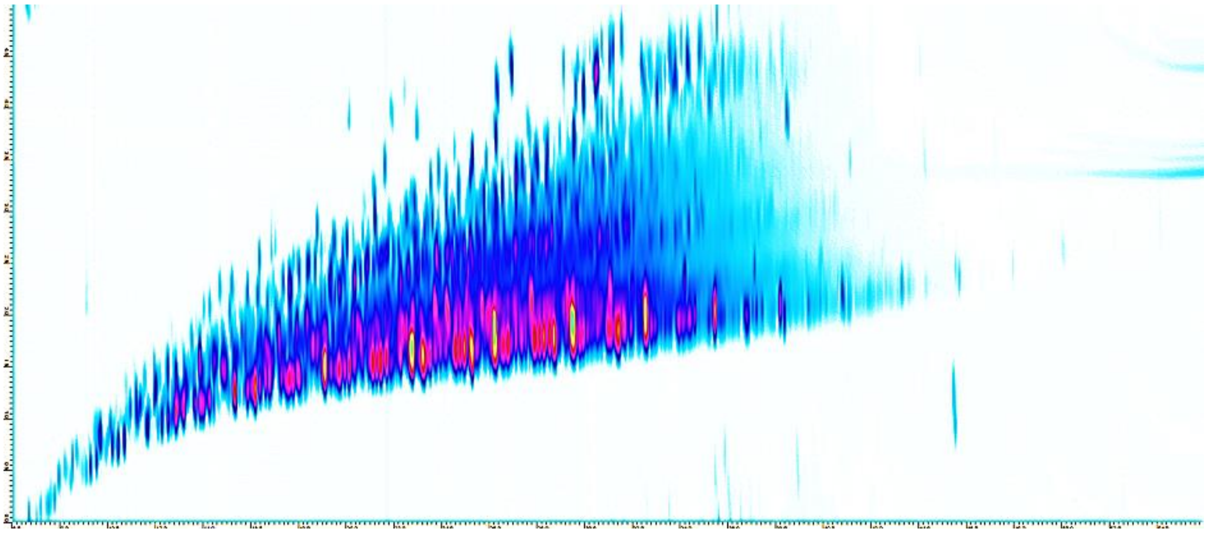


Figure 19 Aviation Fuel spiked with CWA. The spectrum is highly complex and the aviation fuel has aromatic peaks as well as hydrocarbons present. The agents can be detected either as they occupy a different separation space or by zooming into specific regions and relying on the two retention times for a location, like a map. The x- axis shows the primary retention time in seconds and the y-axis shows the secondary retention time in seconds.

In Figure 19 it can be observed that the aviation fuel contains several different types of hydrocarbon species. Starting with the alkanes towards the bottom of the chromatogram and as the time increases in the second dimension, the size of the hydrocarbon species to aromatic compounds also increases. This chromatogram can be considered as a fingerprint for this type of aviation fuel as it will be different depending on how the fuel has been treated and its age. This can also be used as evidential value for an investigation.

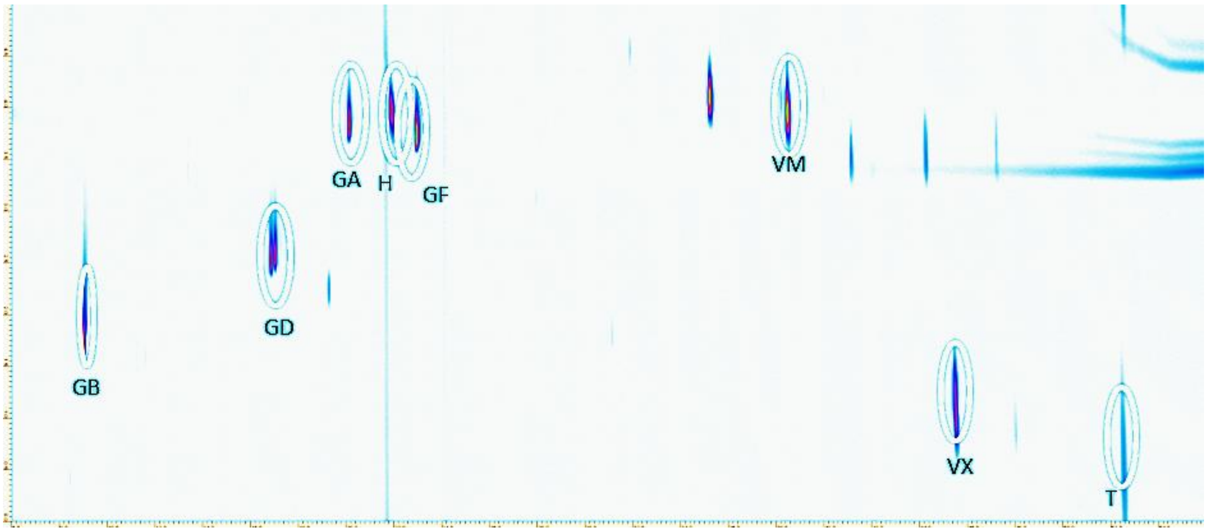


Figure 20 A swab from clothing spiked with CWA. The CWAs detected are highlighted on the chromatogram in white circles and labelled. The x- axis shows the primary retention time in seconds and the y-axis shows the secondary retention time in seconds.

Figure 20 depicts a chromatogram from a swab that has been used to remove material from clothing. It is much less complicated than the chromatogram observed in Figure 19 but there is some material present that could act to mask or suppress a material of interest. In this instance all of the agents were easily detected.

Figure 19 and Figure 20 provide examples of the detection space in the matrices. It is apparent that almost every agent of interest can be detected in the matrices. The clear disadvantage of moving to FID from a TOF is the loss in sensitivity as detailed in Table 5. However, this disadvantage is a small price to pay if portability is desired. The following chapters explore the development of a portable system for defence applications.

Chapter 3

Valve Modulation Comprehensive Gas Chromatography Flame Ionisation Detection for the Analysis of Threat Materials

3 Valve Modulation Comprehensive Gas Chromatography Flame Ionisation Detection for the Analysis of Threat Materials

3.1.1 Portable Detection Equipment

Portable detection equipment is used by military and civilians to detect the presence of explosives or CWAs.[40, 49, 78-80] The threats related to the use of explosives and CWAs have been increasing recently, thus giving rise to further research and development within the field of portable equipment.[80] The current in service detection equipment meets the criteria set and uses a number of different detectors to identify compounds of interest. The requirements are of a high standard and change continuously due to changes in threat and the need to detect at lower concentrations, thus warranting the need for further research. Requirements include; that the detector must analyse the sample quickly, the equipment should be light weight, have low power requirements, have a user friendly interface, have low limits of detection (LOD), utilise a simple method for sample collection and analysis (for example a swab) and detecting the compounds of interest with minimal interference from other compounds present in the environment.[81]

3.1.1.1 Common Detectors

Raman and infrared (IR) are common types of detectors that would be seen in both a military and civilian detection suite. Both Raman and IR are a form of bulk functional group detection, meaning that they search for a specific set of functional groups that would relate to the compound of interest. They allow for fast detection and in some case can provide a form of standoff detection (at a distance) or through barrier detection (through materials e.g. plastic bottles or envelopes) which can be useful if minimal sample contact is desired.[82, 83] Minimal sample contact is highly important for CWA analysis due to their high toxicity. However, these techniques have their disadvantages, as there are some compounds that they cannot analyse because the functional groups are not IR and/or Raman active and in terms of standoff detection, interaction with the open air can cause interference.[84] There are other considerations when detecting explosives, as attenuated total reflection IR would not be suitable for shock sensitive explosives due to the need for sample contact.

Ion mobility spectrometry (IMS) is the most popular portable detector as it is a small, cost effective, robust and a low powered instrument.[42, 80] It works by forming reactant ions in air, typically using Ni^{63} or corona discharge, the ions then meet the vapour sample presented and form ion clusters. These clusters pass through into the IMS drift cell and each cluster travels through the cell at different speeds depending on their size and shape. Essentially, the smaller and more compact the cluster, the faster it will travel down the drift cell and the larger the compound the longer it will take. This has some obvious disadvantages: several other clusters could have a similar size and shape to that of the compound of interest and, if very small drift cells are used, there is not enough space for the sample to fully separate. When using a radioactive source, there are both environmental and safety considerations, which has shifted focus onto using the corona discharge approach. [85, 86] However, corona discharge has its own disadvantages as its process of ionisation can cause variability in the reactant ions produced as well as degradation of those ions. Yet due to its popularity and advantages, there is further research in the area developing systems for portable use. These range from using capillary columns or sorbent polymers for pre-separation to changing or adding electronics and dopants to increase ion production or ion focusing.[87-91]

GC-MS and GC-IMS are also used for field measurements of explosives and CWAs. Normally the size of a mass spectrometer would mean that it is not considered as a field portable piece of equipment but there are some examples where development has been undertaken to vastly reduce the size of the MS. A key example would be the Guardion, which uses a low thermal mass column connected to a miniaturized toroidal ion trap.[81] GC can also be coupled to IMS for detection such as the Themo EGIS defender, which is utilised in military and civilian defence globally. This provides very fast GC separation which acts as a pre-separation step before the IMS[80] cell allowing for ng range detection in a matter of seconds.[92]

Other techniques are also applicable but are not mentioned here for brevity. In detection it is common that a combination of these techniques are used to provide orthogonal detection and give increased certainty to the users. However, there is always a drive to improve detection capability by making systems, smaller, lighter, faster and more sensitive.

3.1.2 Valve Modulated GCxGC

The cryogenically modulated GCxGC is a highly sensitive piece of equipment and the addition of a TOF allows for EIC, which removes most of the peaks that do not contain an ion of interest. The problem with this is that a cryogenic GCxGC-TOFMS cannot be taken out into the field and used in an operational scenario. If samples are being sent back to the lab, the instrument would enhance capability as the compound of interest could be removed from a large matrix with minimal manipulation of the sample.

However, in the field a portable solution is required. The cryogenic modulation in Figure 21 demonstrates that, in most instances, the compounds of interest occupy a different separation space to the matrix.

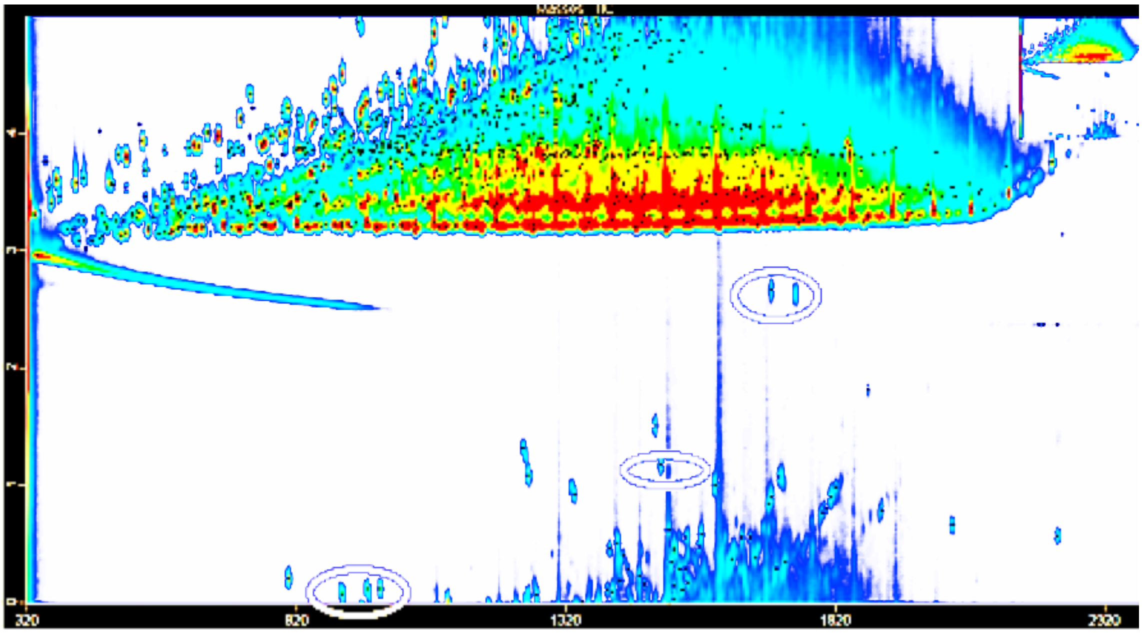


Figure 21 Diesel with 1 ppm of 8330 standard on a cryogenic GCxGC-TOF. The explosives are circled and clearly occupy a different separation space to that of the diesel background.

If this could be translated over to the valve technology (discussed in 1.3.6.3), there is the possibility of undertaking this analysis in the field. This could be integrated into current portable equipment to decrease false alarm rates and allow more challenging areas to be sampled, thus increasing user confidence in the equipment.

A valve modulated GCxGC system from the University of York was established and a FID was used for the detector as it is a more “portable” detector than a mass spectrometer but can also handle the high flow rates required in the second dimension. To the author’s knowledge this is the first time a diaphragm valve has been used to analyse both explosives and chemical warfare agents.

3.2 Experimental

3.2.1 Valve Modulated GCxGC System

An Agilent 6890N GC with a split splitless inlet and a FID was modified to incorporate a 6 port diaphragm valve (AFP valves, Canada) to allow for GCxGC. Actuation of the valve was controlled by an Arduino board programmed to switch a solenoid valve connected to a compressed air line. The valve was actuated at the start of the run and stopped at the end of the run. The valve port setup is shown in Figure 22. Stainless steel tubing (Restek) was used to connect the valve ports to low dead volume unions (Valco Silco Steel), which were then connected to the GC columns.

Visualisation of the 2D data was performed using GC image 2.7^{TM} or R Studio.

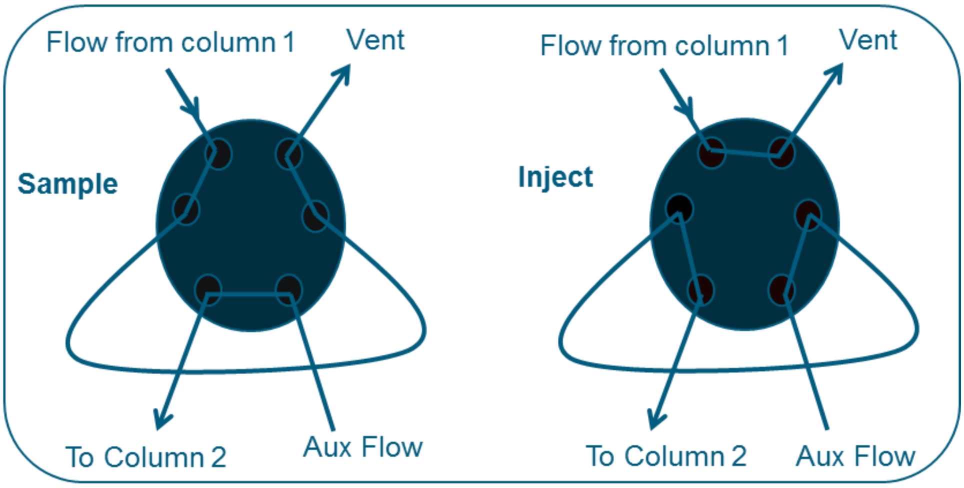


Figure 22 Diaphragm valve schematic to demonstrate port arrangement

3.2.2 Test Mixtures, Calibration Standards and Matrices

3.2.2.1 Test Mixtures

A range of test mixtures were used to develop methods for operational samples.

An alkane and aromatic standard - both standards were purchased from Sigma Aldrich at 1 mg/mL and used to obtain a standard of 50 µg/mL in dichloromethane (DCM).

Diesel – diesel purchased from military stocks (unknown origin).

3.2.2.2 Calibration Standards

All CWA work was carried out at Porton Down, Salisbury Dstl at highly dilute concentrations.

Chemical Warfare Agents (CWA) – CWA standards were made from stocks that had been distilled to a purity of 90+ % (checked by NMR). The agents were then diluted in DCM to 1 mg/mL and further diluted for use in the calibration standards. Standards of 50 µg/mL to 0.2 µg/mL were made up in DCM.

Explosive standards – an 8330 B standard from Restek was purchased and diluted accordingly in acetonitrile to produce standards of 50 µg/mL to 0.5 µg/mL.

Drug standard – a drug standard was purchased from Restek and injected directly into the GC.

3.2.3 Matrices

Matrices of relevance to military operations and matrices of relevance to post-event sampling were replicated. The samples can be split into three sections: dirt, fats and oils, and others.

3.2.3.1 Dirt

1. Swab of a kitchen floor
2. Aerosol sample taken in London from the University of York
3. Aerosol sample from Porton Down range x2
4. Swabs of hotels relevant to search operations x2

5. Sahara Desert aerosol sample

3.2.3.2 Fats and Oils

1. Diesel
2. Light weight weapons oil
3. Gasoline
4. Aviation fuel
5. OX-24
6. OM-33
7. OMD-90
8. Swab of the inside of an oven
9. Vacuum pump oil

3.2.3.3 Others

1. One swab was used to swab a pair of shoes (inside and outside), neck/underarm area of a T-shirt, and the waistband of a pair of jeans

In each case the samples were analysed on their own and then spiked with a known amount of CWA or explosive to determine if detection could be achieved.

3.2.4 Method Development

Several different columns, modulation periods and oven ramps were tested to develop the final method. Each method was assessed based on the separation between two hydrocarbons. Table 7 details methods and columns that were tested.

Table 7 Method development table

Primary	Secondary	Modulation	Pressure	Oven	Method
20 m Rxi-5Sil-MS 0.15 mm 2.0 µm	8 m VF-WAX	2500 ms 400 ms injection	1 = 80 psi 2= 40 psi	50 °C (2 min) 8°/min to 250 °C (8 min)	12
				50 °C (2 min) 10°/min to 250 °C (13 min)	14
				50 °C (2 min) 5°/min to 250 °C (5 min)	21
				50 °C (2 min) 10°/min to 150 °C (5 min) 10°/min to 250 °C (8 min)	20
				50 °C (2 min) 10°/min to 220 °C (10 min) 7°/min to 250 °C (3 min)	22/23
	5 m VF-WAX	3000 ms 300 ms injection	1 = 80 psi 2= 40 psi	50 °C (2 min) 8°/min to 250 °C (8 min)	12
	3 m VF-WAX	3000 ms 300 ms injection	1 = 80 psi 2= 40 psi	50 °C (2 min) 8°/min to 250 °C (8 min)	12
		1800 ms 300 ms injection		50 °C (2 min) 8°/min to 250 °C (8 min)	12
				50 °C (2 min) 5°/min to 250 °C (5 min)	21
	2m Rtx-50	4700 ms 300 ms injection	1 = 75 psi 2= 25 psi	50 °C (2 min) 5°/min to 250 °C (5 min)	21
			1 = 60 psi 2= 25 psi	50 °C (2 min) 6°/min to 280 °C (8 min)	7

3.2.5 GCxGC-FID Method for Analysis of CWA

GCxGC-FID analysis was performed using a 6890 GC from Agilent Technologies fitted with an AFP modulator valve interfaced to a FID. The primary GC column was a RTX-5Sil-MS (20 m x 0.15 mm x 2.0 µm). The secondary GC column was a DB-Wax-MS (8 m x 0.25 mm x 0.25 µm). The primary oven was programmed from 50 °C (2 min) at 8 °C/min to 250 °C (8 min). The modulation period was 1800 ms with a 300 ms injection period. Helium carrier gas was used at a pressure of 80 psi in the primary and 40 psi in the secondary. 1 µL of sample was injected using an Agilent autosampler in splitless mode with the inlet held at 280 °C.

3.2.6 GCxGC-FID Method for Analysis of Explosives

GCxGC-FID analysis was performed using a 6890 GC from Agilent Technologies fitted with an AFP modulator valve interfaced to a FID. The primary GC column was a RTX-5Sil-MS (20 m x 0.15 mm x 2.0 µm). The secondary GC column was a DB-Wax-MS (8 m x 0.25 mm x 0.25 µm). The primary oven was programmed from 50 °C (2 min) at 8 °C/min to 250 °C (8 min). The modulation period was 1800 ms with a 300 ms injection period. Helium carrier gas was used at a pressure of 80 psi in the primary and 40 psi in the secondary. 1 µL of sample was injected using an Agilent autosampler in splitless mode with the inlet held at 175 °C.

3.2.7 GCxGC-FID Method with Secondary Heater

GCxGC-FID analysis was performed using a 6890 GC from Agilent Technologies fitted with an AFP modulator valve interfaced to a FID. The primary GC column was a RTX-5Sil-MS (20 m x 0.15 mm x 2.0 µm). The secondary GC column was a DB-Wax-MS (8 m x 0.25 mm x 0.25 µm). The primary oven was programmed from 50 °C (2 min) at 10 °C/min to 200 °C (3 min). The secondary oven was programmed from 80 °C (2 min) at 10 °C/min to 230 °C (3 min). The modulation period was 1800 ms with a 300 ms injection period. Helium carrier gas was used at a pressure of 80 psi in the primary and 40 psi in the secondary. 1 µL of sample was injected using an Agilent autosampler in splitless mode with the inlet held at 175 °C. If split was required, the split flow was set at 80:1.

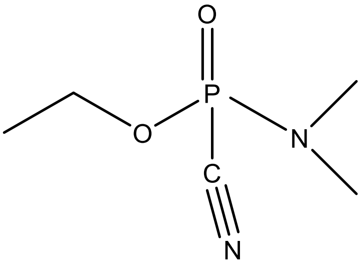
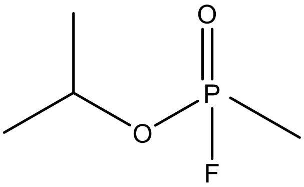
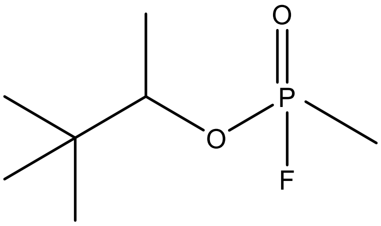
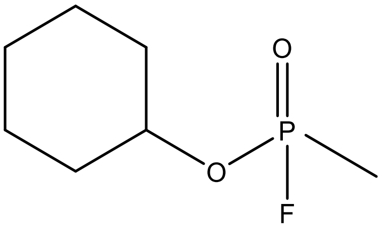
3.3 Results and Discussion

3.3.1 Chemical Warfare Agents

3.3.1.1 CWA Stability

In order to determine the performance and stability of the valve for modulation a study took place using the same samples analysed over the course of a month. Samples of six different CWA at 100 ppm were injected five times and this was repeated every week for a month to establish the stability of the system. The CWAs used can be seen in Table 8.

Table 8 Chemical warfare agents used to determine stability of the valve system

Number	CWA	Structure
1	GA	
2	GB	
3	GD	
4	GF	

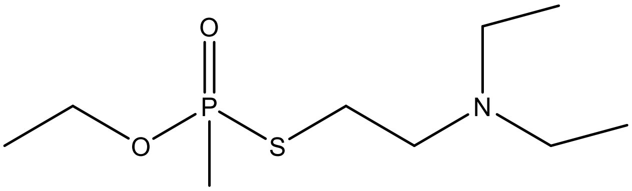
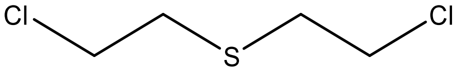
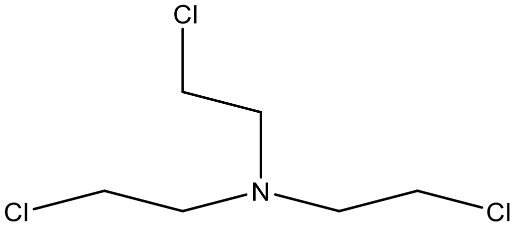
5	VM	
6	HD	
7	HN3	

Table 9 Averages of the volumes detected for each agent and the total average over the 3 weeks.

	1			2			3			Total		
Compound	Average	Stdev	% relative	Average	Stdev	% relative	Average	Stdev	% relative	Average	Stdev	% relative
GA	2502638.62	624192.60	24.94	1051374.45	125002.32	11.89	1127967.28	142080.31	12.60	1560660.12	816675.72	52.33
GB	2103088.86	34845.41	1.66	1964851.45	92026.52	4.68	1912331.33	156220.32	8.17	1993423.88	98536.27	4.94
GD1	1545149.08	51859.82	3.36	1425855.72	45829.56	3.21	1402672.72	52064.07	3.71	1457892.51	76450.28	5.24
GD2	1799865.77	66431.21	3.69	1673074.84	124864.98	7.46	1593478.93	86445.46	5.42	1688806.51	104088.88	6.16
GF	4255768.14	119384.87	2.81	4210542.32	177919.87	4.23	3927949.32	75342.83	1.92	4131419.93	177655.74	4.30
HD	2636017.17	96325.60	3.65	3811411.67	153303.47	4.02	5979828.55	156023.84	2.61	4142419.13	1696302.77	40.95
HN3	2632108.10	59485.12	2.26									
VM	847845.65	120356.88	14.20	1242568.91	161381.40	12.99	986200.15	200014.64	20.28	1025538.24	200280.37	19.53

Table 9, shows that in general the modulation provided good results and only varied marginally over the weeks which is of great benefit as if an unknown sample is being analysed. A library of known retention times could be developed for comparison to the unknown. Where there are large RSDs this may be due to the compound being analysed decreasing in concentration over the time period. This is an important consideration, as often samples must be stored both before and after analysis for traceability. Samples that contain GA, for example may need to be stored at lower temperatures to avoid losses of the sample. (the samples were stored in a fridge maintained at 10 °C)

3.3.1.2 CWA Calibration

The method used for this analysis was detailed in section 3.2.5. Three standards were analysed over a calibration range of 10 to 0.5 μg/mL and three repeats of each standard were conducted:

- First standard contained GB, GD, GA, H, GF, VM, VX, and T

- Second standard contained VX and VM
- Third standard contained H, T and HN3

The calibration for each agent was calculated individually and are in Table 10 to Table 17. The calibration graphs can be found in the Annex 1. It should be noted that, although T could be detected, it struggled to pass through the system and was often at variable levels and wrapping, therefore the results for T are not presented here.

Table 10 Calibration of GB

	Repeats (Intensity)					
Concentration µg/mL	1	2	3	Mean Intensity	Standard Deviation	Relative StandDev %
10.0	27997961	33555947	34252463	31935457	3427709	11
5.0	17338665	21419281	27015435	21924460	4858125	22
2.5	6711907	7527020	9423401	7887443	1391215	18
1.0	2065927	2790680	3535423	2797343	734771	26
0.5	730617	977470	1092954	933680	185095	20

Table 11 GD Calibration first isomer

	Repeats (Intensity)					
Concentration µg/mL	1	2	3	Mean Intensity	Standard Deviation	Relative StandDev %
10.0	17841513	20938874	21706475	20162287	2046167	10
5.0	11318438	13910646	17437131	14222072	3071212	22
2.5	4681879	5191845	6172857	5348860	757789	14
1.0	1560996	1887125	2509870	1985997	482102	24
0.5	634813	705592	866380	735595	118663	16

Table 12 GD Calibration second isomer

	Repeats (Intensity)					
Concentration µg/mL	1	2	3	Mean Intensity	Standard Deviation	Relative StandDev %
10.0	17953256	21193933	21208373	20118521	1875188	9
5.0	12196570	13368475	17079074	14214706	2548881	18

2.5	3789429	4123425	5194603	4369152	734108	17
1.0	1211311	1531241	1948855	1563802	369849	24
0.5	432562	3046405	551320	1343429	1476015	110

Table 11 and

Table 12 demonstrate the two isomers for GD. The isomers are not fully separated in one dimensional GC and methods exist to separate them using super critical fluid chromatography and LC-MS with chiral coloumn.[93, 94] Only two of the isomers have been separated, a chiral column would need to be used for all four isomers to be separated thus allowing for blood studies.

Table 13 GA calibration, 0.5 µg/mL could not be detected

	Repeats (Intensity)					
Concentration µg/mL	1	2	3	Mean Intensity	Standard Deviation	Relative StandDev %
10.0	19158663	23099542	22551262	21603156	2134669	10
5.0	14307871	16091174	20191749	16863598	3017032	18
2.5	7956862	8290743	10053441	8767015	1126516	13
1.0	4259903	5322918	7137850	5573557	1455252	26
0.5						

It should be noted that in Table 13 that only 1 µg/mL can be detected and not 0.5 µg/mL. This is likely due to the chemical properties of GA and it may be lost during the valve modulation process.

Table 14 Mustard calibration, in first repeat of 0.5 µg/mL mustard could not be detected

	Repeats (Intensity)					
Concentration µg/mL	1	2	3	Mean Intensity	Standard Deviation	Relative StandDev %
10.0	26346750	30193085	31637566	29392467	2734762	9
5.0	17331145	21310369	26453164	21698226	4573361	21
2.5	8804990	9995767	11703929	10168229	1457144	14
1.0	4476221	5352495	4332680	4720465	552039	12
0.5		3373036	4057755	3715396	484169	13

Table 15 GF calibration, in the first repeat of 0.5 µg/mL GF could not be detected

	Repeats (Intensity)					
Concentration µg/mL	1	2	3	Mean Intensity	Standard Deviation	Relative StandDev %
10.0	21890931	27721841	27208078	25606950	3228403	13
5.0	13534787	16064454	20380221	16659821	3461335	21
2.5	4943004	5284060	7028268	5751777	1118549	19
1.0	1503135	1759218	638340	1300231	587341	45
0.5		607986	355191	481589	178753	37

Table 16 VM calibration

	Repeats (Intensity)					
Concentration µg/mL	1	2	3	Mean Intensity	Standard Deviation	Relative StandDev %
10.0	23928001	26379243	25328780	25212008	1229786	5
5.0	10712498	11124763	12467896	11435052	917913	8
2.0	3668087	3491859	3740044	3633330	127691	4
1.0	1553160	1485335	1811181	1616559	171926	11
0.8	1023990	1085071	1092069	1067043	37449	4

Table 17 VX calibration

	Repeats (Intensity)					
Concentration µg/mL	1	2	3	Mean Intensity	Standard Deviation	Relative StandDev %
10.0	23042503	25874401	24468595	24461833	1415961	6
5.0	11279512	11607437	13024105	11970351	927190	8
2.0	4669233	4572897	4714308	4652146	72237	2
1.0	2545966	2443078	890434	1959826	927548	47
0.8	1324699	1887439	1243247	1485128	350783	24

3.3.1.3 Limit of Detection and Quantification

The limit of Detection (LoD) was calculated as three times the standard deviation divided by the gradient of the slope, similarly the limit of quantification (LoQ) was calculated as 10 times the standard deviation divided by the gradient of the slope, Table 18.

Table 18 Limit of Detection and Quantification for each agent. RT^1 is the retention time in the first dimension, RT^2 is the retention time in the second dimension. R^2 is the coefficient of determination.

Compounds	$RT_{1\ (min)}$, $RT_{2(sec)}$	LOD (µg/mL)	LOQ (µg/mL)	R^2
GB	7.1, 1.9	0.15	0.46	0.9585
GD	12.5, 0.4	0.01	0.34	0.9525
GD2	12.6, 0.5	0.18	0.46	0.9473
GA	14.4, 1.0	1.06	3.92	0.9520
GF	16.3, 0.8	0.27	0.47	0.9682
H	15.9, 0.8	0.78	1.92	0.9972
HN3	20.3, 1.9	0.83	2.28	0.9997
VX	25.2, 1.7	0.44	1.24	1.000
VM	23.3, 0.4	0.06	0.14	0.9998

3.3.1.4 Development of Software to Alarm on Detection

The idea behind developing a portable GCxGC is to allow it to be used by non-expert users. With this in mind, a piece of software was developed that allows for detection and identification of the system with the simple input of the GCxGC data. Below is an example to demonstrate the use of this system.

A calibration standard of nerve and blister agents at 5 µg/mL was used as a test subject, Figure 23. This was then converted into a greyscale image and then into a black and white image. Bounding boxes around each area of interest were set in the software and this was tested to ensure different concentrations of agent were still present within the bounding box.

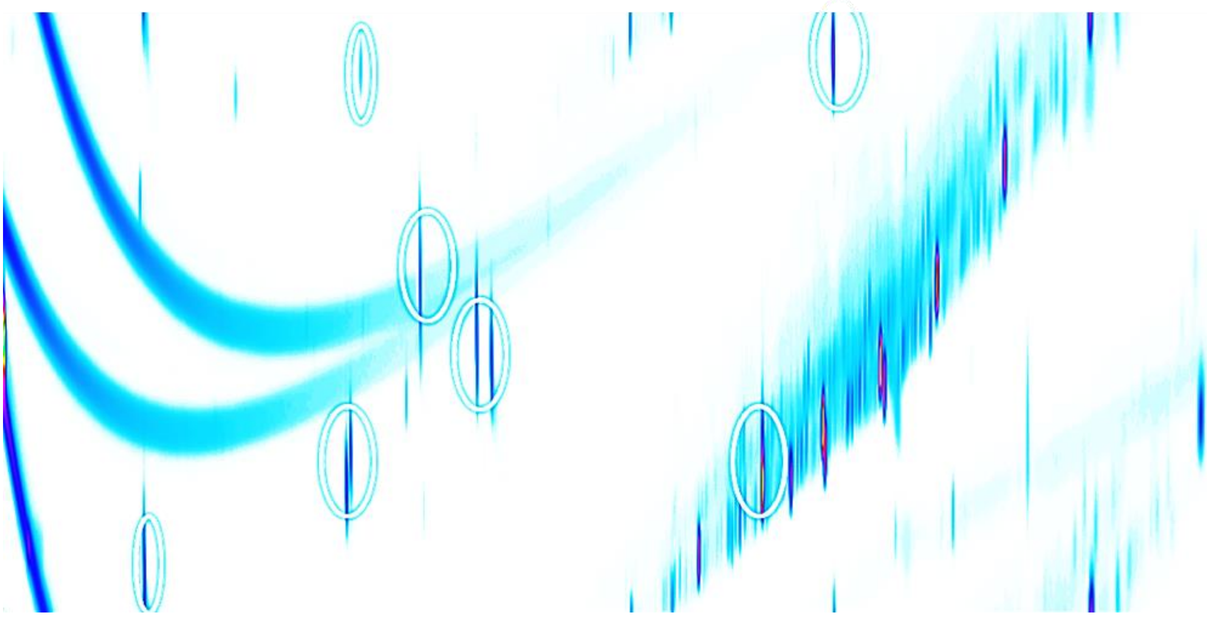


Figure 23 Calibration standard of CWAs at 5 µg/mL – the circled regions relate to different agents. The axis has been removed to allow for input into the software

The software allowed for an image to be loaded and for the user to either detect or identify, Figure 24. “Detect” will tell the user if nerve agent or blister agent is present otherwise it will display not detected, Figure 25. Identify will tell the user what the nerve agent is e.g. GB or GD etc. and will then alarm and the user can undertake appropriate action as per their standard operating procedures, Figure 26. The code for this can be found in the Annex.

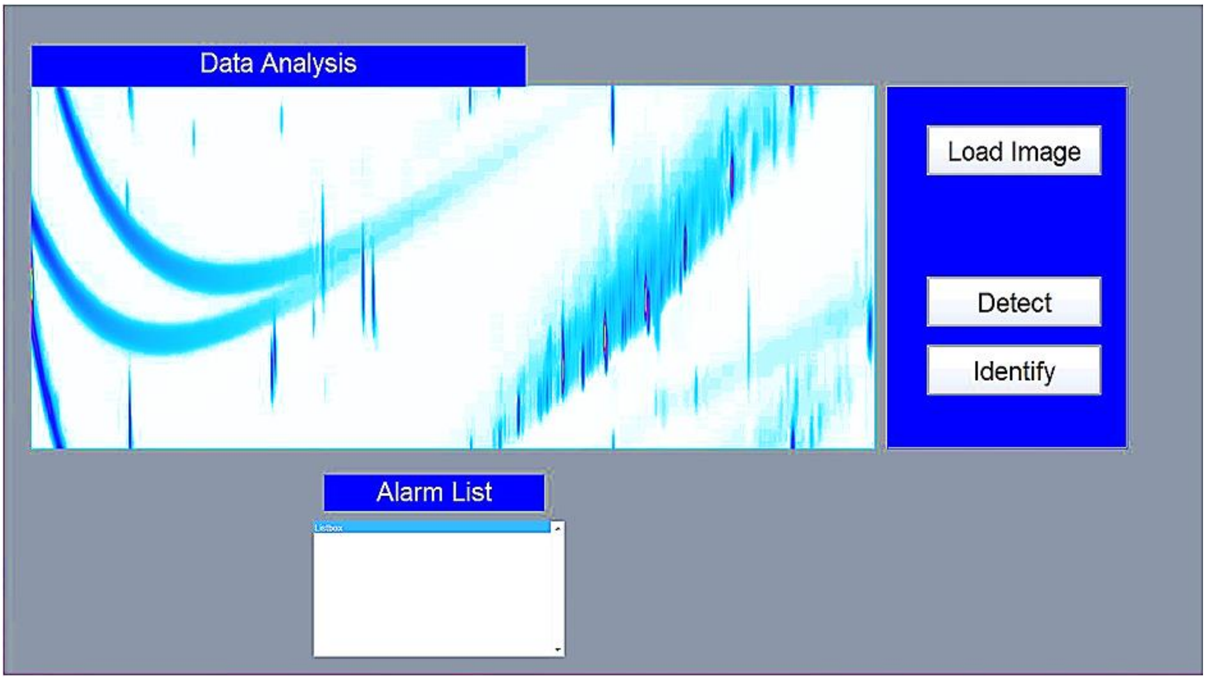
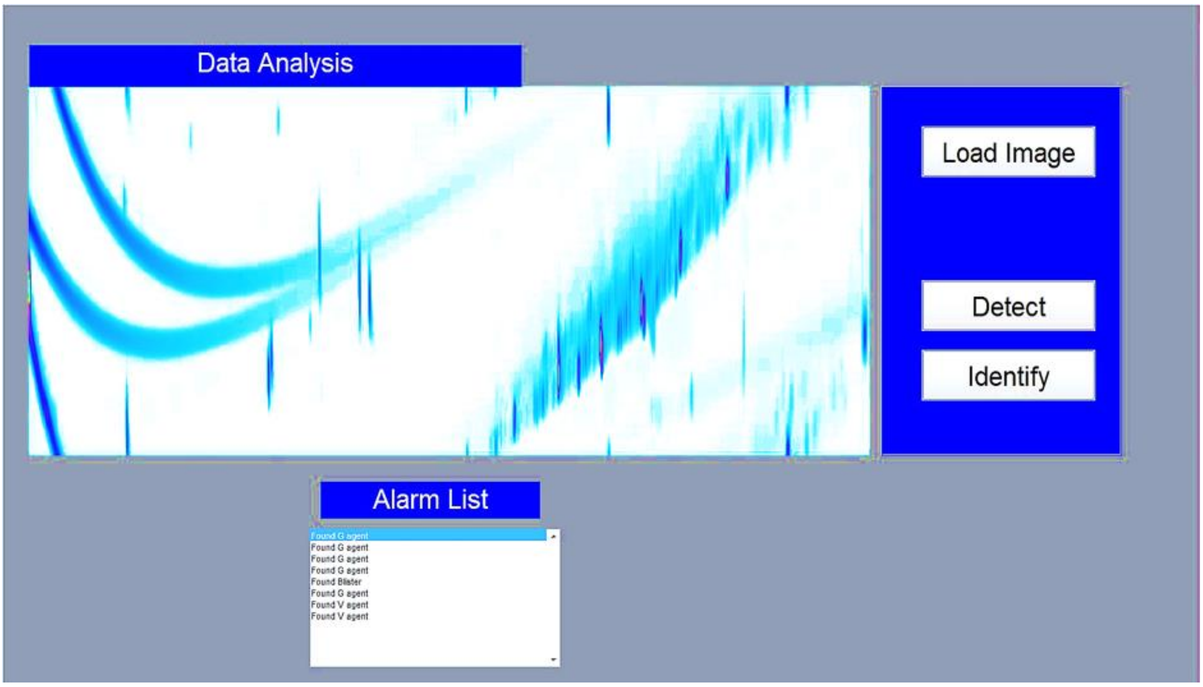
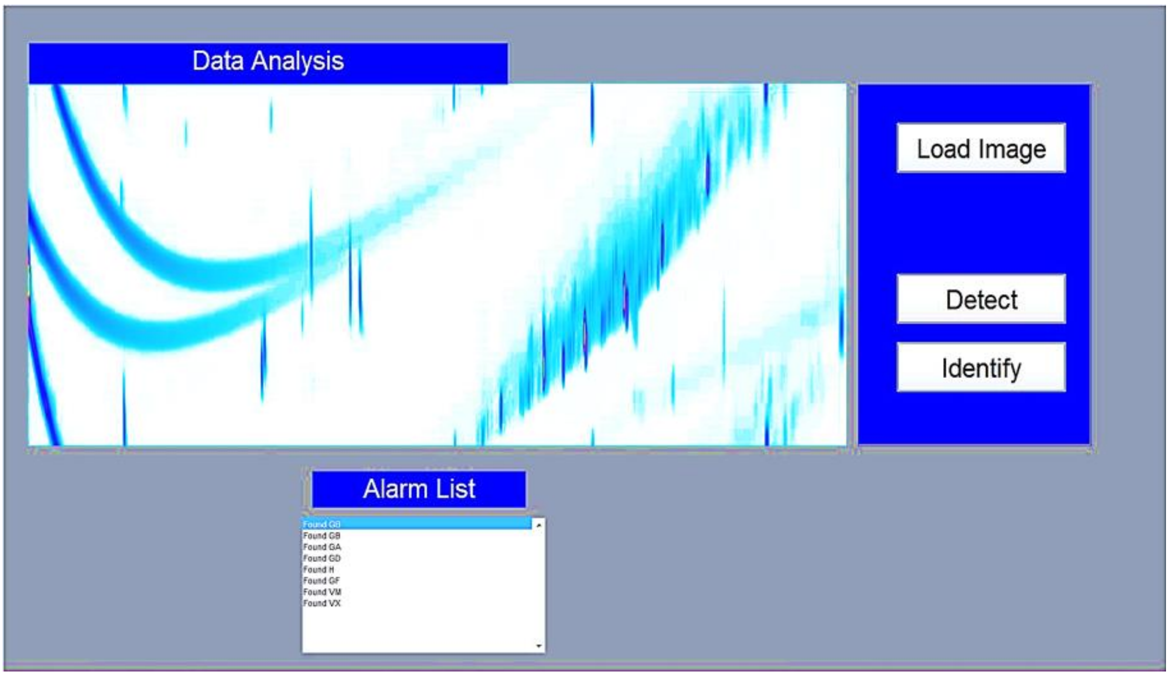


Figure 24 User display of software for data analysis. A chromatogram has been preloaded by using the "Load Image" button



Found G agent
Found G agent
Found G agent
Found G agent
Found Blister
Found G agent
Found V agent
Found V agent

Figure 25 User display once "detect" has been pressed. Under the alarm section it states "found G/V agent" or "not found"



Found GB
Found GB
Found GA
Found GD
Found H
Found GF
Found VM
Found VX

Figure 26 User display showing alarm list after “identify” has been pressed. Each agent is displayed as either “found” or “not found” followed by the agent.

The graphical user interface (GUI) shown in Figure 24 to Figure 26, could be used to allow for quick identification and detection of compounds of interest in complex matrices for a non-expert user. This is important in a military and civilian context, as generally the users are not scientists. Therefore, there is a requirement for the software to be simple and easy to use. This GUI only works for the GCxGC method described in 3.2.5, if the method is changed the positions of the agents must be re-calibrated.

3.3.2 Detection of CWA in Complex Matrices

The matrices listed in section 3.2.3 were spiked with a known concentration of: GB, GD, GA, H, GF, VX and VM. The samples were spiked to produce a final concentration of 6 ng on column to enhance the possibility of detection for this investigation. The results are detailed below in Table 19 and an example chromatogram can be observed in Figure 27. Other chromatograms can be found in Annex 2.

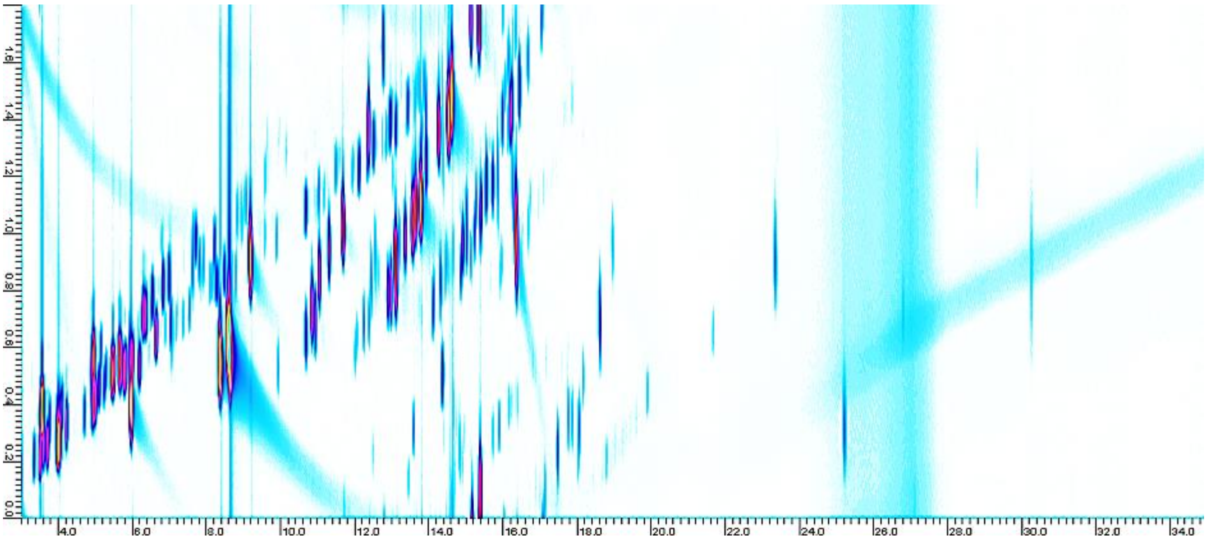


Figure 27 Gasoline spiked with CWA of interest. The x- axis shows the primary retention time in seconds and the y-axis shows the secondary retention time in seconds.

Table 19 Table detailing results of agents detected in complex matrices. Mo3 and Mo4 are aerosol samples from Porton Down range. Green means detected and red means that it was not detected.

	Agent							
Matrix	GB	GD	GD	GA	H	GF	VM	VX
Diesel	✓	✓	✓	✗	✓	✓	✗	✗
Gasoline	✓	✓	✓	✗	✓	✓	✓	✓
Clothing	✓	✓	✓	✗	✓	✓	✓	✓
Oven	✓	✓	✓	✗	✓	✓	✓	✓
Kitchen floor	✓	✓	✓	✗	✓	✓	✓	✓
OMD 90	✓	✓	✓	✗	✓	✓	✓	✓
OM 33	✓	✓	✓	✗	✓	✓	✓	✓
OX 24	✓	✓	✓	✗	✓	✓	✓	✓
AV Gas	✓	✓	✓	✗	✓	✓	✓	✓
Mo3	✓	✓	✓	✗	✓	✓	✓	✓
Mo4	✓	✓	✓	✗	✓	✓	✓	✓
Hotel 1	✓	✓	✓	✗	✓	✓	✓	✓
Hotel 2	✓	✓	✓	✗	✓	✓	✓	✓
LWO	✓	✓	✓	✗	✓	✓	✓	✓
Dust	✓	✓	✓	✗	✓	✓	✓	✓

GA was not detectable in any matrix, in some cases such as aviation fuel, this was due to the overlapping of the peaks, but this could be circumvented with the use of a more selective detector. VX and VM were not easily detectable in diesel due to the high number of heavy hydrocarbons present in diesel making it harder to detect. However, in general GCxGC-FID was able to detect and separate out CWAs of interest from complex matrices that could be found at a scene or in the field. It is becoming increasingly important to be able to reduce the amount of sample manipulation before analysing it both in the field and in the laboratory, thus this technique provides a possible route to analyse the sample without manipulating it.

3.3.3 Explosives

3.3.3.1 Method Development

The system was first tested with three explosives, TNT, EDGN and NG. EDGN and NG could not be observed under these conditions, but TNT could be observed, Figure 28.

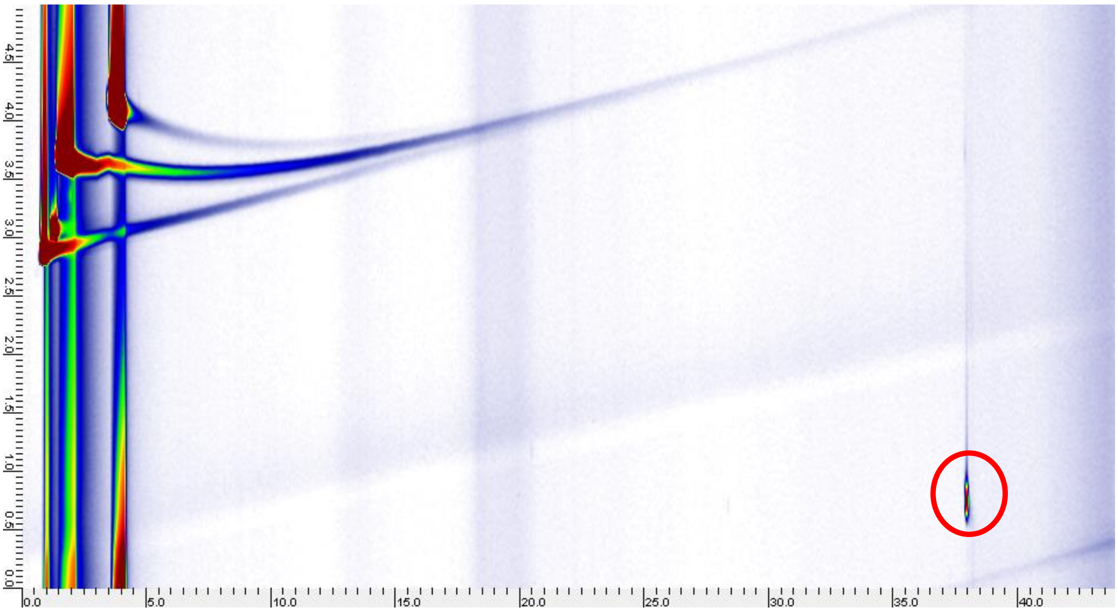


Figure 28 1 ppm of TNT using the valve GCxGC system. The x- axis shows the primary retention time in seconds and the y-axis shows the secondary retention time in seconds. The TNT peak is circled in red.

Although TNT produced an intense peak, it suffered from wrapping in the second dimension and low sensitivity. This necessitated a redesign of the system and silcosteel sample loops were purchased along with silcosteel zero dead volume unions. This was to try to reduce the number of possible active surfaces that could be causing explosives to either not be detected or to wrap in the second dimension.

3.3.3.1.1 Changing Inlet Temperature

The method used for detection of CWAs was tested with explosives to determine if a common method could be used to detect all compounds. Commonly in the field several different detectors are required for detection of both CWAs and explosives, so if one instrument and method could be used this would prove beneficial. When the 8330b standard of explosives was analysed at 280 °C streaks before the peaks were observed, Figure 29.

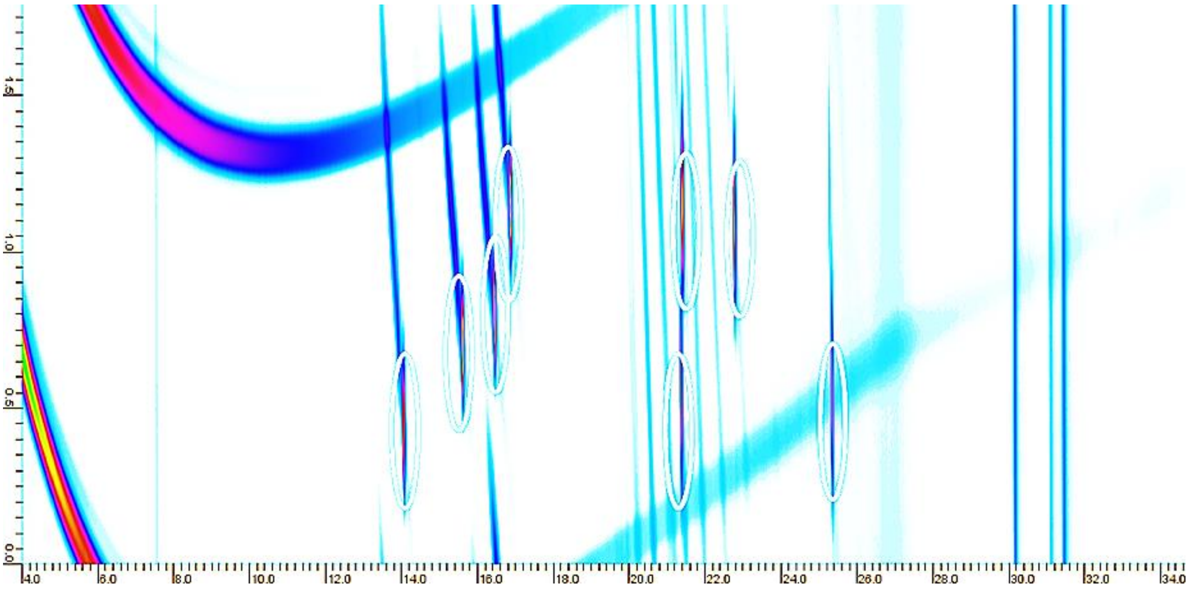


Figure 29 50 ppm of an 8330 b standard, the explosives of interest are circled in white but it can be seen that there are dark blue streaks before each eluted compound. The dark blue streaks are speculated to be thermal breakdown products of the explosives due to the high inlet temperature. The x- axis shows the primary retention time in seconds and the y-axis shows the secondary retention time in seconds.

In Figure 29 it is suspected that the dark blue streaks before each compound were break down product of the explosive due to the higher temperature as this effect is greatly decreased with a drop in the inlet temperature. This theory needs to be tested with a detector and a library that would be able to identify the compounds not just detect. Due to the degradation effect, the method for analysing explosives was changed to run the inlet at 175 °C, Figure 30, this provided a decrease in the effect and a better signal to noise ratio (SNR).

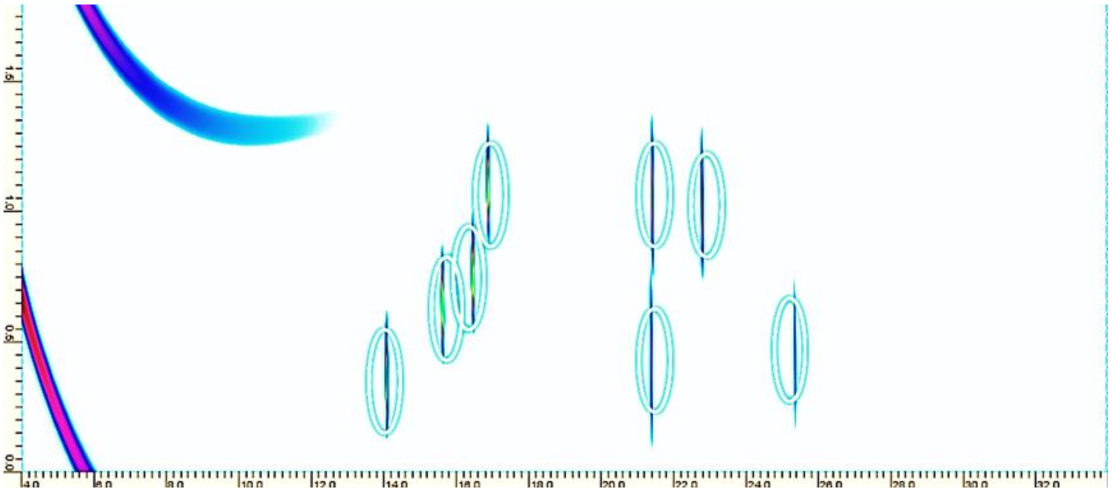


Figure 30 50 ppm of 8330b explosive standard analysed at 175 °C. Explosives are detected (circled in white) but there is no degradation product present before the compounds. The x- axis shows the primary retention time in seconds and the y-axis shows the secondary retention time in seconds.

Interestingly, comparing Figure 29 and Figure 30, towards the end of the chromatogram in Figure 29, there are three bands between 30 – 32 minutes that are not present in Figure 30. These peaks relate to 2-amino-4,6-dinitrotoluene, 3,5-dinitroaniline and 2-amino-4,6-dinitrotoluene respectively. They are not modulated as they probably interact with the valve or with one of the phases too strongly so appear to bleed through in the chromatography. At the lower temperature they are not observed, which implies that to detect these compounds a re-design of the system would need to take place to elute these compounds. However, they are not currently of military interest so no further work to elute them took place.

3.3.3.2 Explosive Calibration

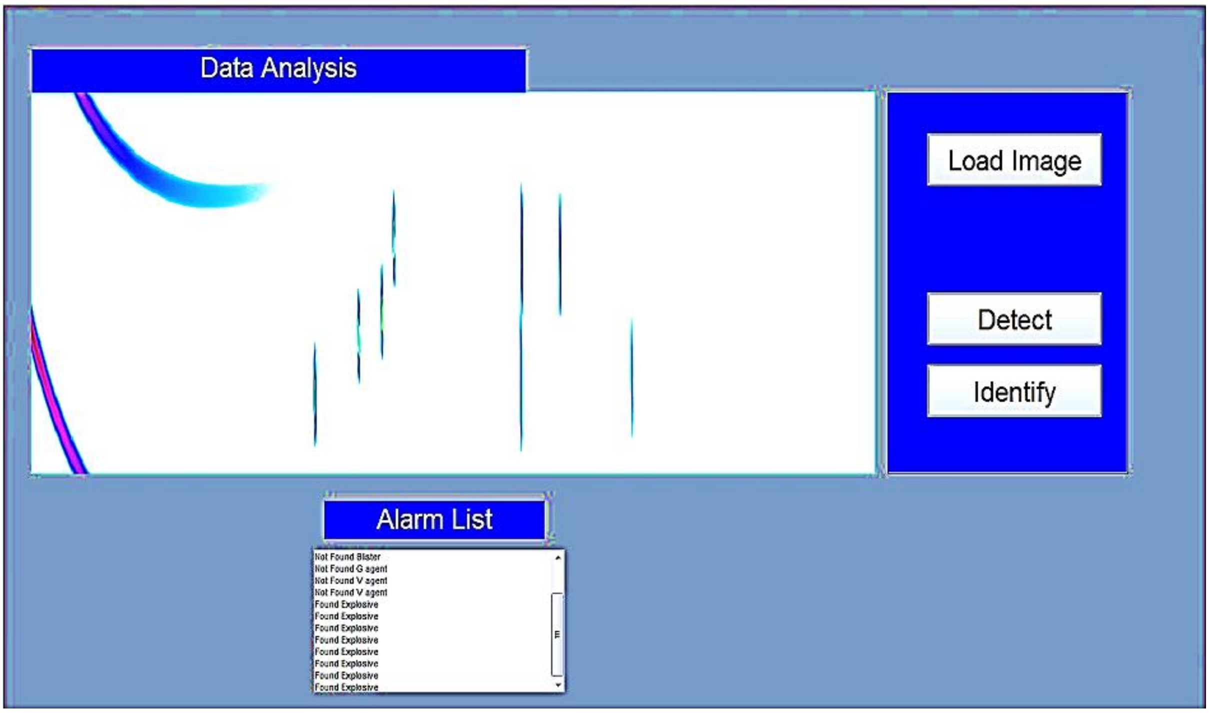
The final method used to determine the limits of detection is detailed in section 3.2.6. Standards of explosives were made in acetonitrile between 50 – 0.5 µg/mL and repeated three times. The limit of detection has been calculated using a 97.5 % confidence interval using the t-test value multiplied by the standard deviation for the lowest recorded concentration. The limit of detection is ten times the standard deviation of the lowest recorded concentration. The results are shown in Table 20. It should be noted that RDX, HMX, NG and EDGN could not be detected in the system. This could be due to their high activity with active surfaces or sensitivity to heat i.e. they may be breaking down in the inlet before detection can take place. It is also notable for the higher boiling point explosives; that it became difficult to detect them by eye in the chromatogram as the concentration decreased. This is likely to be due to their higher boiling point, so they may be sticking to an active surface in the valve thus not reaching the detector.

Table 20 – Calibration of explosives detailing the limit of detection (LoD) and quantification (LoQ) based on n=3

Compound	$RT_{1\ (min)}$, $RT_{2(sec)}$	LOD (µg/mL)	LOQ (µg/mL)	R^2
2-NT	14.04, 0.38	0.28	1.10	0.9954
3-NT	15.60, 0.61	0.41	1.32	0.9956
4-NT	16.40, 0.72	0.29	0.99	0.9962
1,3-DNB	16.86, 1.10	0.26	0.84	0.997
2,6-DNT	21.33, 2.19	0.05	2.71	0.9993
2,4-DNT	21.39, 1.02	0.78	1.91	0.9986
1,3,5-TNB	22.77, 0.97	1.25	2.97	0.9997
2,4,6-TNT	25.32, 0.45	0.59	5.01	0.999

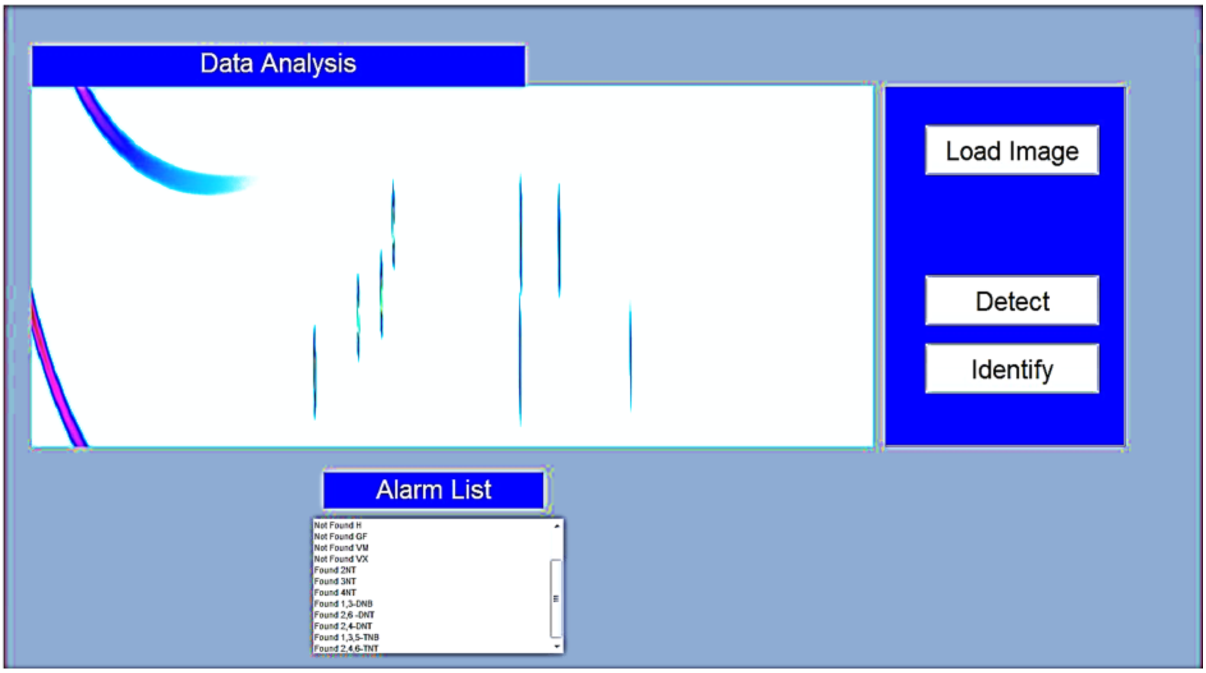
3.3.3.3 Detection of Explosives using Software

The GUI written above for the chemical warfare agents was modified to include explosives. The GUI was tested with five samples of varying concentration. For each concentration the GUI could detect the explosive and, more importantly, it could not find any CWAs meanings the explosives and CWAs occupy different separation space, Figure 31 and Figure 32.



```
Not Found Blister
Not Found G agent
Not Found V agent
Not Found V agent
Found Explosive
Found Explosive
Found Explosive
Found Explosive
Found Explosive
Found Explosive
Found Explosive
Found Explosive
```

Figure 31 Detection function in GUI on a 50 ppm standard of the 8330 explosive mix



```
Not Found H
Not Found GF
Not Found VM
Not Found VX
Found 2NT
Found 3NT
Found 4NT
Found 1,3-DNB
Found 2,6 -DNT
Found 2,4-DNT
Found 1,3,5-TNB
Found 2,4,6-TNT
```

Figure 32 Identify function on a 50 ppm standard of explosives. It displays "not found" for each agent and "found" followed by each explosive.

3.3.3.4 Detection of Explosives in Complex Matrices

Matrices of relevance to explosive detection were used to simulate environments that could cause false alarms. In this instance as well as some of the matrices from previous sections, perfume was also used as explosive detection often requires screening of a person. The hotel sample was also collected during a military exercise at a hotel and contained interferents that could not be resolved using one piece of equipment during the exercise. Explosives were spiked into each matrix and injected using a 100:1 split producing a final concentration of 1 µg/mL, therefore 1 ng on column. Table 21 details which explosives were detected in each matrix. An example chromatogram is shown in Figure 33. All other chromatograms can be seen in Annex 3.

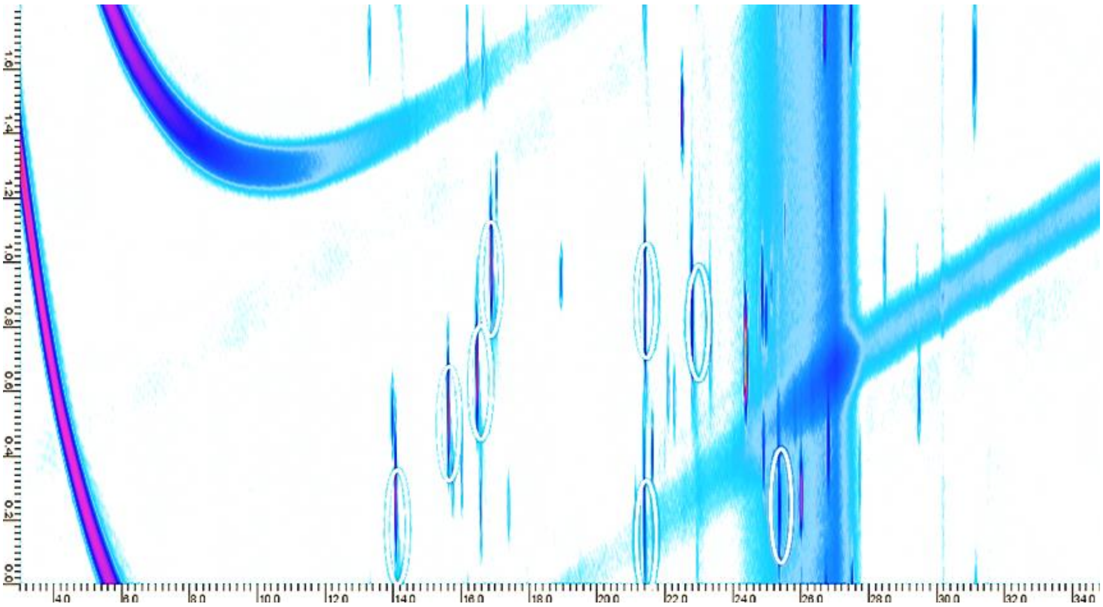


Figure 33 Perfume spiked with 1 ppm of 8330b explosive standard. The explosives detected are circled in white. The x- axis shows the primary retention time in seconds and the y-axis shows the secondary retention time in seconds.

Perfumes are a typical example of a possible interferant that could be found in aviation detection. Most perfumes contain musks that can cause some difficulty for explosive detection. However, this is compensated for in the chromatogram above and the explosives are identifiable in the complex mixture.

Table 21 Table of matrices detailing which explosives could be detected using the method detailed in section 3.2. Green means detected and red means not detected.

		Compound							
		2-NT	3-NT	4-NT	1,3-DNB	2,6-DNT	2,4-DNT	1,3,5-TNB	2,4,6-TNT
Matrix	Perfume	✓	✓	✓	✓	✓	✓	✓	✓
	PTN aerosol	✓	✓	✓	✓	✓	✓	✓	✓
	Hotel 2	✓	✓	✓	✓	✓	✓	✓	✓
	Sahara	✓	✓	✓	✓	✓	✓	✓	✓
	Aviation fuel	×	×	×	×	×	×	×	×
	Diesel	×	✓	×	×	×	×	×	×
	Gasoline	×	×	×	×	✓	✓	✓	✓
	Light Weight Oil	✓	✓	✓	✓	×	×	×	✓
	OX-24	✓	✓	✓	✓	×	×	×	×

As observed in Table 21, explosives are difficult to detect in these complex matrices thus false alarms could be caused using this method. In the aim to produce a portable system that would not produce false alarms, modifications to this system took place.

3.3.4 Secondary Oven

In the more commonly used, cryogenic two-dimensional gas chromatography; the secondary column sits within a separate oven. This allows the compounds to be separated by polarity as the oven emulates isothermal conditions thus helping to improve the chromatography in many cases. The issues with a secondary column oven include; higher power requirements, slower run times and slower turn around due to the second oven taking longer to cool. However, despite this a secondary oven was built to test if it could improve separation and if it could then, whether it could decrease false alarm rates and have fast methods.

3.3.4.1 Secondary Oven Design

The secondary oven design was based upon that used in the cryogenic work with a few exceptions. High temperature insulation was fitted in to a rectangle of aluminium and pieces were drilled out to help improve airflow and the column was coiled tightly and placed inside the oven, Figure 34. An engineer at Porton Down, Jack Vincent, helped to build the system due to electrical hazard and health and safety considerations.

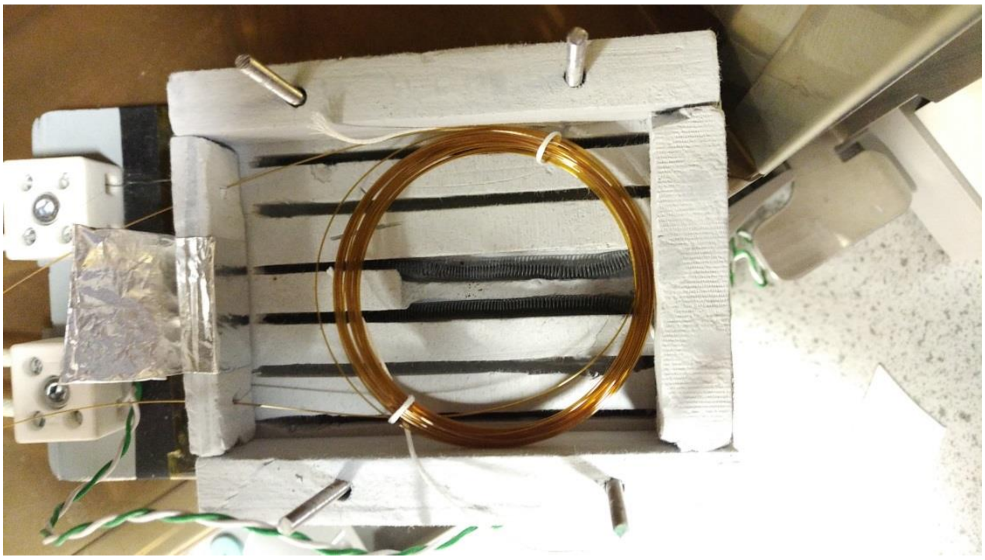


Figure 34 Inside of secondary oven, the insulation has had pieces removed to help with airflow and 8 meters of column has been tightly coiled and placed inside.

This system had a lid, again covered in insulation, that had similar pieces removed. Heater wire was coiled and placed inside the holes where the insulation previously was. A thermocouple was placed in the middle of the oven to ensure even heating and finally this was placed inside the GC, Figure 35. The oven was connected to the GC using AUX 2 and controlled using the Chemstation software.

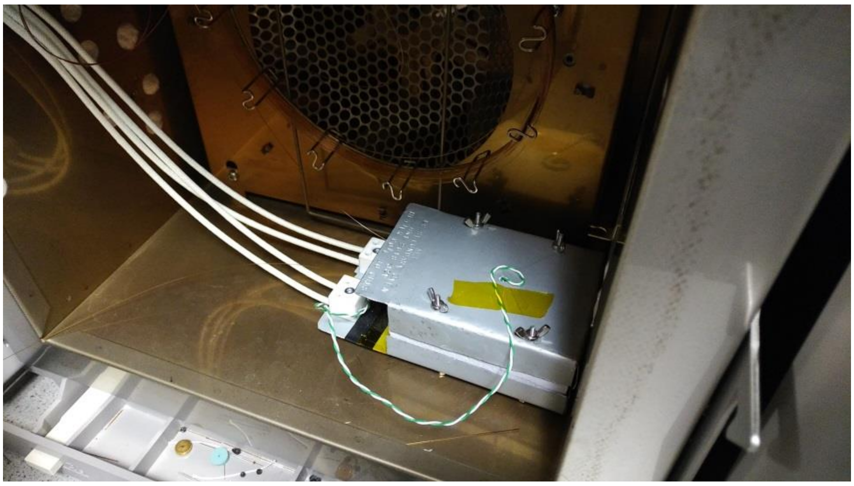
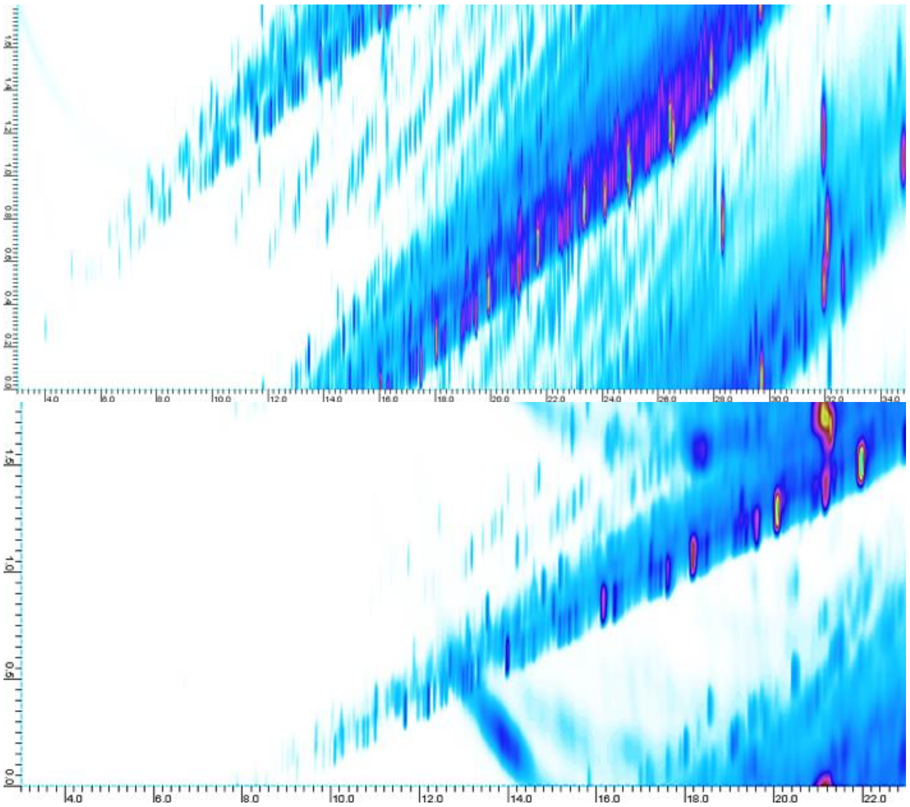


Figure 35 Secondary oven placed inside the oven. The green and white wire is the thermocouple placed in the centre of the oven. The large white wires continue to outside the GC and connect to the AUX 2 port.

Diesel was analysed using the system using the same method but with the secondary oven placed at a 40 °C offset and a difference in the chromatography can be seen quite clearly, Figure 36.



Diesel without secondary oven

Diesel with secondary oven

Figure 36 Diesel run using the same method, on the top without a secondary oven and on the bottom with a secondary oven and a 40 °C offset. The x- axis shows the primary retention time in seconds and the y-axis shows the secondary retention time in seconds.

In Figure 36 it can be observed that the secondary oven allows the alkane series to not wrap around, this can be observed as they are almost in a vertical line across the chromatogram instead of three diagonal lines, thus providing the possibility for separation of this type of matrix from explosives. Further method development took place to improve the separation and speed of separation for analysis of complex matrices.

3.3.4.2 CWA Analysis in Complex Matrices

The matrices listed in section 3.2.3 were spiked with a known concentration of: GB, GD, GA, H, GF, VX and VM. The samples were spiked to produce a final concentration of 6 ng on column to enhance the possibility of detection for this investigation. The results are detailed below in Table 22 and an example chromatogram is shown in Figure 37. Other chromatograms can be found in Annex 2.

Table 22 Matrices Spiked with CWA. A tick means that the agent could be detected and a cross means that it could not be detected

		Agent							
		GB	GD1	GD2	GA	H	GF	VM	VX
Matrix	Aerosol sample 3	✓	✓	✓	✓	✓	✓	✓	✓
	Clothing	✓	✓	✓	✓	✓	✓	✓	✓
	Inside Oven	✓	✓	✓	✓	✓	✓	✓	✓
	Hotel	✓	✓	✓	✓	✓	✓	✓	✓
	Gasoline	✓	✗	✗	✗	✗	✗	✓	✓
	LWO	✓	✓	✓	✓	✓	✓	✗	✗
	AV Fuel	✓	✓	✓	✗	✗	✓	✓	✓
	Diesel	✓	✓	✓	✗	✗	✓	✗	✗
	Sahara Dust	✓	✓	✓	✓	✓	✓	✓	✓

In this instance the detection using the secondary oven is worse for the detection of CWAs in complex matrices (Table 22 cf. Table 19). The GC method was modified to create a fast separation method, in an attempt to counteract the increased total run time caused by the secondary oven having to cool. This had a secondary and expected effect of creating less separation space in the chromatogram and thus making it harder to detect some of the agents of interest. Again, a non-universal detector or another separation technique followed by a detector would help to deconvolute the signal. An example of this would be ion mobility (IMS) as it contains a drift region which would allow for further separation followed by a universal detector plate.

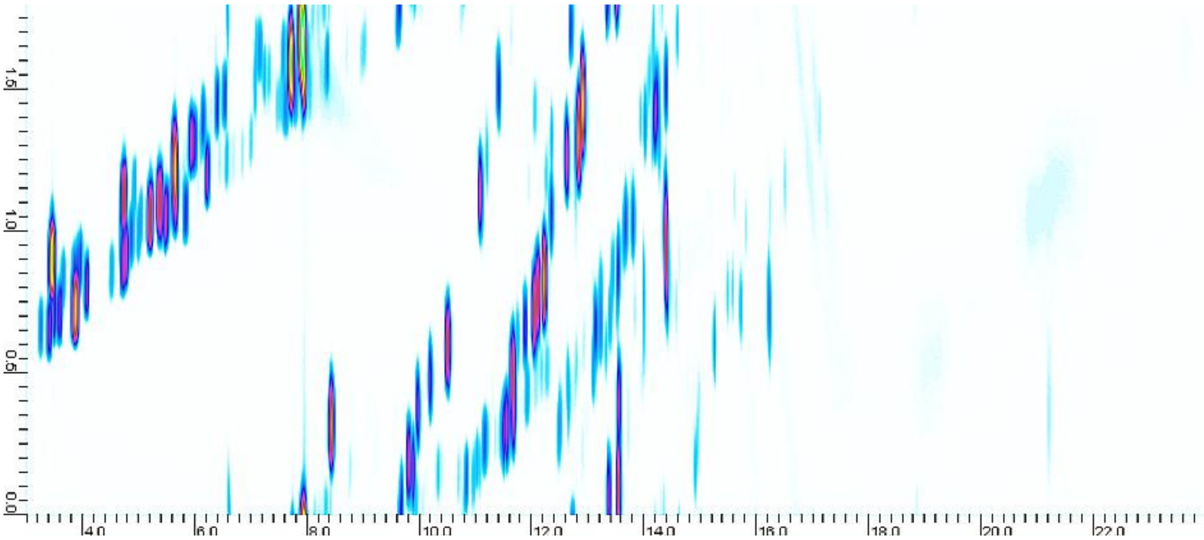


Figure 37 Chromatogram of Gasoline spiked with CWA nerve and blister standard. The x- axis shows the primary retention time in seconds and the y-axis shows the secondary retention time in seconds.

Figure 37 shows a chromatogram of gasoline spiked with a CWA standard containing both nerve and blister agents. Gasoline contains a much lighter fragment of hydrocarbon species in comparison to a matrix such as diesel, which makes it easier to detect the higher boiling point agents but more difficult to detect the species that sit in the middle of the chromatogram as they have similar retention times to the hydrocarbon species.

3.3.4.3 Secondary Oven for Explosives Detection

The matrices listed in section 3.2.3 were spiked with a known concentration of 8330b explosive standard. This was analysed using GCxGC-FID and the secondary oven to try to improve detection in complex matrices. The results are detailed below in Table 23 and the chromatograms can be found in Annex 2.

Table 23 List of Matrices with Explosives that could be detected in the matrix. A tick indicates that it was detected, and a cross indicates that it was not detected.

		Compound					
		4-NT	1,3-DNB	2,6-DNT	2,4-DNT	1,3,5-TNB	2,4,6-TNT
Matrix	Perfume	✓	✓	✓	✓	✓	✓
	PTN aerosol	✓	✓	✓	✓	✓	✓
	Hotel 2	✓	✓	✓	✓	✓	✓
	Sahara	✓	✓	✓	✓	✓	✓
	Aviation fuel	✗	✗	✗	✗	✗	✗
	Diesel	✗	✗	✗	✗	✗	✗
	Gasoline	✓	✓	✓	✓	✓	✓
	Light Weight Oil	✓	✓	✓	✓	✓	✓

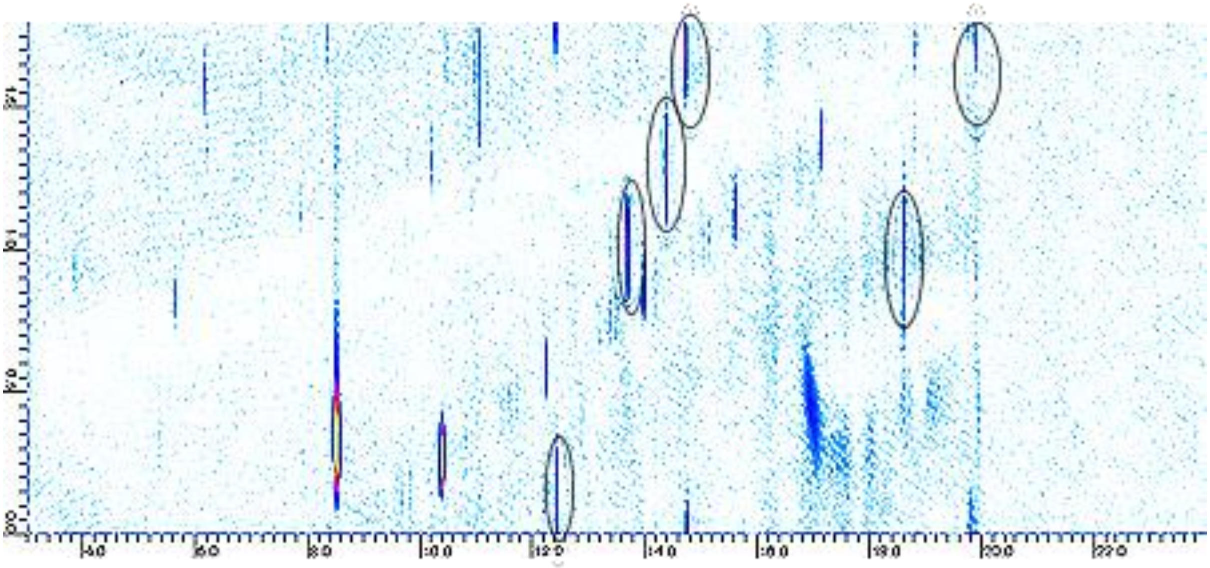


Figure 38 Light weight oil (LWO) spiked with 8330b explosives standard. The explosives detected are highlighted in white circles. The x- axis shows the primary retention time in seconds and the y-axis shows the secondary retention time in seconds.

As can be observed in Figure 38 the explosives are in a different chromatographic region and are easily detected in light weight oil. However, none of the explosives could be detected in diesel or aviation fuel, Table 23. In this instance, the loss of chromatographic separation in the first dimension, due to the decrease in run time caused by using a fast oven ramp, caused significant overlap between the explosives and hydrocarbon species. Having a secondary oven, which should have in theory improved the secondary separation, did not compensate this for this effect. However, in stating that it was not improved for those two matrices, the separation was complete for all other matrices in a much shorter time frame than without the secondary oven.

Chapter 4

Portable Valve Modulation Comprehensive Gas Chromatography Flame Ionisation Detection for the Analysis of Threat Materials

4 Portable Valve Modulation Comprehensive Gas Chromatography

4.1 Introduction

Portable detection equipment is used by military and civilians to detect the presence of explosives or CWAs.[40, 49, 78-80] This has been explored in Chapter 3, however we shall highlight the main points of interest again for this chapter.

The threat of CWAs and explosives is increasing rapidly. Some of the raw materials are easily accessible, and the internet allows a platform to not only share notes but detailed videos on how to make materials that are life threatening. Therefore, as highlighted before, there is a need to be able to detect and identify these materials in the field. The current in-service detection equipment must meet regulations set up by users and subject matter experts in the area and is generally broken down into a few techniques; IMS/DMS, Raman, IR, GC and MS. These were explored in Chapter 3.

4.1.1 Portable Gas Chromatography

GC-MS and GC-IMS are also used for field measurements of explosives and CWAs. Normally the size of a mass spectrometer would mean that it is not considered as a field portable piece of equipment but there are some examples where research has been undertaken to vastly reduce the size of the MS. A key example in detection would be the Guardion, this uses a low thermal mass column connected to a miniaturized toroidal ion trap.[81]

4.1.2 Portable Two Dimensional Gas Chromatography (GCxGC)

Comprehensive two dimensional gas chromatography (GCxGC) has already demonstrated that it has high value in the analytical laboratory for forensic sample analysis, as it has highly increased peak capacity in comparison to GC and the ability to resolve exceedingly complex matrices with minimal sample preparation.

However, conventional GCxGC require cryogenics, typically liquid nitrogen, which makes then non-portable. They also require long analysis times to provide the best results, typically 45 – 60 minutes. This is an issue if a tight turnaround is required. There is a need to be able to analyse complex samples in the field with minimal false alarm rates. If GCxGC is made portable and more accessible, then there is the possibility of using it in a field environment. An example of this in the literature is research undertaken by W.R. Collin et al, J. Lee et al. and S.Edwards et al. [95-98]

In all of the literature it is clear that it is very difficult to achieve even heating, no cold spots, and the correct balance of flows. Each group took a different engineering approach to the problem, but all had very similar issues with heating and cold spots. A few papers demonstrated that the VOCs measured could be separated despite the engineering problems. However, none of them offer a full solution.

This chapter explores the possibility of producing a portable GCxGC for military operation and threat materials. This research uses a 6-port afp valve similar to that of S.Edwards and uses very similar pressures to try to achieve the correct flows required for using a valve i.e. higher pressure in the primary and lower pressure in the secondary.

4.2 Experimental

4.2.1 Fast GCxGC

4.2.1.1 Sample Preparation

A 1000 ppm 8330 explosive standard (Restek) was diluted in ethyl acetate (Sigma Aldrich) to achieve a 10 ppm standard and a 1 ppm standard. Explosives present in 8330 standard can be seen in Table 24.

Table 24 8330 standard components

Explosive
4-amino-2,6-dinitrotoluene
2-amino-4,6-dinitrotoluene
1,3,5-trinitrobenzene
TNT
Tetryl
Nitrobenzene
RDX
HMX
2-nitrotoluene
3-nitrotoluene
4-nitrotoluene
1,3-dinitrobenze
2,6-dinitrotoluene
2,4-dinitrotoluene

4.2.1.2 GCxGC-TOF Parameters

4.2.1.2.1 Starting Conditions

GCxGC-TOF analysis was performed using a 6890B GC from Agilent technologies fitted with a Leco thermal modulator system interfaced to an Agilent TOF-MS. The primary GC column was a BX5 (30 m x 0.25 mm x 0.25 µm). The secondary GC column was a BX50 (4 m x 0.2 mm x 0.2 µm). The primary oven was programmed from 40 °C (2 min) at 7 °C/min to 270 °C (5 min). The secondary oven was programmed from 70 °C (2 min) at 7 °C/min to 300 °C (5 min). A dual jet liquid nitrogen modulation system was used with a 5 second modulation period. Helium carrier gas was used at a flow rate of 1 mL/min. 1 µL of sample was injected using a Gerstal auto-sampler in splitless mode.

4.2.2 Portable GC

A portable GC was designed using a block of aluminium hollowed out and surrounded by a heater band, Figure 39. The heater band was controlled using an OMEGA PID which provides its own software to allow for calibration of the heating.

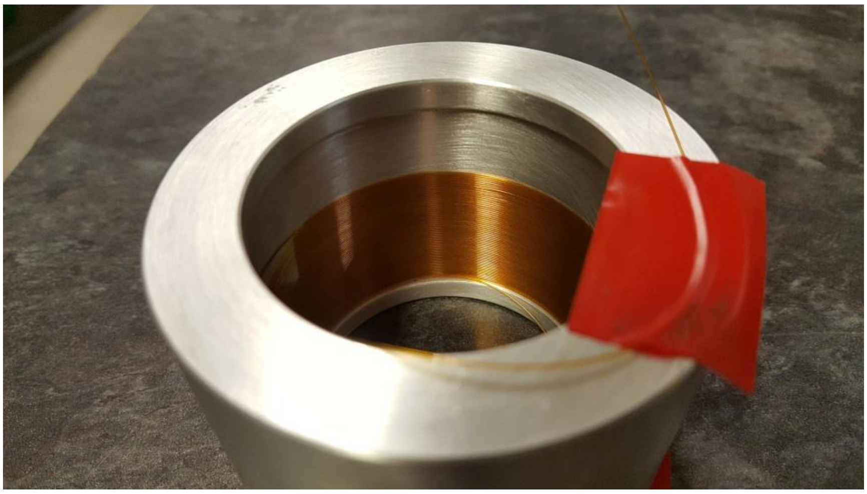


Figure 39 Picture of the aluminium holder with the column wound inside.

4.2.2.1 Portable GC Method

GC analysis was performed using a 6890B GC from Agilent technologies fitted with a portable GC as described above fitted to a FID. The GC column was a BX5 (15 m x 0.25 mm x 0.25 µm). The oven was programmed to hold under isothermal conditions at 140 °C. The oven the system was placed in was held at 100 °C. Helium carrier gas was used at a flow rate of 1.5 mL/min. 1 µL of sample was injected using an Agilent auto-sampler in splitless mode.

4.2.3 Portable GCxGC

A portable GCxGC was designed based on the portable GC and is described in detail below. The system was capable of controlling all heating zones but required an external injector and detector.

A series of experiments were designed to test the code and the chromatography. Note that the code was written by Timothy Ayers and modified by the author as and when bugs in the software were detected.

4.2.3.1 GCxGC Method Development

Method 1:

The primary GC column was a BX5 (17 m x 0.18 mm x 0.36 µm). The secondary GC column was a BX50 (6 m x 0.18 mm x 0.2 µm). The primary oven was programmed from 40 °C at 10 °C/min to 250 °C (5 min). The secondary oven was programmed from 60 °C at 10 °C/min to 270 °C (5 min). A diaphragm valve was used with a 5 second modulation period. Helium carrier gas was used at a flow rate of 1.5 mL/min. 1 µL of sample was injected using an Agilent autosampler in splitless mode.

Method 2:

The primary GC column was a BX5 (17 m x 0.18 mm x 0.36 µm). The secondary GC column was a BX50 (6 m x 0.18 mm x 0.2 µm). The primary oven was programmed from 55 °C at 14 °C/min to 240 °C (5 min). The secondary oven was programmed from 75 °C at 14 °C/min to 260 °C (5 min). A diaphragm valve was used with a 5 second modulation period, 4700 ms modulation with 300 ms injection. Helium carrier gas was used at a flow rate of 1.5 mL/min. 1 µL of sample was injected using an Agilent autosampler in splitless mode.

Method 3:

Same as above but the front inlet was changed to 50 psi

Method 4:

Same as above but with a ramp rate of 6 °C/min.

Method 5:

Same as above but the primary flow was 60 psi and the secondary was 30 psi.

4.3 Results and Discussion

4.3.1 Fast GCxGC

GCxGC is excellent for an analytical laboratory environment where time constraints are not necessarily a problem. Therefore, a fast GCxGC method with limited loss of resolution is desired. As described above a series of methods were tested to increase the speed of analysis to determine if it is possible with limited loss of information. The “gold” standard of GCxGC was used as a starting point for development of the fast method.

Figure 40 shows the first attempt at trying to decrease the analysis time. The separation was completed in 15 minutes for the 8330b standard and was carried out by increasing the oven ramp. It was clear that the separation efficiency has decreased and that there was less clear definition between each explosive especially the isomers, Figure 40.

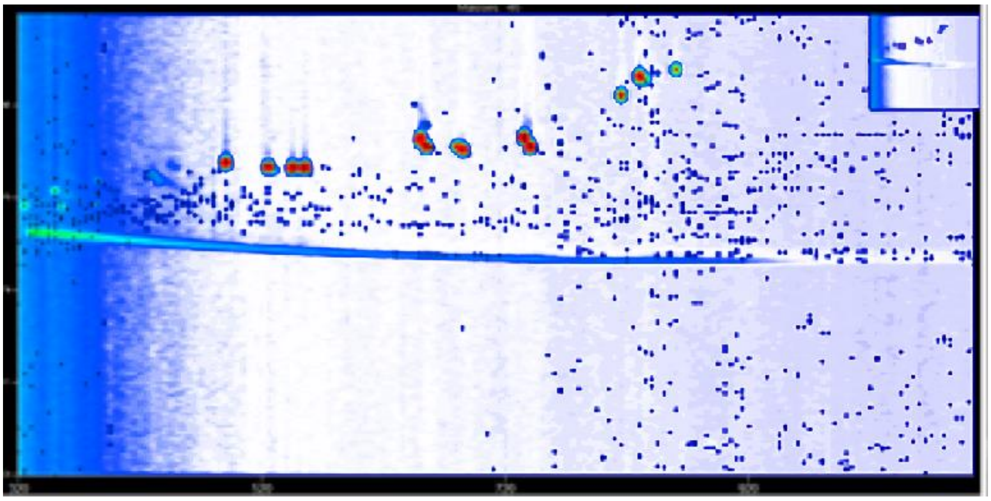


Figure 40 “fast GCxGC” a fast increase in the oven temperature by setting a ramp of 20 °C/min. The separation between these compounds has been reduced both in the 1st and 2nd dimension. The x- axis shows the primary retention time in seconds and the y-axis shows the secondary retention time in seconds.

In Figure 41 the separation takes place in less than 6 minutes and it is an example of where separation efficiency has decreased such that explosives are co-eluting with one another.

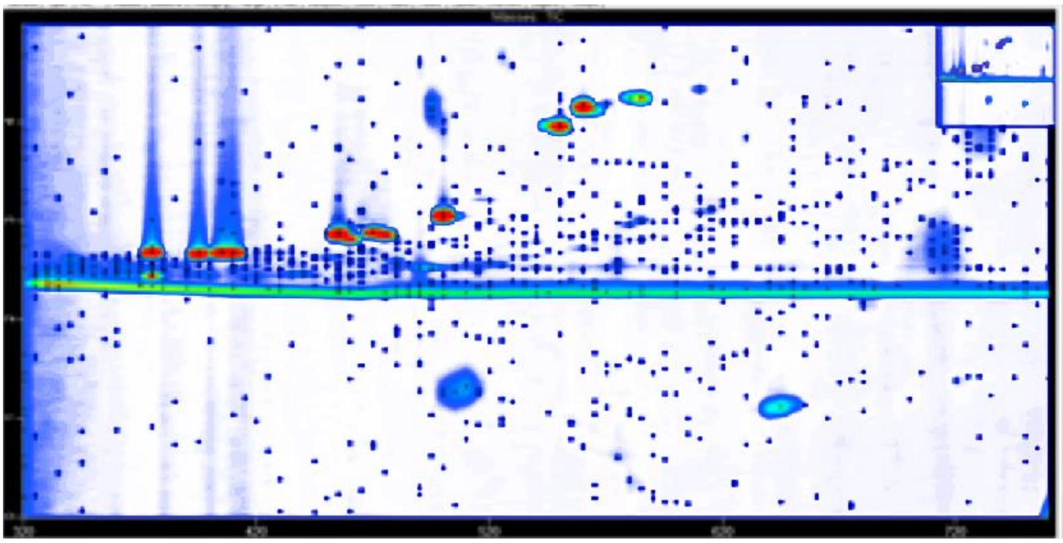


Figure 41 “fast GCxGC” a fast increase in the oven temperature by setting a ramp of 40 °C/min. The separation between these compounds has been reduced both in the 1st and 2nd dimension. The x- axis shows the primary retention time in seconds and the y-axis shows the secondary retention time in seconds.

Further development on creating a short method was carried out using a GCxGC – NCD and a more methodical approach to decreasing analysis times was undertaken with the NCD.

Method 1: The primary GC column was a BX5 (30 m x 0.32 mm x 0.25 µm). The secondary GC column was a BX50 (3 m x 0.1 mm x 0.1 µm). The primary oven was programmed from 40 °C (2 min) at 7 °C/min to 270 °C (5 min). The secondary oven was programmed from 70 °C (2 min) at 7 °C/min to 300 °C (5 min). A dual jet liquid nitrogen modulation system was used with a 5 second modulation period. Helium carrier gas was used at a flow rate of 1 mL/min. 1 µL of sample was injected using a Gerstal auto-sampler in splitless mode.

Using method 1 all explosives were detected in less than 34 minutes with good separation between the peaks and only a slight shift in position in the second dimension, this can be observed in Figure 42.

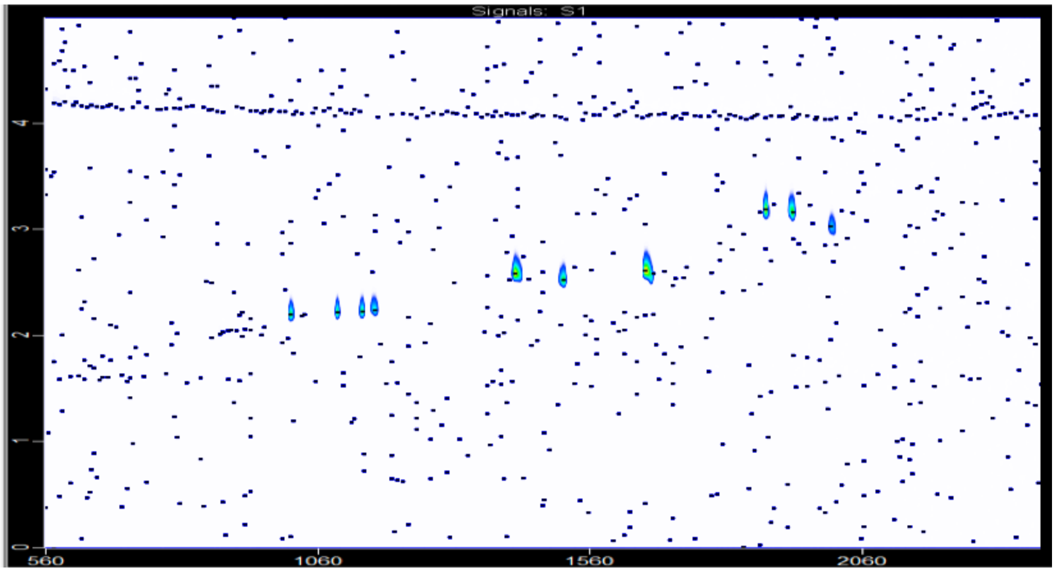


Figure 42 1 ppm of the 8330 standard analysed using GCxGC-NCD. Only nitrogen containing species are observed using NCD. The peaks are sharp and there is limited wrapping. The x- axis shows the primary retention time in seconds and the y-axis shows the secondary retention time in seconds.

Method 2: The primary GC column was a BX5 (30 m x 0.32 mm x 0.25 µm). The secondary GC column was a BX50 (3 m x 0.1 mm x 0.1 µm). The primary oven was programmed from 40 °C (2 min) at 10 °C/min to 270 °C (5 min). The secondary oven was programmed from 70 °C (2 min) at 10 °C/min to 300 °C (min). A dual jet liquid nitrogen modulation system was used with a 5 second modulation period. Helium carrier gas was used at a flow rate of 1.4 mL/min. 1 µL of sample was injected using a Gerstal auto-sampler in splitless mode.

Using method 2 the same standard was analysed, and the analysis was completed in less than 25 minutes. This can be observed in Figure 43. It is clear that the peaks were not as sharp and started to become wider in the second dimension.

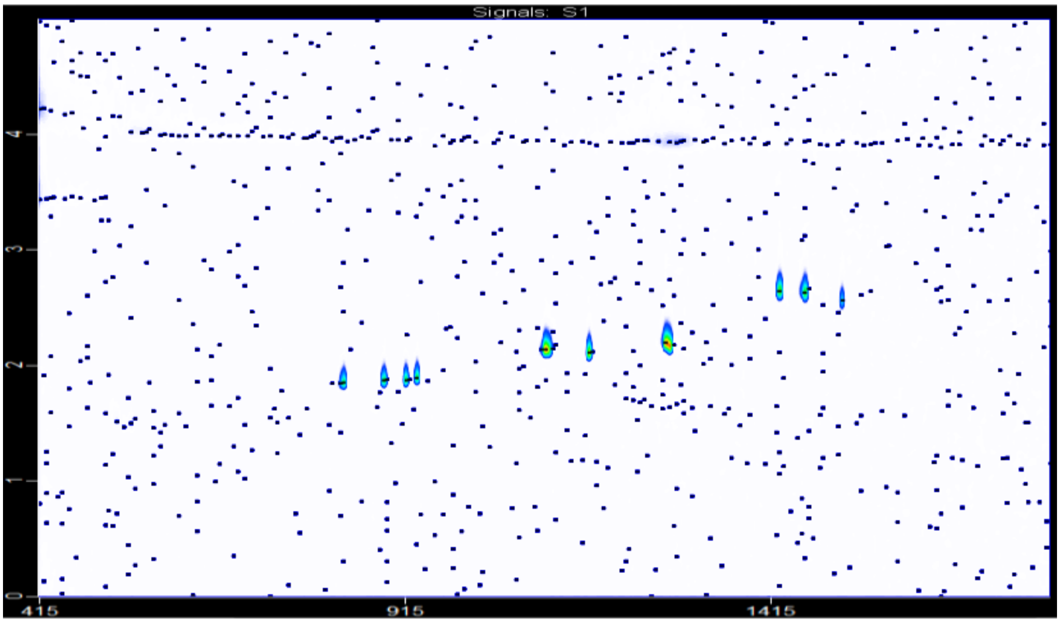


Figure 43 1 ppm of the 8330 standard analysed using method 2. The x- axis shows the primary retention time in seconds and the y-axis shows the secondary retention time in seconds.

The Figures above demonstrate that it is possible to get the analysis time below 6 minutes, but this is at the expense of peak shape and clarity, as by just simply ramping the oven there is a significant drop in separation efficiency. Taking a more targeted approach and changing column dimensions allowed for a 25 minute method to be reach that still has excellent separation and good peak shape. Further research needs to be carried out to continue decreasing the analysis time this will include, amongst other things, decreasing the length of column or reducing column ID and increasing flow rates along with a suitable oven ramp rate.

Although 25 minutes is a significant reduction in time, there is still a need for a fast and portable system. Development has taken place to produce a prototype system to allow for portable detection.

4.3.2 Portable GC

In order to produce a “portable” system several components of the GC need to be vastly reduced in size. To try to achieve that, a miniature GC has been developed with the University of York that uses a band heater to heat the column up to 300 °C in less than 120 seconds. The system itself is approximately the size of a roll of duct tape and contains 15 m of column, with the possibility of containing 25 m of column if required.

The GC was only tested under isothermal conditions holding the system at 140 °C, with the outer oven at 100 °C and using FID for detection. 1 ppm of the 8330 standard, described earlier, was analysed using the system and some of the analytes that can be detected with FID were separated out in around 5 minutes with relatively good peak shape, Figure 44. As the run reaches 10 minutes the peak shape is poor, this could be due to the isothermal conditions and the flow rates.

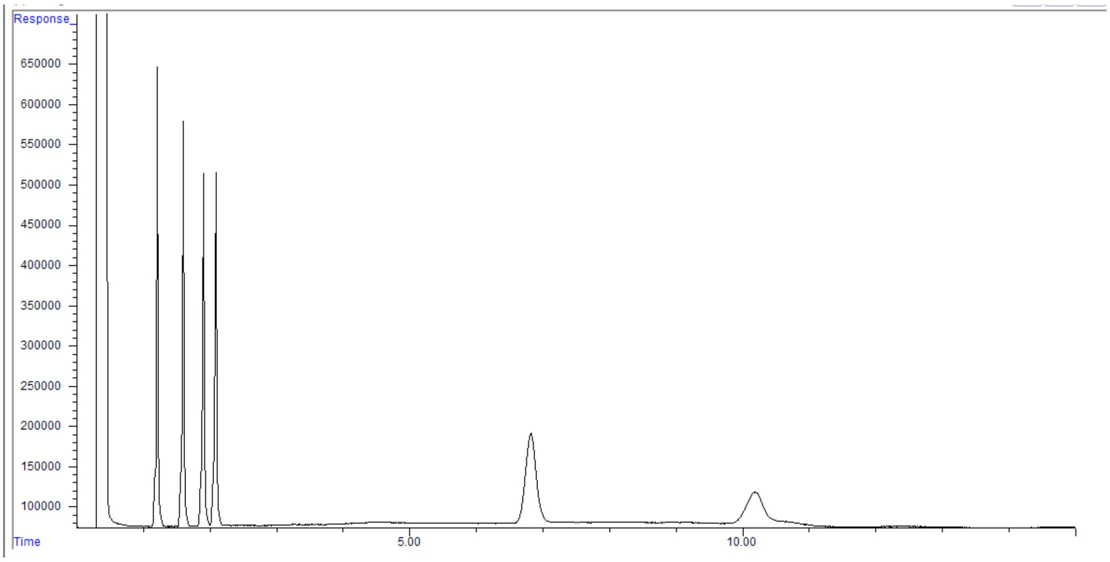


Figure 44 1 ppm of the 8330 standard separated out under isothermal conditions using the micro GC

The system was using the external oven to heat the transfer lines between the inlet and the FID. Heated transfer lines were developed using 1/16 inch copper wrapped in heater wire and Kapton tape. This was then connected to a power supply and set to produce the required temperatures by varying the voltage, the temperature was measured by using an external thermocouple.

Figure 45 details the results for the 1 ppm of the 8330 standard using the same GC method as before but using the new heated transfer lines.

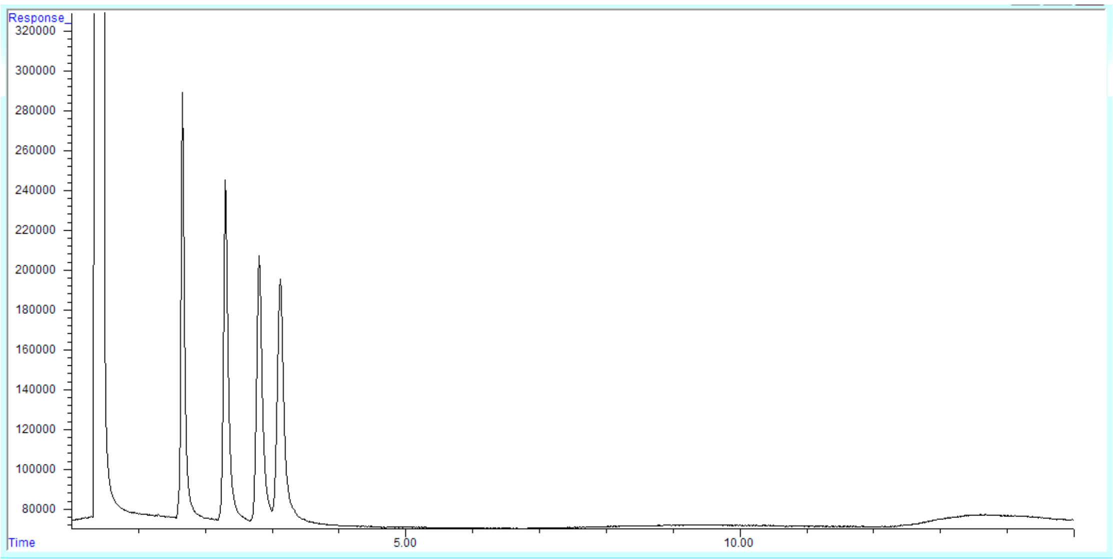


Figure 45 1 ppm of the 8330 standard separated out under isothermal conditions using the micro GC and heated transfer lines.

Figure 45's chromatography is not as good as that of Figure 44, however, this could be due to uneven heating of the transfer lines, variability of the pressure in the GC oven between the two runs or that 80 °C was not the correct temperature for the transfer line. The transfer lines are now capable of reaching 200 degrees, which should reduce the possibility of a cold spot.

Heat imaging of the transfer lines was undertaken to determine if the heating is even or not, Figure 46.

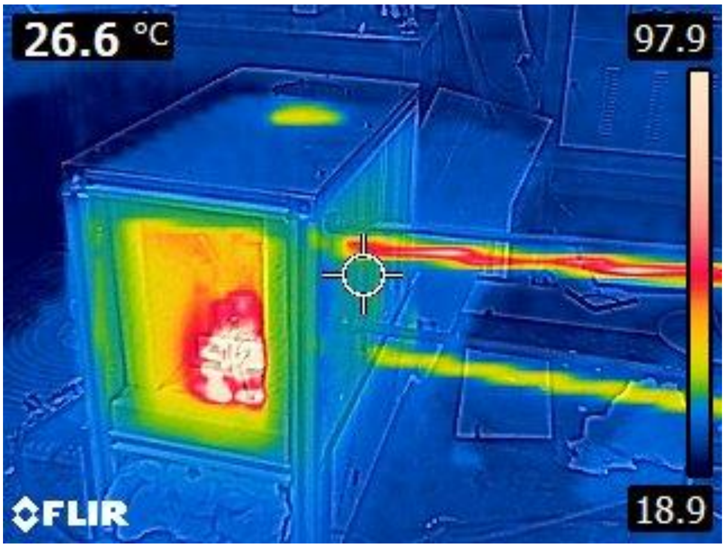


Figure 46 Heat image of transfer lines and valve. The colour scale is presented on the right hand side and ranges from 18.9 to 97.9 degrees

It is clear in Figure 46 that the top transfer line which runs to the detector has hot spots within the system, represented by the white areas and that is has cold areas represented by the yellow. Re wrapping of the heater wire around the transfer line was carried out to try to produce more even heating, Figure 47.

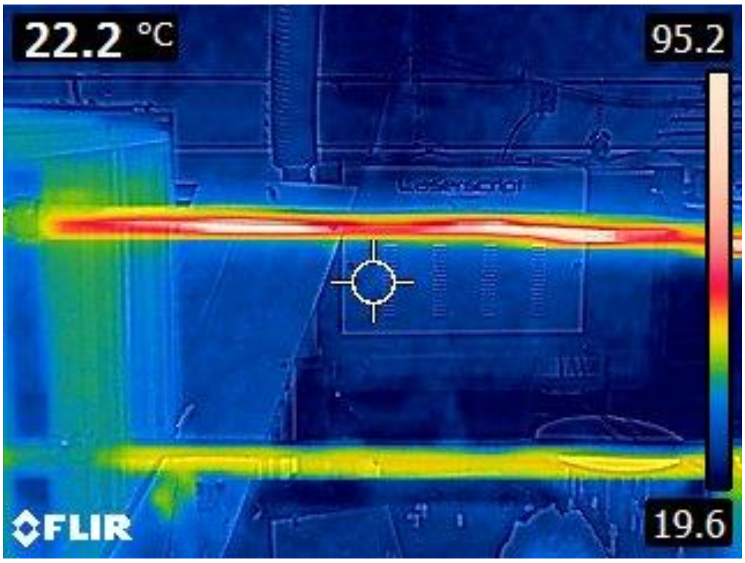


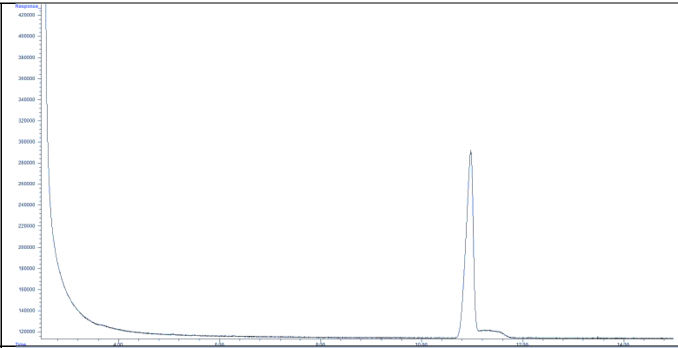
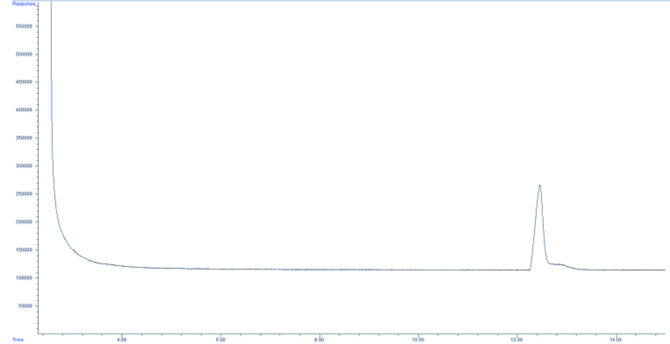
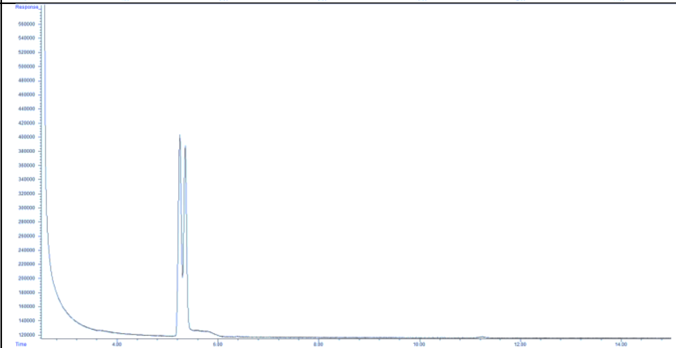
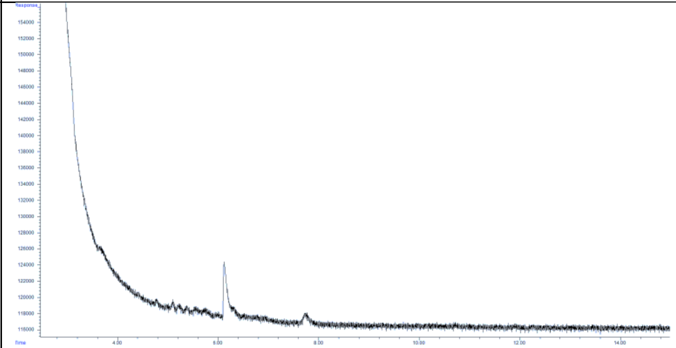
Figure 47 Heat image of transfer lines. The colour scale is presented on the right hand side and ranges from 19.6 to 95.2 degrees.

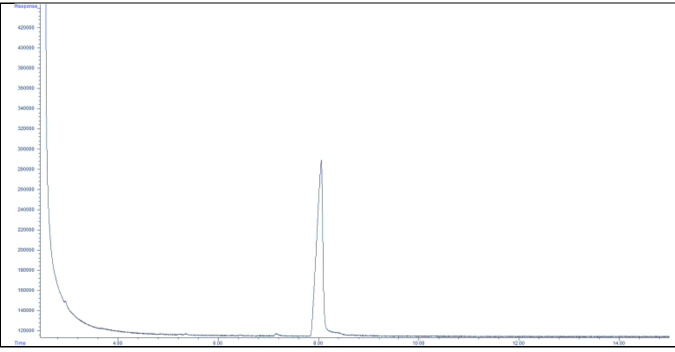
There is a difference in the final temperature between Figure 46 and Figure 47 but the error on the heat imager accounts for this difference. In Figure 47, there does appear to be more even heating but there are still two hot spots demonstrated by the white colour. It is also notable that the connection into the box is only warm and could act as a cold spot. To try to achieve even heating the system is allowed to equilibrate for 30 minutes before testing commences.

4.3.2.1 Portable GC-FID CWA Data

Five chemical agents were tested on the system separately to determine if they could travel through the transfer lines and the columns. The resulting chromatograms can be observed in Table 25. It is clear that there is a shoulder appearing after each peak, which is indicative of poor chromatography likely due to column positioning or column degradation. The agents were introduced at a relatively high concentration of 50 µg/mL but this concentration is not normally expected to have a large tailing effect. However, the system was held under isothermal conditions at 125 °C with the transfer lines at 60 °C and 130 °C respectively which could have caused an issue or there could have been a problem with the inlet.

Table 25 Results table of chemical agents in portable GC

Mustard (HD)	
GF	
GD	
GB	

GA	

4.3.3 Portable GCxGC

4.3.3.1 Design

Using the micro GC as the building block a GCxGC system was designed in order to fill the requirement for a fully portable system. The entire system in its most basic design is 12 cm x 25 cm x 22 cm, a schematic of the system is shown in Figure 48.

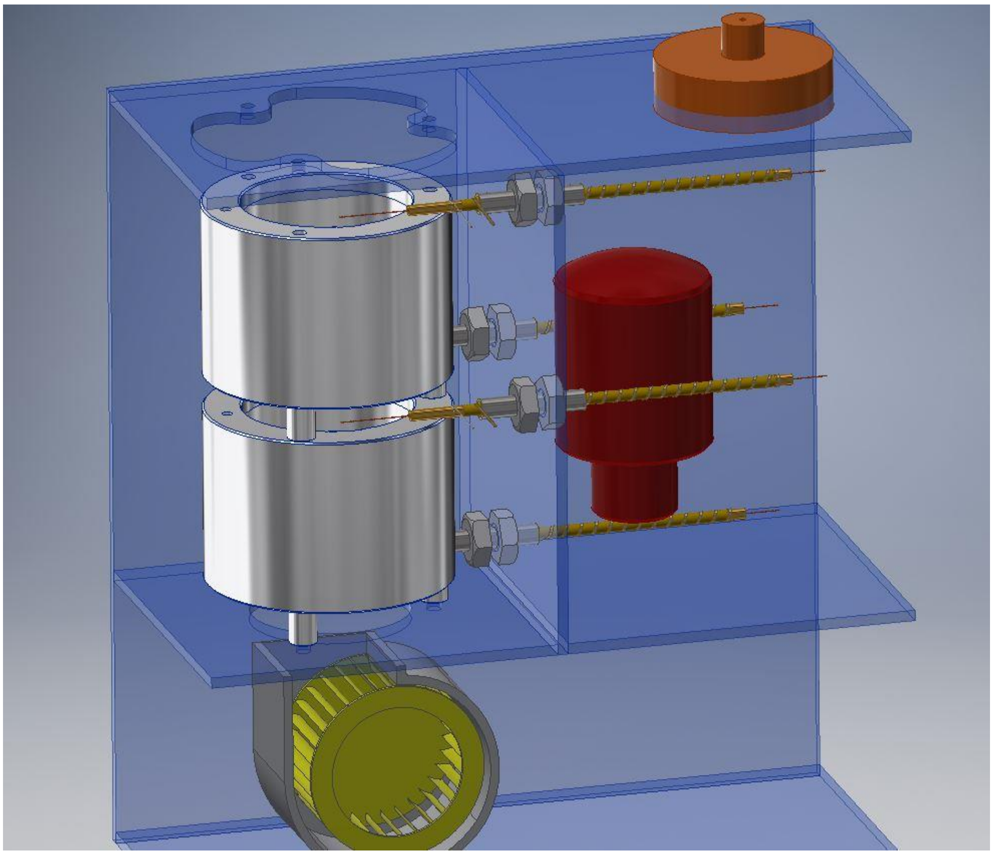


Figure 48 Schematic of the micro GCxGC

Working closely with the University of York allowed for the development of a man-portable if not transportable GCxGC system. The University provided the Engineering and Electronic know-how, Timothy Ayers helped in the build and electronic coding. The “fully” portable system required the design of a portable small GC unit, using a heater band to heat the column (as described above). The GCxGC system used two of these ovens stacked one on top of the other. There were insulated ports through to the valve to allow the columns to pass through with minimal temperature change. The valve itself was heated to a constant temperature. In developing the prototype, there was not enough time to place an inlet and detector into the system so heated transfer lines (as described above) are used to allow the column to pass between areas. Control of the system used Arduino Nano’s to keep the cost low. This was arranged in a master/slave system to control all of the temperatures. The final product and set up can be seen in Figure 49.

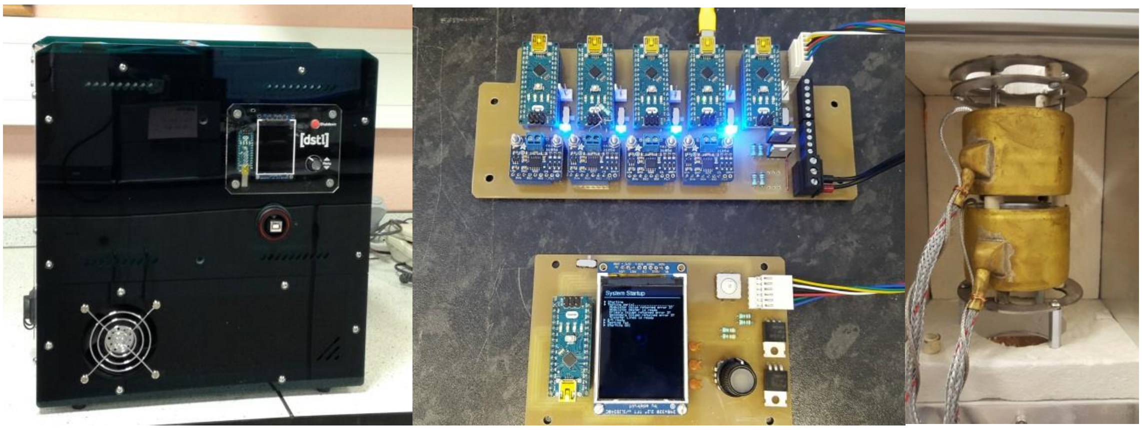


Figure 49 From left to right- Final GCxGC unit WxDxH: 34x20x36 cm, the Arduino Nano set up using a master at the bottom with a screen to allow for a friendly interface and the 5 “slaves” at the top and finally the two GC columns and ovens.

The idea was not only to produce a fast and small system but to keep the costs low. There were some compromises in order to keep the costs low e.g. using non-ideal materials, a small screen, and bulky insulation. The system itself was slightly larger than the original design but this is mostly due to using very bulky insulation. In a future design, more thought would be put into the structure itself and insulation requirements.

In terms of heating the code was capable of controlling the two ovens with an offset with relatively high precision. There was slight oscillation around the ramping temperature but this was to be expected due to how power was supplied to the system. Otherwise it was able to ramp very quickly in temperature and the main issue was attempting to cool the ovens quick enough.

4.3.3.2 Heating

Observing the ovens in more detail, thermal imaging was undertaken to observe the valve and the small ovens. There were some issues with this imaging as the ovens were slightly reflective, but the images were simply to provide an overview of the evenness of the heating, Figure 50.

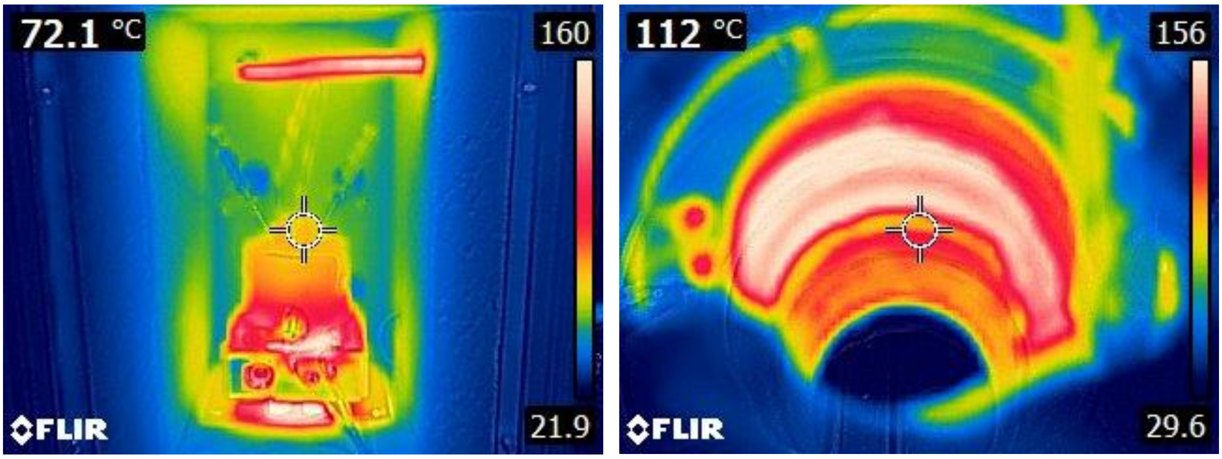


Figure 50 Left represents the valve heating and the connectors into the valve. Right is the inner of the small ovens.

Starting with the valve in Figure 50, it is clear that the base of the valve is heating up significantly and the top of the valve is warm but the outer connections for the column are green suggesting that they are cold and that heat is not being circulated around or creating an oven. If the system is left to equilibrate for 30 minutes or more, the outer connections reach 100 + degrees. On the right in Figure 50 it is clear that the insides of the oven are heating up very quickly and little heat is dissipating to the edges, the oven was more difficult to image as it was metal and not covered so it has not been imaged very well. Again, there does appear to be hot spots represented by the white areas.

4.3.3.3 Complications

As with any prototype there have been a few modifications that have had to take place during testing as the system was not working correctly. This has included;

- Modification of the controlling software – the system would attempt to cool and heat at the same time after one run. This was simply a few lines of code that required changing – see Annex 2 for further details of code and the modified lines.
- The secondary oven column had a hair line fracture from when it was potted into the holder. This was replaced with a new line of column.
- The internal connectors were too large and causing dead volume, smaller zero volume connectors were placed into the system, these are also deactivated.
- Zero dead-volume unions were replaced with silco treated unions so that activate species can be tested in the system
- Cold spots present at the inlet and outlet connections – the connections were insulated

Even with these modifications cold spots/active surfaces were present in the system. This was observed with the following samples.

The alkane standard of C7 to C40 was analysed and only four of the alkanes could be observed. These are widely spread in the second dimension and wrapping. This commonly occurs when a cold spot is present or when an active surface is present. This is demonstrated in Figure 51.

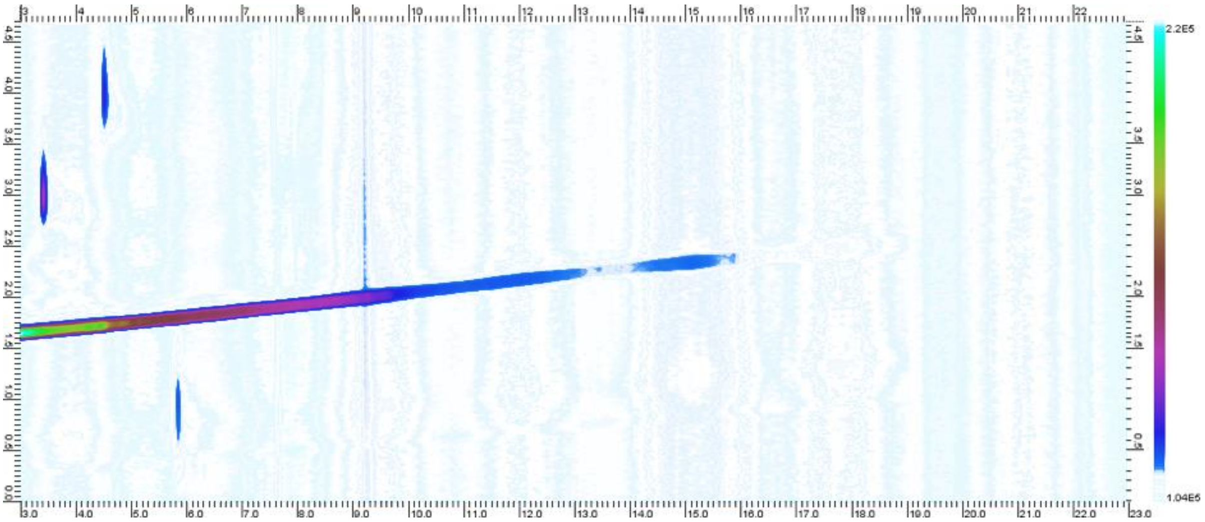


Figure 51 Chromatogram of C7 to C30 alkane standard. Only three of the compounds can be observed – these are broad and starting to wrap. The x- axis shows the primary retention time in seconds and the y-axis shows the secondary retention time in seconds.

From this data, the method was developed to allow further compounds to elute through the system and using a series of methods listed in the experimental (all data included in the Annex). To produce the final method, method 5, which allowed for separation of five alkanes, Figure 52, but the higher boiling species were still not making it through the system as they were either getting trapped as cold spots or there was a leak in the valve.

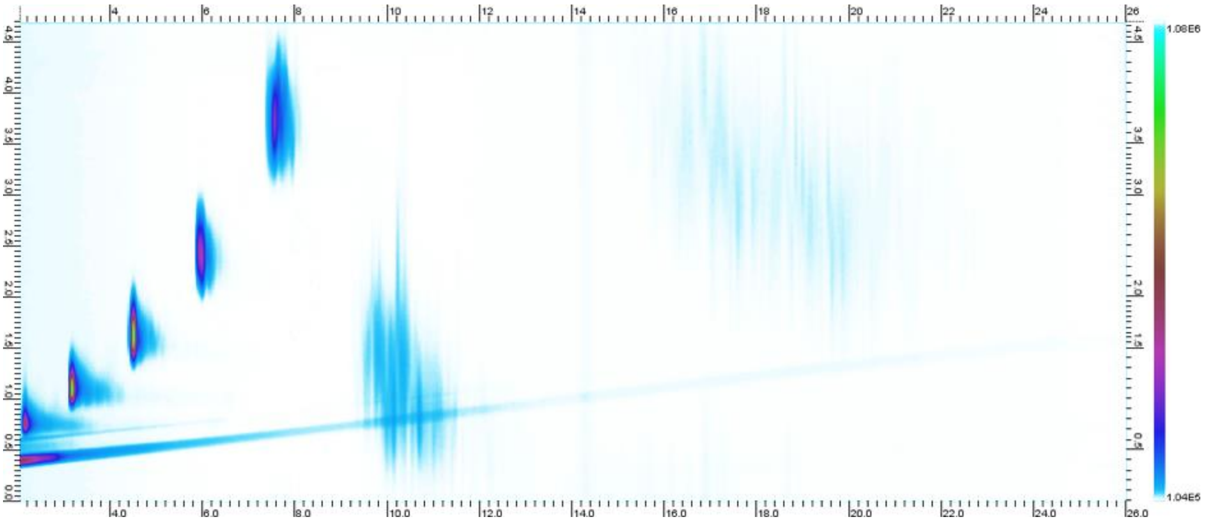


Figure 52 Separation of five of the alkanes in the C7-C30 standard. The x- axis shows the primary retention time in seconds and the y-axis shows the secondary retention time in seconds.

The data presented above describes that there was an issue in the system likely related to cold spots being present or hot spots or another issue. The system was redesigned in an attempt to counter act this problem. The results below describe the design change and the results of the changes.

4.3.4 Second Heater

During the cycling of the instrument, the temperature inside the area where the valve sits was measured. The temperature sits at 98 degrees and does not increase over time. This could explain why the larger molecules are not being separated.

A secondary heater was added into this region, Figure 53, the heater is controlled by an external OMEGA fast control PID (see Annex 3 for a picture of the system). The design of the green metal observed in Figure 53 is to help with heat dispersion in the region. The valve heater remains functional at 250 degrees and the secondary heater was set to 200 degrees for testing.

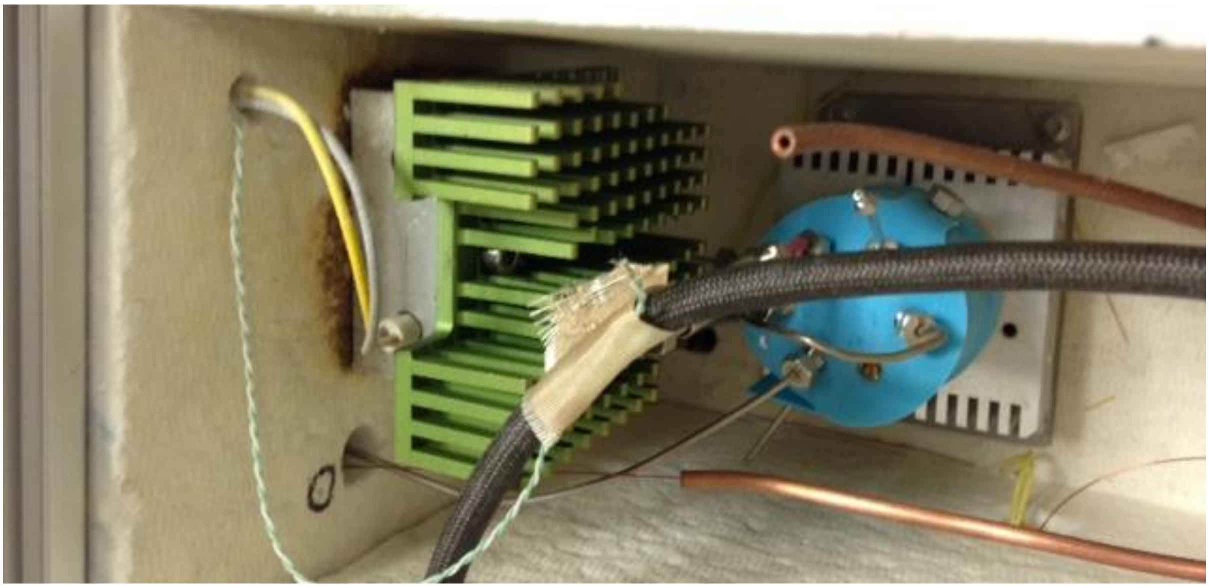


Figure 53 Area inside the valve oven. The green metal piece is the secondary heater that has been added to try to keep the area hot to allow compounds to pass through the system.

The system was run using the same method conditions as previously stated. The only change to the system was the sample loop, going from 25 µl to 10 µl due to availability of sample loops. This required a change in the modulation time to 1 second.

A C7-C30 standard was analysed using the system, Figure 54, a few more species were detected giving a total of 7 species detected. This result is also repeatable and the chromatograms are in Annex 3.

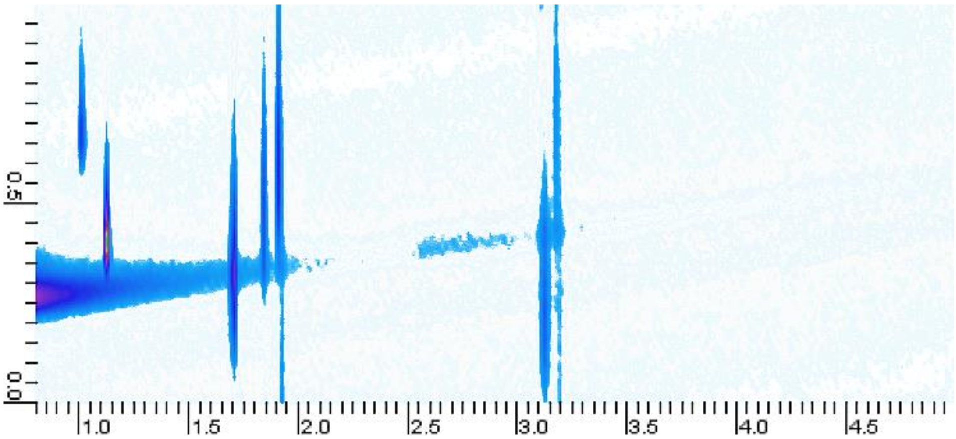


Figure 54 Chromatogram of C7 – C30 at 1 µg/mL analysed on the micro GCxGC unit with a secondary heater held at 200 degrees

As can be observed in Figure 54 there is some 2D separation and the peaks are wrapping. The system could still have a cold spot somewhere or an active surface, which is causing the loss of the rest of the compounds in the sample. To continue to test the system, a C7-C30 standard spiked with aromatics species was analysed, Figure 55.

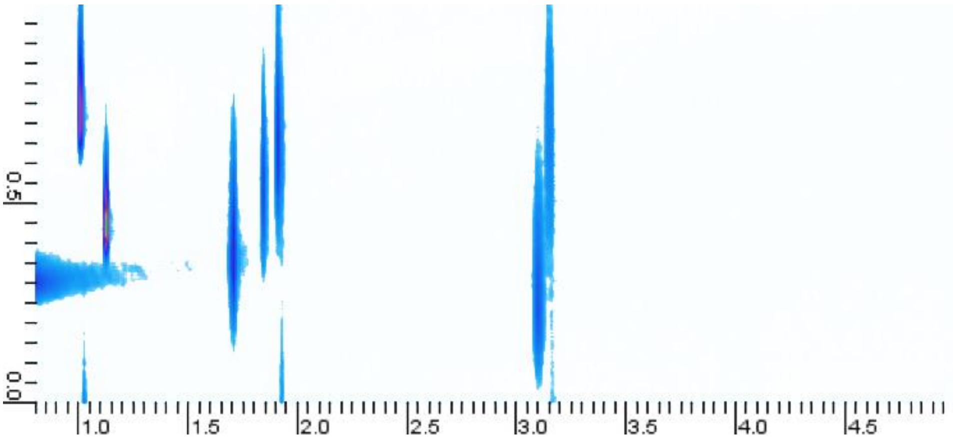


Figure 55 Chromatogram of C7 – C30 at 1 µg/mL spiked with aromatic species analysed on the micro GCxGC unit with a secondary heater held at 200 degrees

Figure 55 and Figure 54 are almost identical suggesting that the larger less volatile aromatics are not making it through the system. It is likely that they are getting stuck on a cold and or active surface or that the flows are still not correctly balanced. However, this result was repeatable, and the other chromatograms can be found in Annex 3.

4.3.5 Conclusions

This work has allowed the portable GCxGC system to be developed and tested. During this process issues have been found in relation to the controlling code and simple issues such as hairline fractures in the column. These have been corrected but the largest issue that still remains is the heating. An engineering approach needs to be taken to the project to find and correct the cold spots.

Chapter 5

Conclusions

5 Conclusions and Future Recommendations

Comprehensive gas chromatography has demonstrated that it can be highly beneficial to the defence sector throughout this research. A new capability for detection of threat materials has been developed for analysis in complex mixtures and this thesis has demonstrated that it could become field portable.

The research detailed in this thesis can be broken down into two distinct areas of defence applications; defence intelligence analysis and in-field analysis. The following sections, 5.1 and 5.2 provide a summary to each area and the future recommendations of research.

5.1 Defence Intelligence Analysis

5.1.1 Conclusion

Within the defence intelligence networks, there is an ongoing requirement to be able to obtain further information from a sample that has returned from areas of interest. Standard practice dictates that the sample will be analysed to determine which, if any, explosive or CWA has been used and what quantity can be found at the site in question. If we take a post-blast environment, the samples collected are in highly complex matrices. These matrices can contain anything from oils, fuels, blood and dust to deposited particulate from cars and weather events. The process for analysing these samples requires the suspected material of interest to be removed from the matrix. This is commonly performed using solid phase extraction (SPE). The problem with SPE is that depending on the chemistry of the target analyte, mass can be lost during the process (10 % return) or in some cases the target is removed completely. This is not a problem in the case of CWA detection, as they are not naturally found in the environment. However, at some locations there is a low concentration of explosives present which means that the significant detection level must be set above this to determine that an explosive has been used and was not present in that environment in the first instance.

This research has used cryogenic GCxGC-TOF-MS to analyse explosives and CWAs in representative matrices for operations such as a post-blast event. The samples required no clean up but a small split was used on the inlet to reduce contamination in the system. The technique proved highly useful in analysing these samples without the need for cleaning the samples, and picogram detection levels have been achieved.

The GCxGC-TOF-MS provides four dimensions of data and high separation, which means that not only can the analyte of interest be determined but it is also possible to identify other compounds within that matrix that could provide other evidential value. For example, if the sample is collected from a road, it is possible to determine if diesel, petrol or other fuels are present; if an accelerant is present this would also be identifiable thus providing further evidence for the investigation.

The methods presented here could provide simply a pre-screening device or evidential detection and identification. There are obvious drawbacks using this technique: it is slow, requires cryogenics, and the sample must be split to avoid high levels of contamination reaching the detector.

5.1.2 Future Recommendations

The research into using GCxGC-TOF-MS for defence purposes has not previously been explored. Future work should look to not only run samples that have returned from operations and other urgent requirements but to begin to research into using headspace analysis to obtain different information from the sample.

Drug analysis should also be explored as it has already been demonstrated within the literature that GCxGC-TOF-MS can provide further detail on drug composition compared to lower resolution techniques such as GC-MS. Therefore, it should be possible to obtain information that would aid the criminal justice system.

The technique should also be validated under UKAS conditions to allow for the analysis of operational samples as it would provide orthogonal data to data often collected.

5.2 In-Field Analysis

5.2.1 Conclusions

When an attack has occurred using explosive or CWAs there are several different responses that take place. Only two are reviewed during this work; firstly, that samples are returned to an analytical laboratory for testing to provide evidence for the defence network and criminal justice system and secondly is the response of service personnel in the field i.e. how to respond to the hazard.

An attack taking place is not the only time that service personnel may be required to undertake chemical analysis in the field. If they are on an operation and discover an area of interest, for example a clandestine laboratory, samples are taken and sent back for analysis. In the meantime, the personnel must decide how to proceed and what chemical hazards are present. GCxGC-FID could provide a portable system that could be used for analysis in the field alongside samples sent back for analysis. It also provides the possibility of being able to analyse samples that may not make it back to the support base.

In this thesis research, has been undertaken looking at a modified Agilent 6890 and a prototype man portable system. The valve system demonstrated in chapter 3 is compared with the cryogenic system for the detection of explosives in Table 26 (a similar trend to that seen in Table 26 also exists for CWA detection). It is clear in the table that there is loss of sensitivity moving from the cryogenic system to the valve system. However, the valve system is more likely to be utilised in an area where a higher concentration of explosives may be present and so the loss of sensitivity may not be a problem.

Table 26 Comparison of the LOD for explosives in the valve and the cryogenic system

Compound	LOD (µg/mL) Valve	LOD (µg/mL) Cryogenic
2-nitrotoluene	0.28	0.0059
3-nitrotoluene	0.41	0.0081
4-nitrotoluene	0.29	0.006
1,3-dinitrobenzene	0.26	0.015
2,6-dinitrotoluene	0.05	
2,4-dinitrotoluene	0.78	0.28
1,3,5-trinitrobenzene	1.25	0.012
2,4,6,-trinitrotoluene	0.59	0.021
Tetryl	n/a	0.47
Nitrobenzene	n/a	0.24
4-amino-2,6-dinitrotoluene	n/a	0.083
2-amino-4,6-dinitrotoluene	n/a	0.073

The modified 6890 GC to a GCxGC-FID has shown significant promise that it is capable of detecting most compounds of interest to defence, with a few explosives as exceptions due to their chemical activity in the valve, Table 26. Explosives such as TNT have a relatively high boiling point and they will "stick" to active surfaces such as stainless steel and any cold spots. During this research, TNT could only be detected in the valve above 0.59 µg/mL below this concentration no peak could be seen for TNT. Other higher boiling point explosives wrapped in the secondary dimension and were only detectable at high concentrations (above 5 µg/mL). This suggests that active surfaces are present in the valve and when a high enough concentration is analysed on the system, the active surfaces are filled, and some material is able to pass through and be detected. This research did explore the changing the temperature of the inlet to reduce break down of the lower volatility explosives successfully and the use of silcosteel coating to reduce the activity in the valve. The silcosteel coating provided a minor improvement but it still requires further investigation into any possible active sites on the diaphragm.

In comparison, all CWAs were detectable using the valve except agent T that was present but due to its high boiling point it wrapped in the secondary dimension and was not detectable at lower concentrations due to its affinity for active surfaces.

A GUI was developed using Matlab which enables non-scientific users to determine firstly if an explosive or CWA is present and secondly what agent/explosive is present as each one occupies its own separation space. If the method for analysis is modified or if the column lengths are reduced, the GUI model will need to be modified as it is based on bounding boxes around the area where the peak is detected for the analyte of interest (based on a minimum of 5 repeat spectra).

The next stage in building a field portable system was to take the lessons learnt during the production of the 6890 GCxGC-FID system and apply them to build a man-portable system. A prototype portable system was developed and tested in two stages.

The first stage was to make a small GC oven. This was completed using a small heater band wrapped around a circular piece of stainless steel that had been designed to house a column. This design was very successful and could analyse CWAs quickly with good peak shape.

Taking this design, a full GCxGC system was built. This was more problematic to build as there are two independent ovens for the GC columns and a secondary oven to house the valve so that is it is not cooled in between runs. When testing the system with a hydrocarbon standard of C7 to C30 only the first seven hydrocarbon species were detected. They were also offset from one another which would suggest that either the secondary oven was not providing an isothermal separation or that the flows were slightly unbalanced for the valve. The flows were investigated by modifying the primary and secondary flows but a better method could not be achieved, suggesting that there is a secondary effect taking place. The testing also demonstrated that cold spots must be present in the system as only the first seven hydrocarbons were detected. A secondary heater was added to the valve region and this did slightly improve detection but for further improvements a redesign of the system would be required.

5.2.2 Future Recommendations

Further research should be undertaken to improve the man portable system for field analysis. Hydrogen fuel cells could be used to power the system and research should take place to determine if, similar to the Thermo EGIS defender, the system could run off a scrubbed air pump instead of helium. This would help to produce a fully man portable system that does not need mains power or a cylinder of helium. Engineers should also re-view the design to determine where weight could be removed to make the system much lighter. In order to address the cold spots, a thermal imaging camera should be used to record each of the chambers while the system is running to firstly determine where the cold spots are. Once they have been located, heating wire or cartridges should be used to increase the heating in those areas. If the cold spots cannot be located, a redesign of the system should take place where everything, columns and valve, are in one box. The system should run isothermally with different isothermal methods for different compounds of interest.

Annex 1

I. CWA calibration data FID

It should be noted that if numbers are missing in the tables, the agent could not be detected at that concentration for the repeat. This applies to all data in this section

Table 27 Mustard (H) calibration data

Concentration (µg/mL)	1	2	3	4	5	Average	Stdev	Rsd (%)
0.50	1989310	4970072	2195040	1570547	1398491	2424692	1458073	60
0.75	1756688	2237238	1515638	3550024	4619171	2735752	1313970	48
1.00	2664467	3446074	3413933	3240689	4508933	3454819	667722.3	19
2.50	10197362	9100383	10512311	10657700	10033828	10100317	611171.3	6
5.00	18337972	21637753	23988539	27155093	31792900	24582451	5163007	21

Table 28 HN3 calibration data

Conc (µg/mL)	1	2	3	4	5	Average	Stdev	Rsd (%)
0.50	1712205	1650289	2564898	3500483	3536582	2592891	919125	35
0.75	3211495	3433294	3887844	5132944	6433392	4419794	1349009	31
1.00	4074278	5658000	4055751	6550478	6685367	5404775	1285181	24
2.50	14200744	14326077	14800745	14880850	14119352	14465554	351517	2
5.00	25536327	31050658	34378609	38985550	46116640	35213557	7822736	22

Table 29 Mustard T (T) calibration data

Conc (µg/mL)	1	2	3	4	5	Average	Stdev	Rsd (%)
0.50			289991	302359	136114	242821	92618	38
0.75	180548	232101	1412996	941393	1515343	856476	632011	74
1.00	500849	778220	393808	1360628	1341184	874938	456635	52
2.50	792876	7697876	8501917	9796774	10379311	7433751	3859150	52
5.00	6642573	21173790	25166768	28596496	35087040	23333333	10635519	46

Table 30 GB calibration data

Conc (µg/mL)	1	2	3	4	Average	Stdev	Rsd (%)
0.20					0	0	0
0.50	208569			220964	214767	8765	4
0.75	596009	604413	333336	322954	464178	157172	34
1.00	1470155	1758111	760886	1174862	1291004	426145	33
2.50	12316879	8564204	10365243	8628915	9968810	1773774	18
5.00	22180557	20023106	21433128	22055403	21423049	988877	5

Table 31 GD1 calibration data

Conc (µg/mL)	1	2	3	4	Average	Stdev	Rsd (%)
0.20							
0.50				158409	158409		
0.75	323274	297661	324728	294927	310148	16047	5
1.00	700767	1115819	462104	841003	779923	273149	35
2.50	7527107	7328000	3736104	5561909	6038280	1770775	29
5.00	8048411	12461400	17872073	12841869	12805938	4017453	31

Table 32 GD 2 Calibration data

Conc (µg/mL)	1	2	3	4	Average	Stdev	Rsd (%)
0.20							
0.50							
0.75				241985	241985		
1.00	379755	1062295	283388	212536	484494	391250	81
2.50	8902764	3206255	5370280	4535674	5503743	2434938	44
5.00	13742518	9459751	20795124	10129501	13531724	5194789	38

Table 33 GA calibration data

Conc (µg/mL)	1	2	3	4	Average	Stdev	Rsd (%)
0.20							
0.50		161790	153427	259147	191455	58772	31
0.75	662398	970784	904884	1397374	983860	305908	31
1.00	975509	2327504	1160824	1936808	1600161	639157	40
2.50	7381814	7616220	5687949	6147707	6708423	936896	14
5.00	11544760	13131456	11017068	12809143	12125607	1007504	8

Table 34 GF calibration data

Conc (µg/mL)	1	2	3	4	Average	Stdev	Rsd (%)
0.20							
0.50	308498	499805	157217	501021	366635	166363	45
0.75	2449538	987946	1439034	1618749	1623817	611097	38
1.00	3458999	3651717	2908067	3589229	3402003	338936	10
2.50	15956108	13825468	12621506	10621227	13256077	2233046	17
5.00	21841778	22541993	21202542	23069444	22163939	814644	4

Table 35 VM calibration data

Conc (µg/mL)	1	2	3	4	Average	Stdev	Rsd (%)
0.20							
0.50		322588	159348	522320	334752	181791	54
0.75	1383057	1649801	2088581	1648669	1692527	292335	17
1.00	1548233	3600388	3314587	2829499	2823177	907572	32
2.50	12295952	12704615	10991747	7866248	10964641	2190941	20
5.00	17093944	19155730	17274924	20083471	18402017	1457940	8

Table 36 VX calibration data

Conc (Vg/mL)	1	2	3	4	Average	Stdev	Rsd (%)
0.20		105839			105839		
0.50	421471	1477120	1038892	982178	979915	270871	28
0.75	1081754	2931328	2334212	2345555	2173212	779240	36
1.00	2056507	5110474	4075658	4324246	3891721	1300549	33
2.50	20377073	17380672	15226068	12709923	16423434	3254300	20
5.00	23708446	24602699	23388587	25728724	24357114	1048879	4

II. GCxGC-TOF CWA Calibration Data

Table 37 GB calibration

	Repeats					
Conc (µg/mL)	1	2	3	Average	Standard Deviation	Relative sd (%)
1.00	2732486	3051328	4561112	3448309	976813	28
0.75	2200604	3666620	767895	2211706	1449394	66
0.50	2071692	2037094	1153151	1753979	520620	30
0.20	595986	934831	238906	589908	348002	59
0.10	349071	125434		237253	158135	67
0.05	60916	19349		40133	29392	73

Table 38 GD peak 1 calibration

	Repeats					
Conc (µg/mL)	1	2	3	Average	Standard Deviation	Relative sd (%)
1.00	2783385	4858943	4123077	3921802	1052316	27
0.75	1474198	2160731	2431353	2022094	493408	24
0.50	1289356	1587528	1732061	1536315	225752	15
0.20	728422	558330	665179	650644	85973	13
0.10	364482	246688		305585	83293	27
0.05	151038	171475		161257	14451	9

Table 39 GD peak 2 calibration

	Repeats					
Conc (µg/mL)	1	2	3	Average	Standard Deviation	Relative sd (%)
1.00	2162738	3798252	3515693	3158894	874188	28
0.75	1433870	1711596	2531343	1892270	570608	30
0.50	930023	1234279	1185439	1116580	163398	15
0.20	577066	455785	388340	473730	95634	20
0.10	281442	180426		230934	71429	31
0.05	80603	136592		108598	39590	36

Table 40 GA calibration

	Repeats					
Conc (µg/mL)	1	2	3	Average	Standard Deviation	Relative sd (%)
1.00	1379085	2878191	2274484	2177253	754268	35
0.75	726470	980109	668675	791751	165662	21
0.50	604960	706939	684958	665619	53670	8
0.20	283482	239966	409229	310892	87898	28
0.10	132602	174986		153794	29970	19
0.05	97963	126272		112118	20017	18

Table 41 GF calibration

	Repeats					
Conc (µg/mL)	1	2	3	Average	Standard Deviation	Relative sd (%)
1.00	5658733	6633578	6127021	6139777	487548	8
0.75	3382826	4836892	4736347	4318688	812038	19
0.50	2036708	3426543	2451937	2638396	713432	27
0.20	729461	1006511	1374803	1036925	323744	31
0.10	361072	238846		299959	86427	29
0.05	136950	150699		143825	9722	7

Table 42 VM calibration

	Repeats					
Conc (µg/mL)	1	2	3	Average	Standard Deviation	Relative sd (%)
1.00	636097	978820	903024	839314	180025	21
0.75	309151	507387	696048	504195	193468	38
0.50	233727	310733	382790	309083	74545	24
0.20	99759	55145	92138	82347	23864	29
0.10	55464	7500		31482	33916	108
0.05	14748	3616		9182	7872	86

Table 43 VX calibration

	Repeats					
Conc (µg/mL)	1	2	3	Average	Standard Deviation	Relative sd (%)
1.00	963574	1519889	1388170	1290544	290723	23
0.75	721694	864916	825008	803873	73913	9
0.50	556459	650913	979577	728983	222100	30
0.20	147256	180605	34974	120945	76297	63
0.10	41284	52192		46738	7713	17
0.05	16715	10233		13474	4583	34

Table 44 Mustard Calibration

Conc (µg/mL)	1	2	3	4	5	Average	Standard Deviation	Relative sd (%)
1.00	4824294	4205734	3364057	2751485	2159964	3461107	1074558	31
0.75	3468863	2566022	2440129	1978491	1707818	2432265	675134	28
0.50	2161008	2036115	1665898	1376050	1057738	1659362	457125	28
0.20	922125	746116	689715	549593	282276	637965	239503	38
0.10	374371	226793	193378	204664	135370	226915	89084	39

Table 45 HN3 calibration

	Repeats							
Conc (µg/mL)	1	2	3	4	5	Average	Standard Deviation	Relative sd (%)
1.00	3655670	4601951	3365950	3269514	2545472	3487711	744982	21
0.75	2942589	2259657	2483290	2087050	1674043	2289326	470304	21
0.50	1820949	1508703	1723461	1577412	1297044	1585514	202262	13
0.20	705021	804769	708142	606866	348901	634740	174441	27
0.10	313650	254346	227411	209483	174739	235926	52192	22

Table 46 T Calibration

	Repeats							
Conc (µg/mL)	1	2	3	4	5	Average	Standard Deviation	Relative sd (%)
1.00	734829	613892	628859	520475	514197	602450	90638	15
0.75	581356	443022	440566	397761	320318	436605	94923	22
0.50	330290	261357	301820	249964	224245	273535	42292	15
0.20	151250	97777	122366	136686	77352	117086	29710	25

III. GCxGC-TOF Matrix Chromatograms

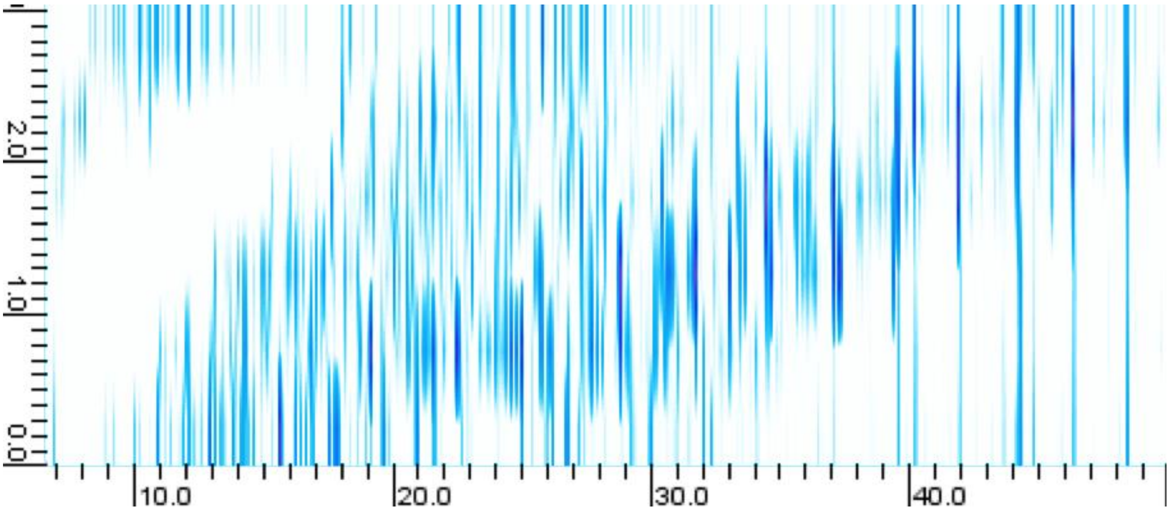


Figure 56 Diesel spiked with a CWA nerve and blister agent standard at 2 µg/mL

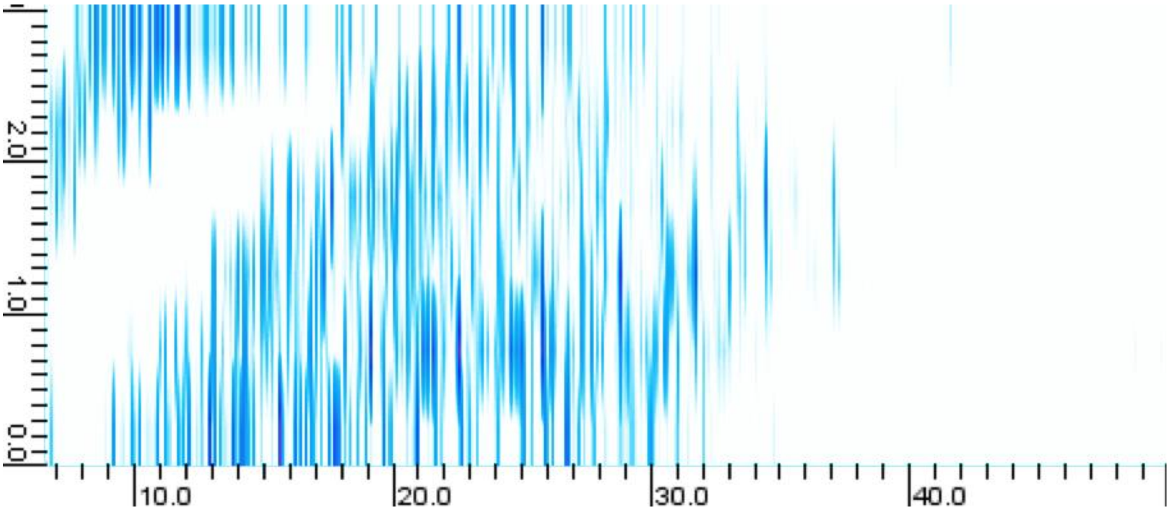


Figure 57 AV fuel spiked with a CWA nerve and blister agent standard at 2 µg/mL

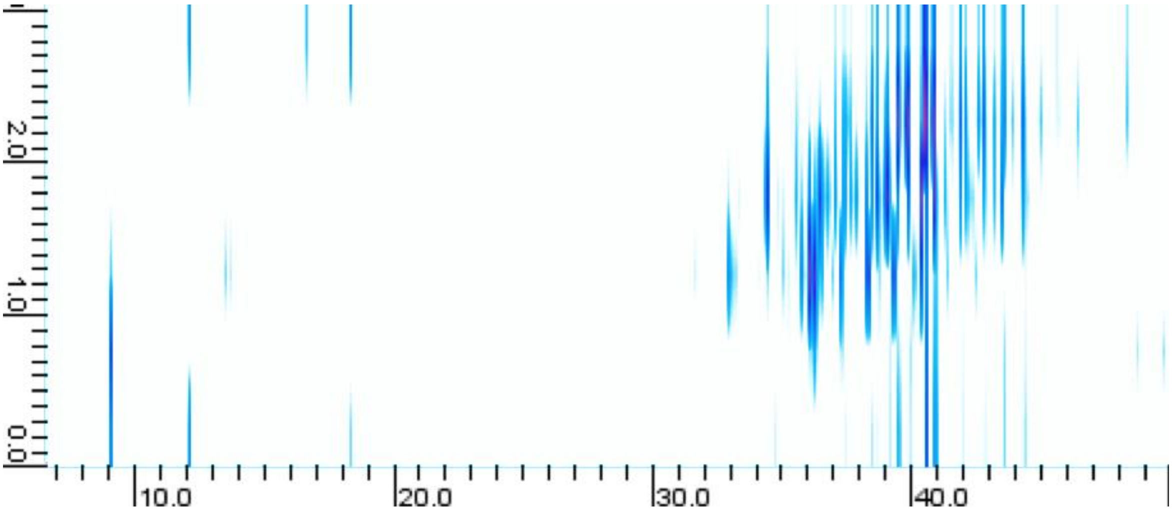


Figure 58 LWO spiked with a CWA nerve and blister agent standard at 2 µg/mL

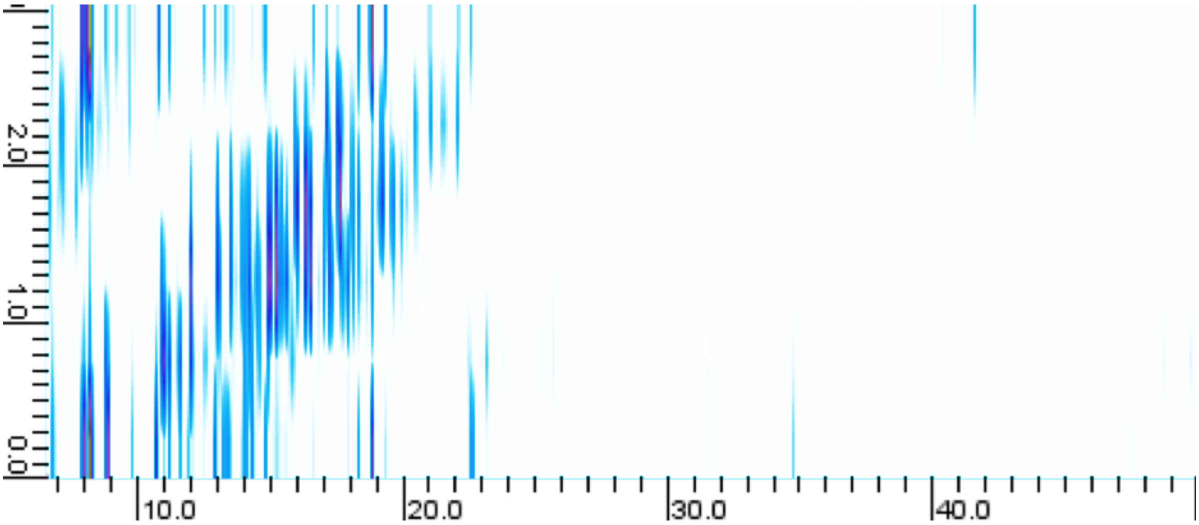


Figure 59 Gasoline Spiked with a CWA nerve and blister agent standard at 2 µg/mL

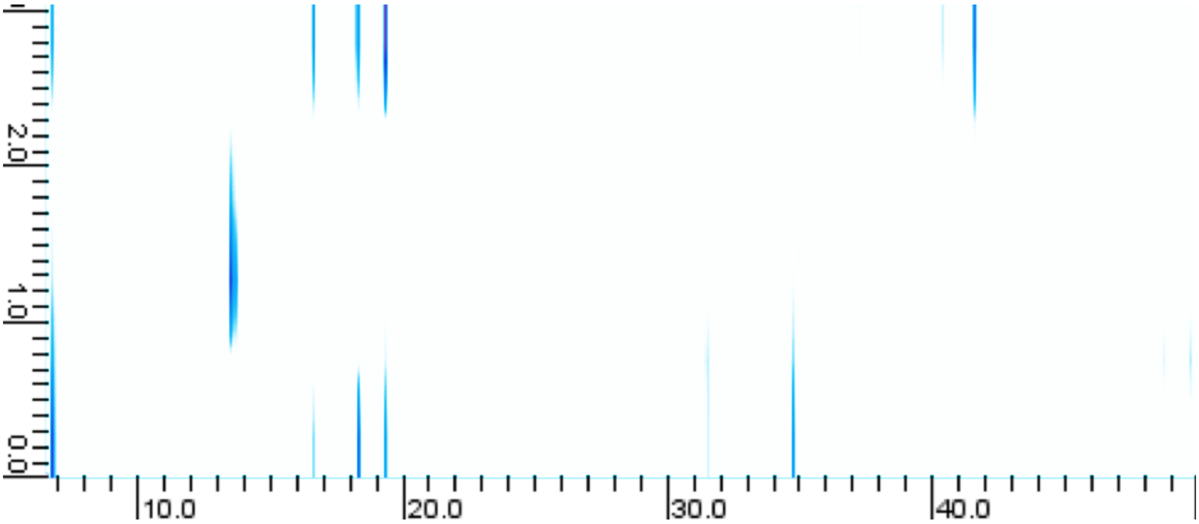


Figure 60 Swab of a hotel room spiked with a CWA nerve and blister agent standard at 2 µg/mL

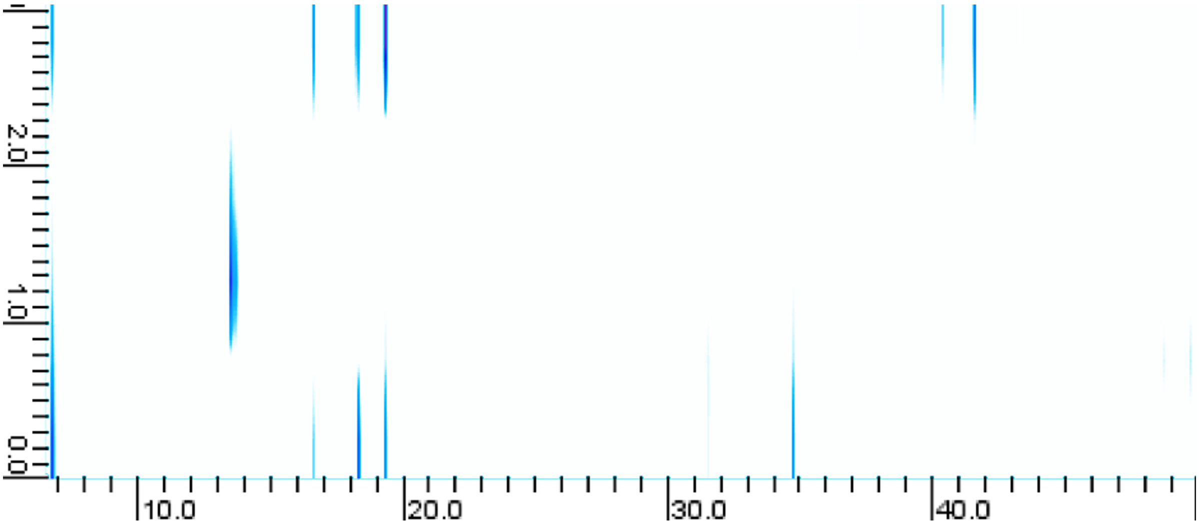


Figure 61 Swab of inside an oven spiked with CWA standard of nerve agents

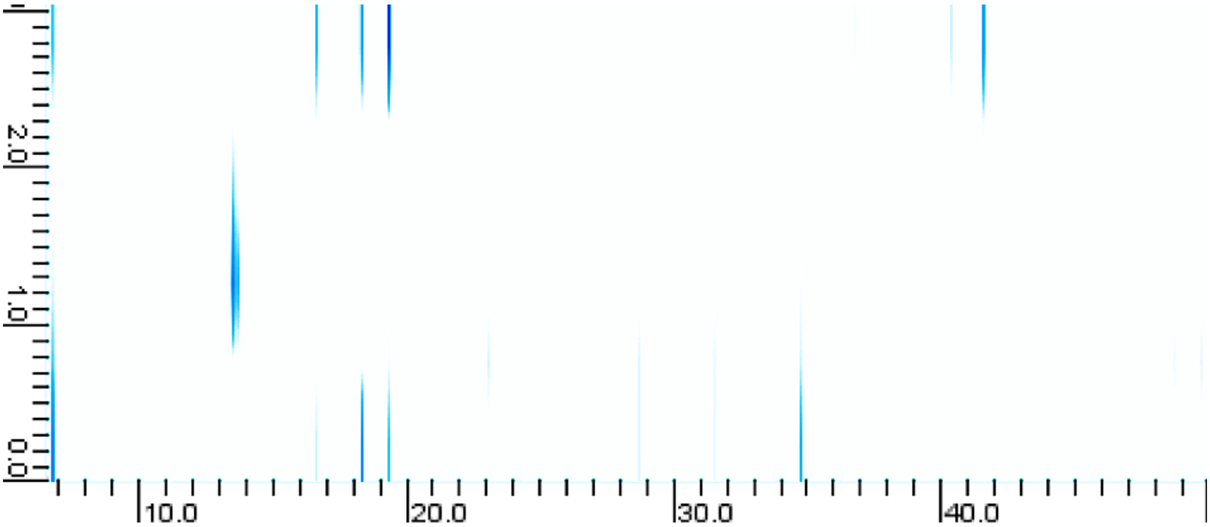


Figure 62 Swab of Clothing spiked with a CWA nerve and blister agent standard at 2 µg/mL

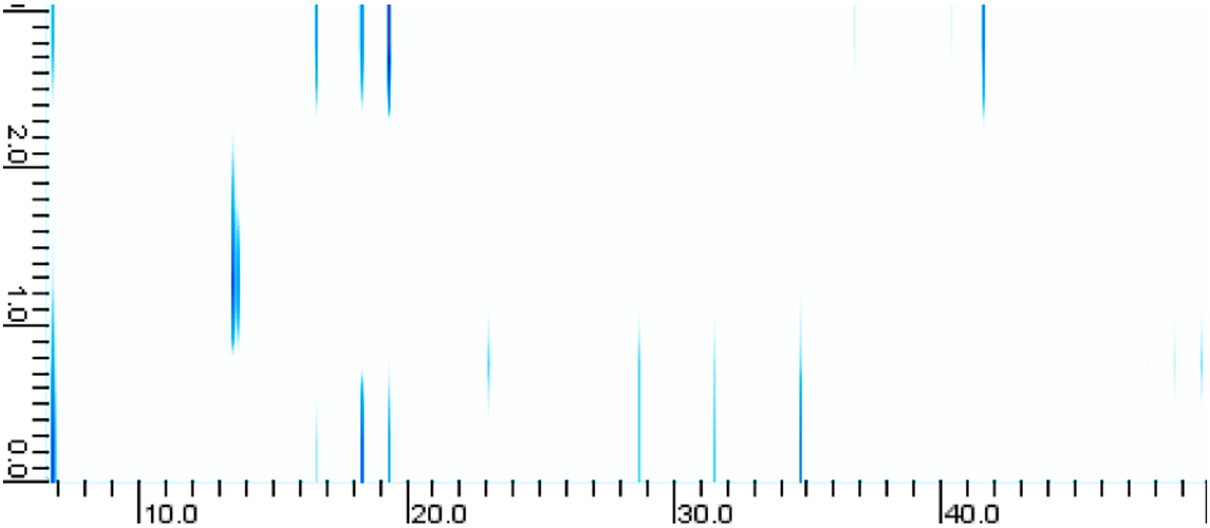


Figure 63 Aerosol sample from Porton Down spiked with a CWA nerve and blister agent standard at 2 µg/mL

IV. CWA GCxGC-FID Matrix analysis

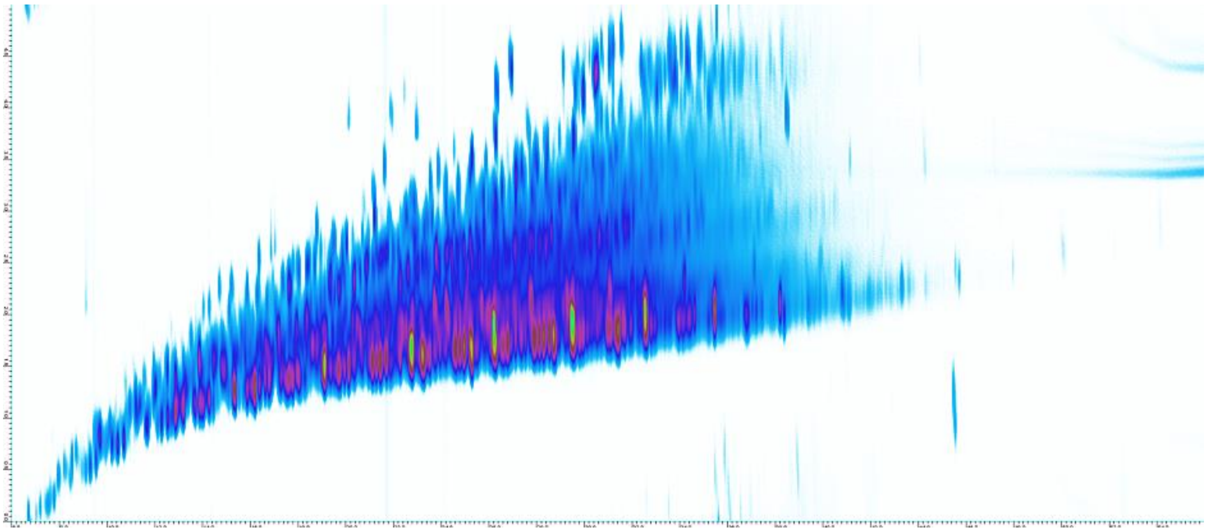


Figure 64 Aviation Fuel spiked with a CWA nerve and blister agent standard at 2 µg/mL

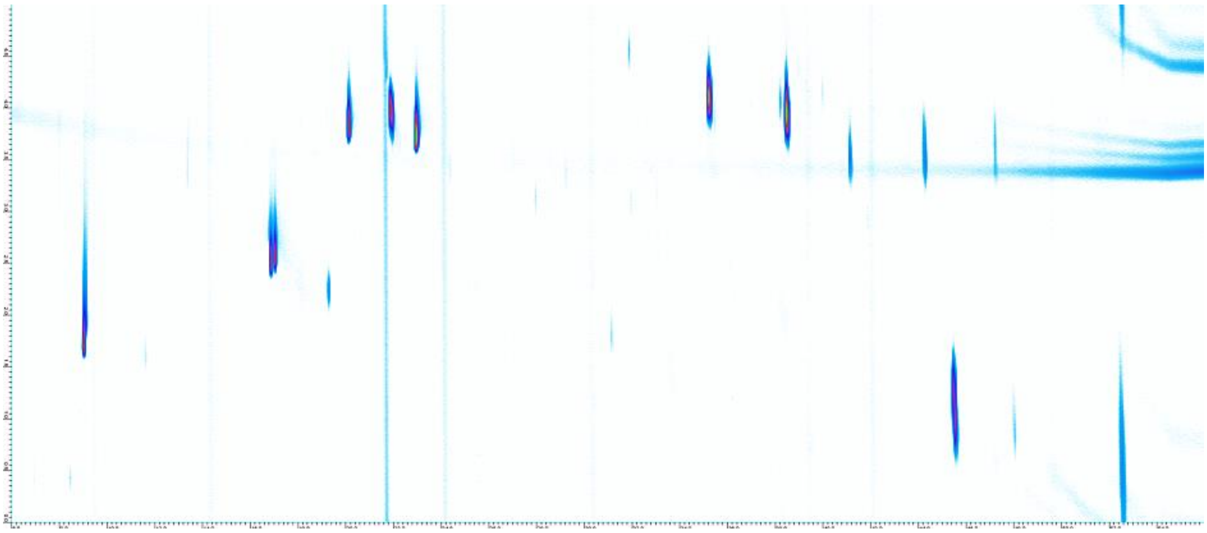


Figure 65 A swab of clothing spiked with a CWA nerve and blister agent standard at 2 µg/mL

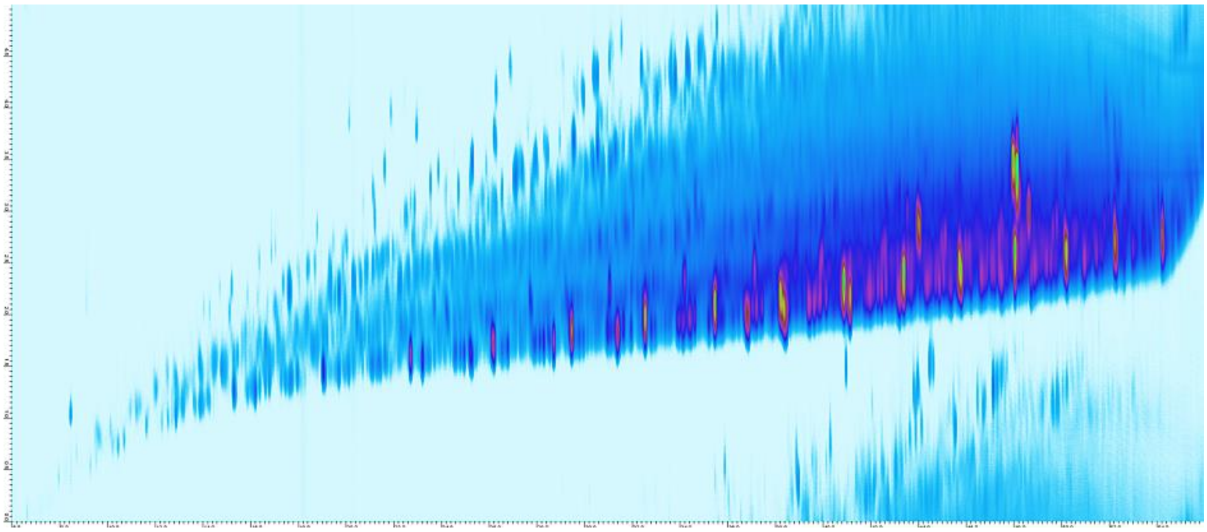


Figure 66 Diesel spiked with a CWA nerve and blister agent standard at 2 µg/mL

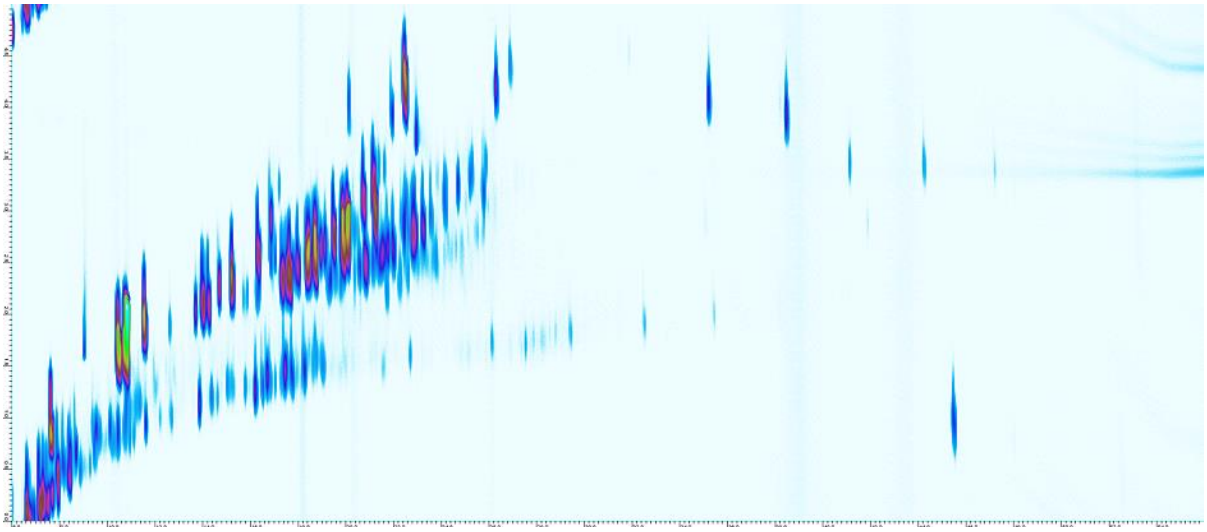


Figure 67 Gasoline spiked with a CWA nerve and blister agent standard at 2 µg/mL

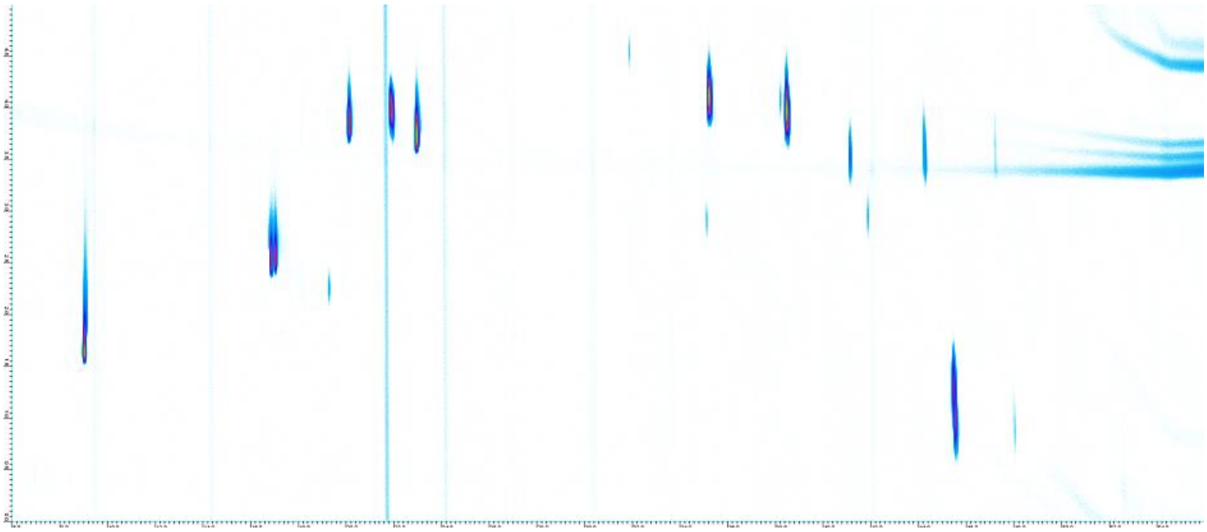


Figure 68 A swab of a hotel room spiked with a CWA nerve and blister agent standard at 2 µg/mL

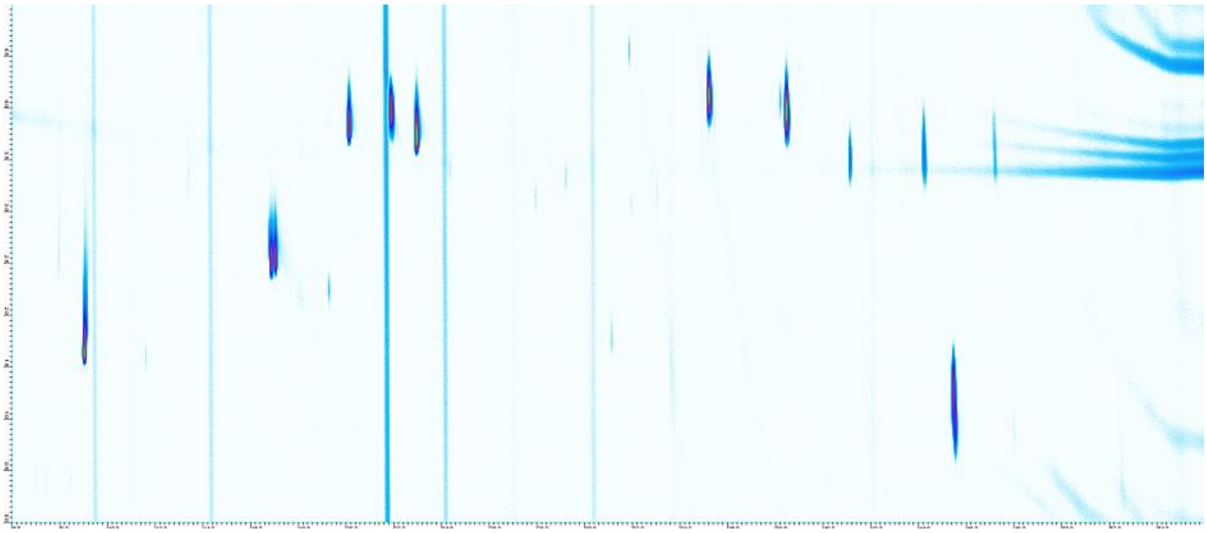


Figure 69 A swab of the inside of an oven spiked with a CWA nerve and blister agent standard at 2 µg/mL

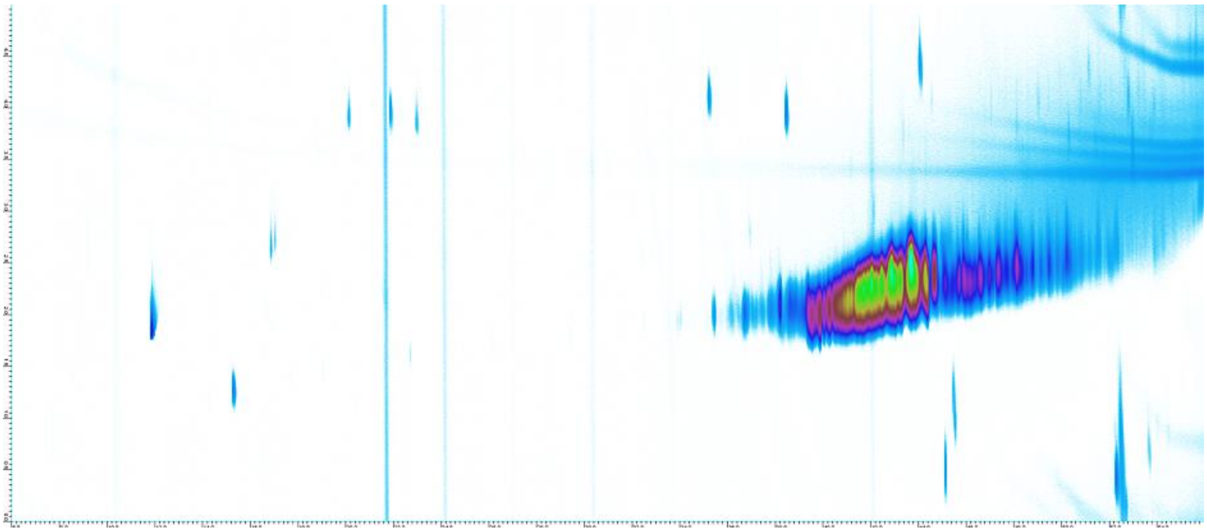


Figure 70 Light weight oil used for weapons spiked with a CWA nerve and blister agent standard at 2 µg/mL

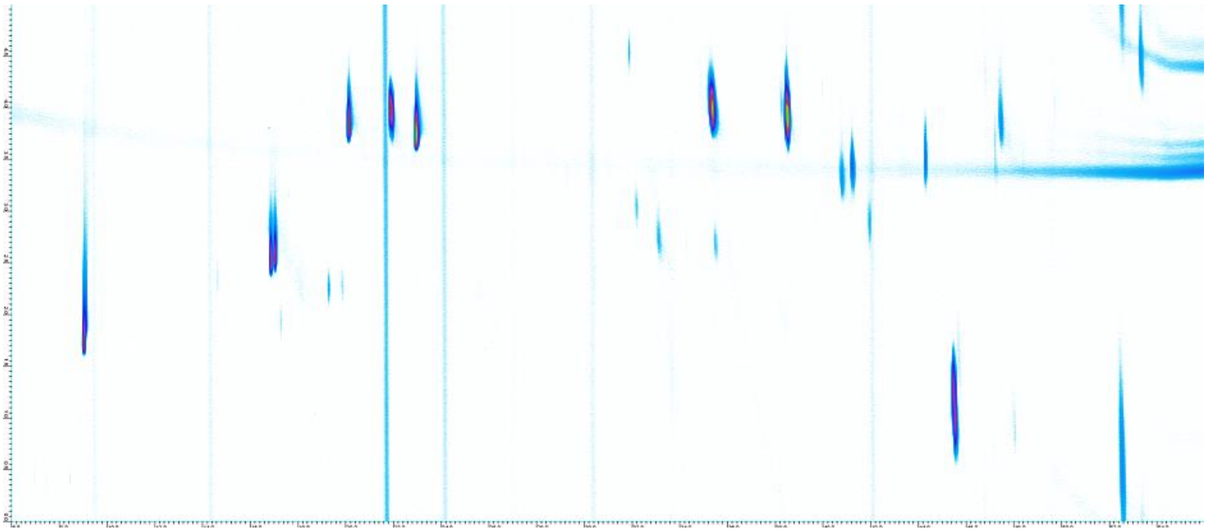


Figure 71 Aerosol sample from PTN range spiked with a CWA nerve and blister agent standard at 2 µg/mL

Annex 2

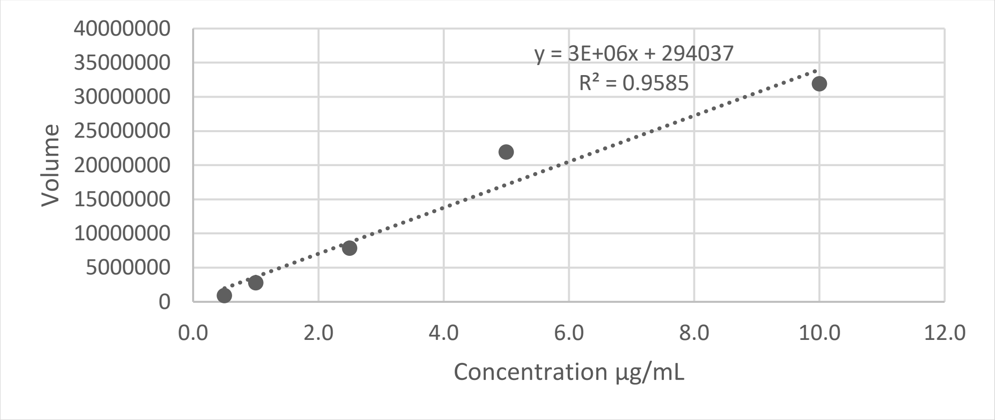


Figure 72 GB calibration

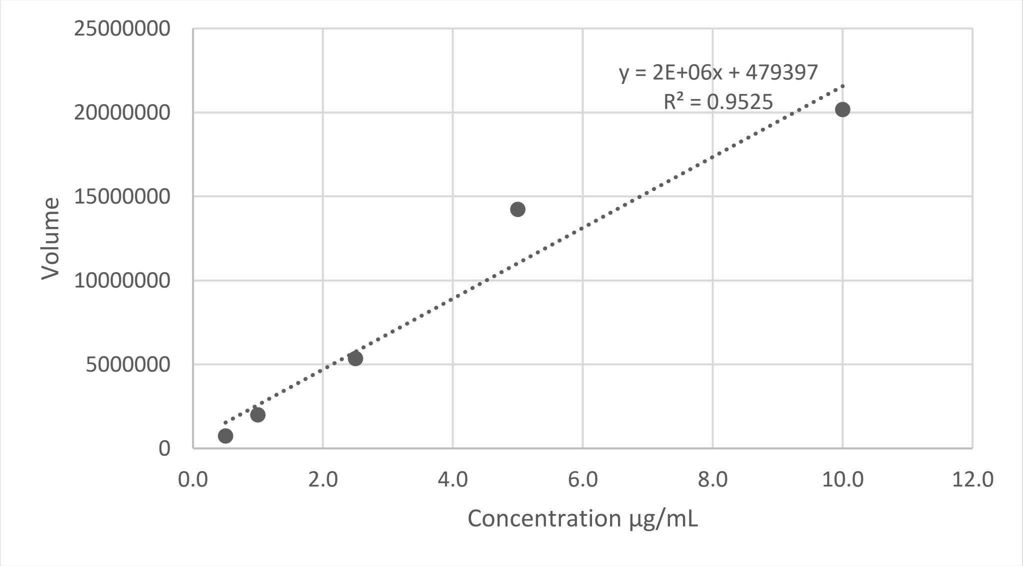


Figure 73 GD peak 1 calibration

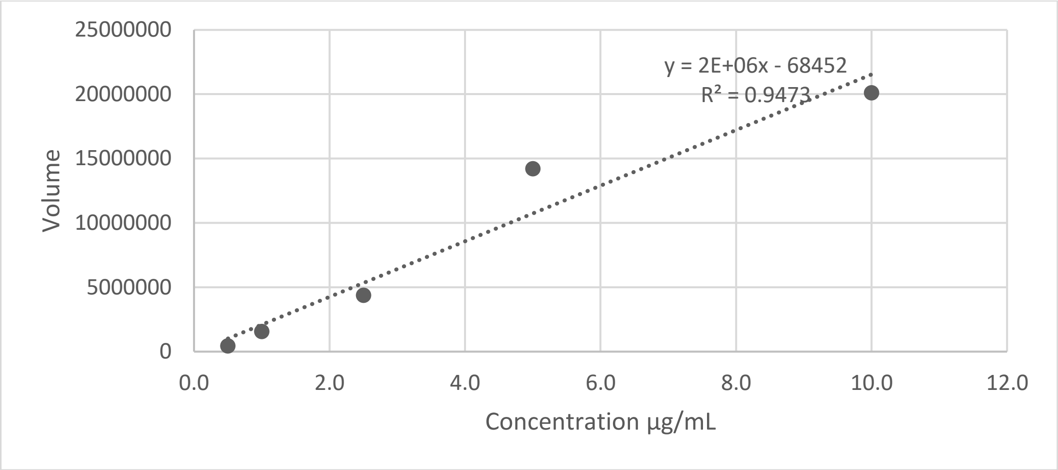


Figure 74 GD peak 2 calibration

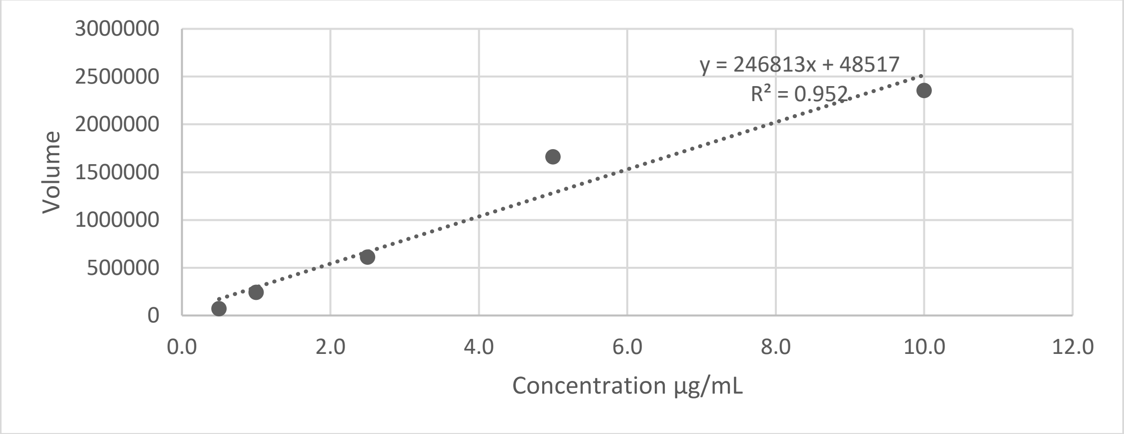


Figure 75 GA calibration

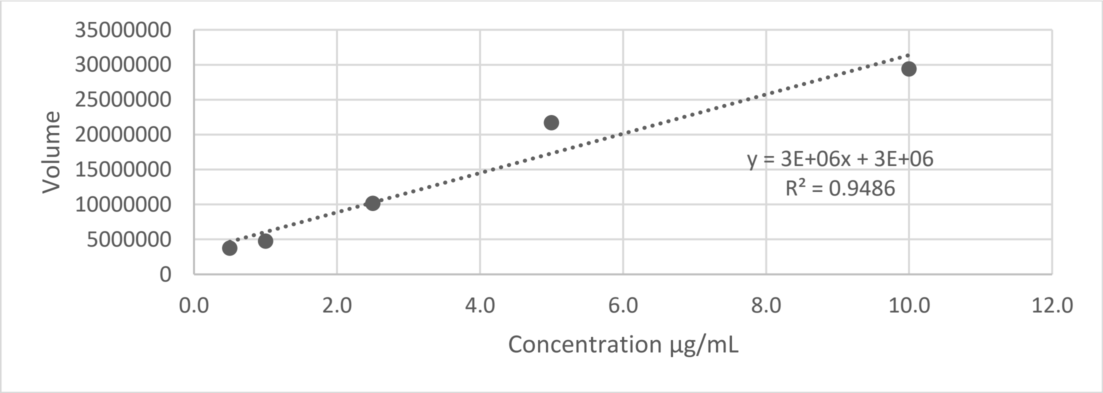


Figure 76 H calibration

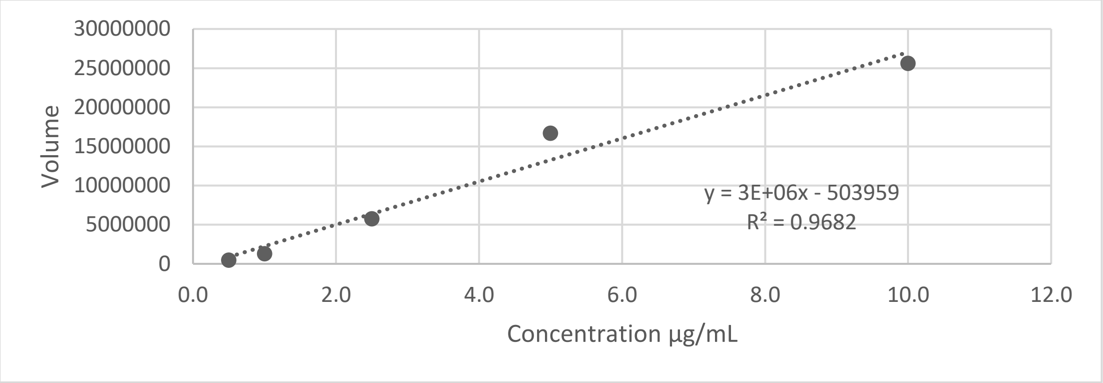


Figure 77 GF calibration

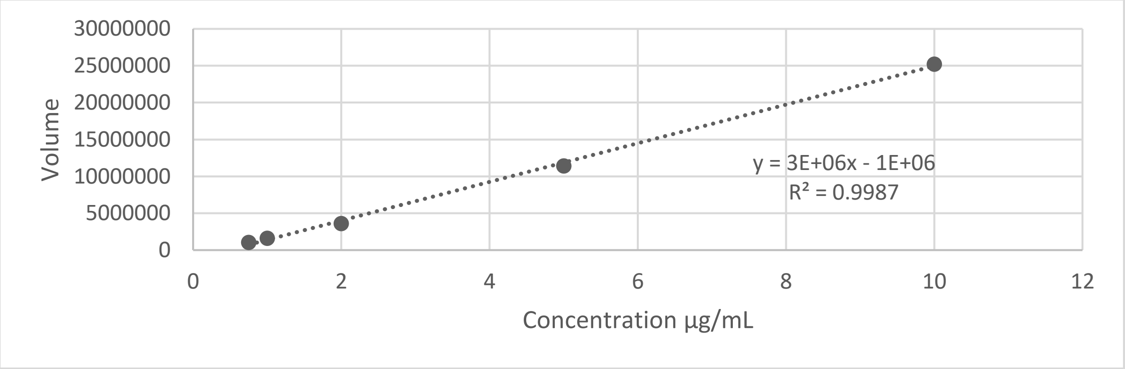


Figure 78 VM calibration

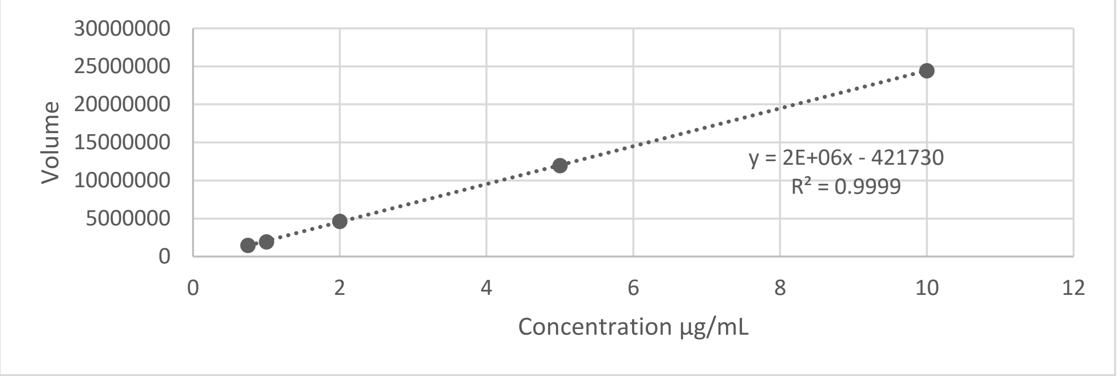


Figure 79 VX calibration

I. Explosive Calibration Data

Table 47 2-NT Calibration data

Concentration	1	2	3	mean	SD	RSD
50.00	1.03E+08	1.04E+08	1.57E+08	121181344	31346024	26
30.00	91266887	96576412	87298484	91713928	4655091	5
25.00	63076865	57921060	76590995	65862973	9641754	15
20.00	57072172	57715414	53343041	56043542	2360714	4
15.00	47751118	42864122	44053271	44889504	2548558	6
10.00	31479194	27027044	26487818	28331352	2739411	10
5.00	14059899	14805800	15668944	14844881	805234	5
2.50	6976879	5287640	6501059	6255193	871045	14
1.00	2866566	2445786	2891146	2734499	250335	9
0.75	3000760	2682479	2713664	2798968	175452	6
0.50	2789400	2830427	2135101	2584976	390143	15

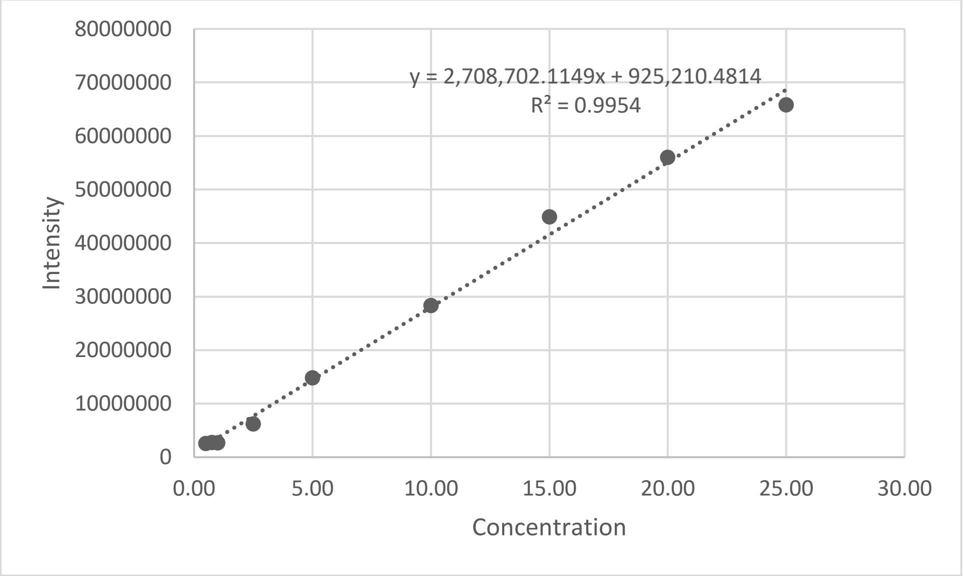


Figure 80 2-NT calibration

Table 48 3-NT calibration

Concentration	1	2	3	mean	SD	RSD
50.00	1.28E+08	1.27E+08	1.9E+08	148447012	35822248	24
30.00	1.08E+08	1.13E+08	1.04E+08	108296174	4669772	4
25.00	76087842	71314698	90821711	79408084	10168525	13
20.00	70521840	68618463	63936972	67692425	3388699	5
15.00	56292216	52295980	53045781	53877992	2124125	4
10.00	36882208	32703206	32746540	34110651	2400336	7
5.00	16647637	18159100	18069869	17625535	848059	5
2.50	8140225	6417211	7739687	7432374	901679	12
1.00	3091903	2981573	3266141	3113206	143475	5
0.75	3246847	2953673	3177415	3125978	153206	5
0.50	3499209	3214444	2479604	3064419	526098	17

Figure 81 3-NT calibration

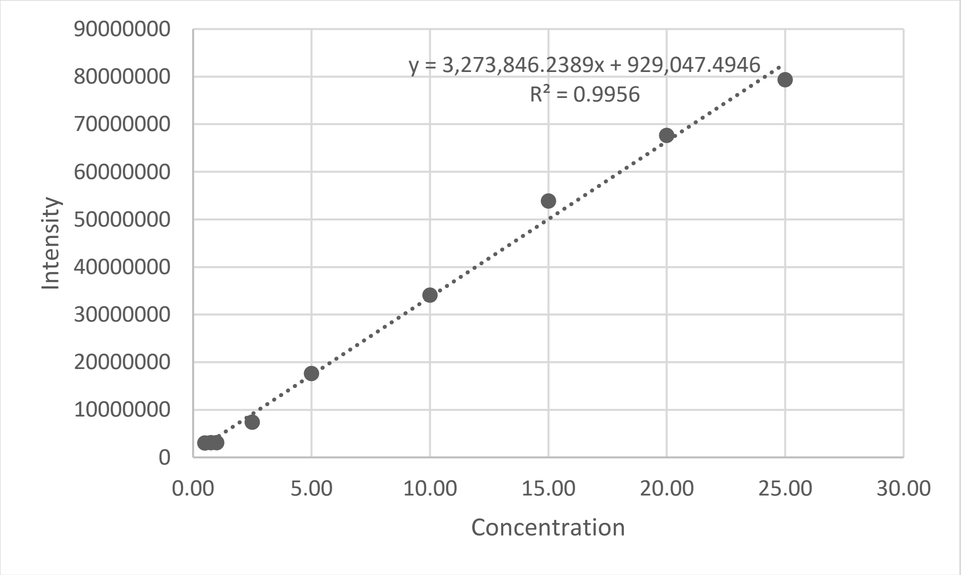


Table 49 4-NT Calibration

Concentration	1	2	3	mean	SD	RSD
50.00	1.35E+08	1.32E+08	1.91E+08	152539289	33072175	22
30.00	1.08E+08	1.13E+08	1.05E+08	108629982	4024364	4
25.00	77509017	72604788	91213840	80442548	9645124	12
20.00	68341044	70206076	64207840	67584987	3069760	5
15.00	56132970	52120158	54185229	54146119	2006692	4
10.00	36481827	32639487	32189388	33770234	2359068	7
5.00	16600795	17477895	17876465	17318385	652622	4
2.50	8121516	6478202	7341934	7313884	822016	11
1.00	3170637	2903737	3222103	3098826	170900	6
0.75	3163157	2982738	3255024	3133640	138522	4
0.50	3606659	3416528	2832038	3285075	403695	12

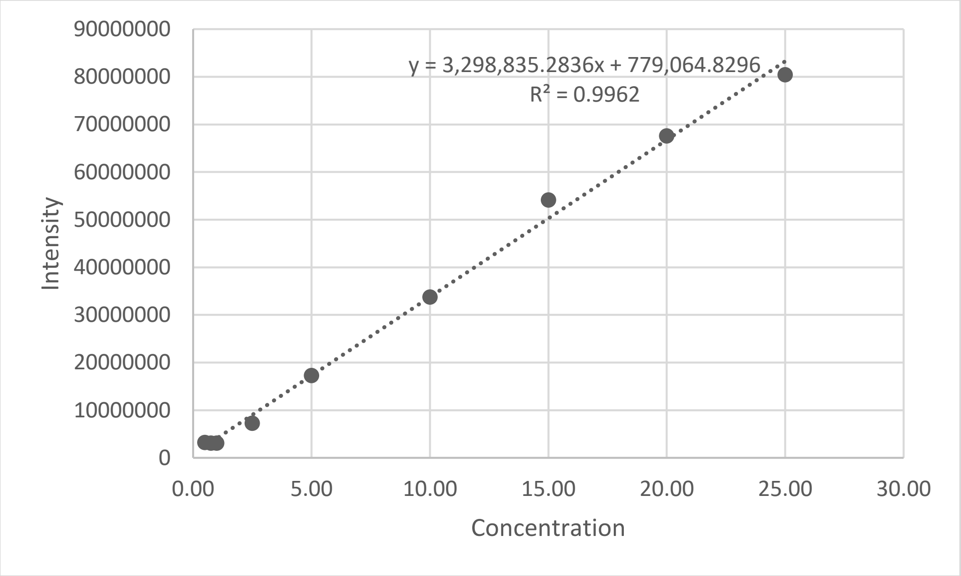


Figure 82 4-NT Calibration

Table 50 1,3-DNB Calibration

Concentration	1	2	3	mean	SD	RSD
50.00	1.4E+08	1.36E+08	1.96E+08	157348433	33112165	21
30.00	1.11E+08	1.17E+08	1.08E+08	112095917	4141725	4
25.00	80511245	75059051	93765436	83111911	9620541	12
20.00	70927152	71067364	66714310	69569609	2473755	4
15.00	56888795	53650213	54564412	55034473	1669677	3
10.00	37037384	33020586	33433707	34497226	2209518	6
5.00	16713184	17728060	18133086	17524777	731453	4
2.50	8053962	6712326	7521875	7429388	675583	9
1.00	3193346	2957330	3270163	3140280	163028	5
0.75	3324458	3020099	3215229	3186595	154187	5
0.50	3448904	3444075	2847321	3246767	345939	11

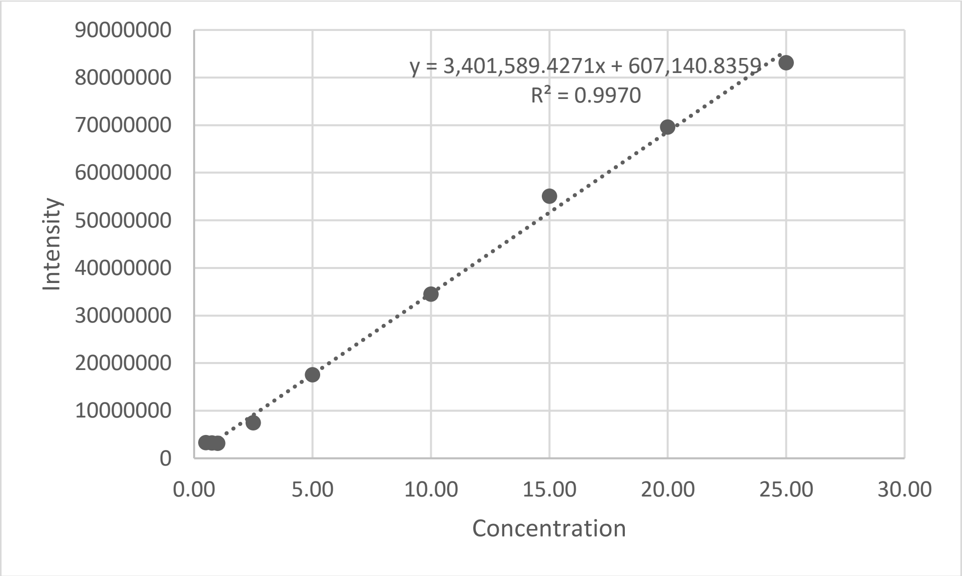


Figure 83 1,3-DNB calibration

Table 51 2,6-DNT Calibration

Concentration	1	2	3	mean	SD	RSD
50.00	1.13E+08	1.16E+08	1.34E+08	120872191	11106111	9
30.00	78386800	77359910	73673665	76473458	2478459	3
25.00	64421815	60328121	63124523	62624820	2092095	3
20.00	52281054	49042531	51889645	51071077	1767639	3
15.00	40998011	41898001	40690857	41195623	627365	2
10.00	30144257	27636221	26647789	28142756	1802430	6
5.00	15657777	16698665	15532955	15963132	640040	4
2.50	8274242	10075839	10258876	9536319	1096816	12

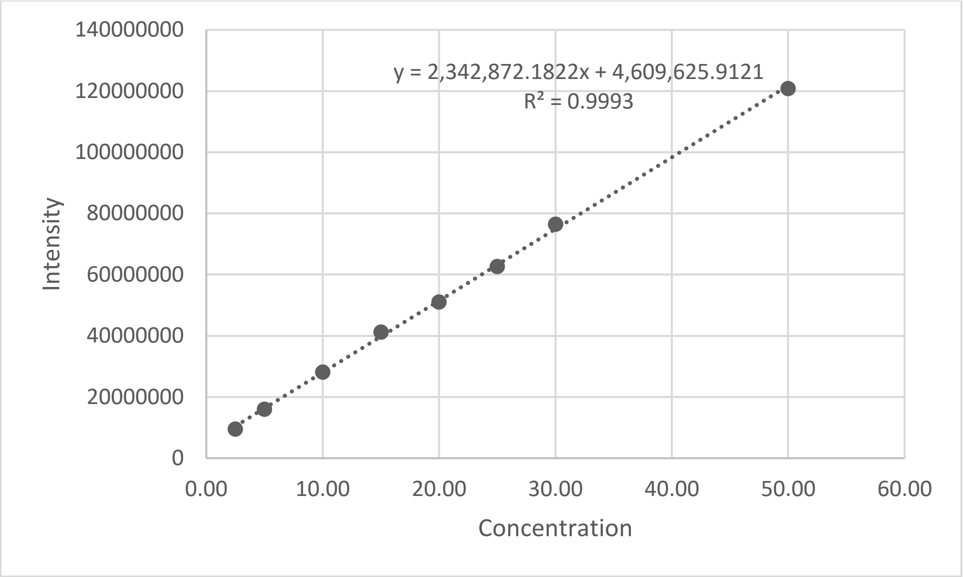


Figure 84 2,6-DNT Calibration

Table 52 2,4-DNT Calibration

Concentration	1	2	3	mean	SD	RSD
50.00	1.35E+08	1.33E+08	1.58E+08	141899763	13817363	10
30.00	89827458	91065846	87911451	89601585	1589282	2
25.00	70901954	67777844	74041145	70906981	3131654	4
20.00	57248717	57582425	56538895	57123346	532942	1
15.00	45186069	44707974	44529981	44808008	339290	1
10.00	29449420	27927448	26827145	28068004	1316776	5
5.00	13347633	14849982	14550411	14249342	795138	6
2.50	6652462	5883942	6995517	6510640	569197	9

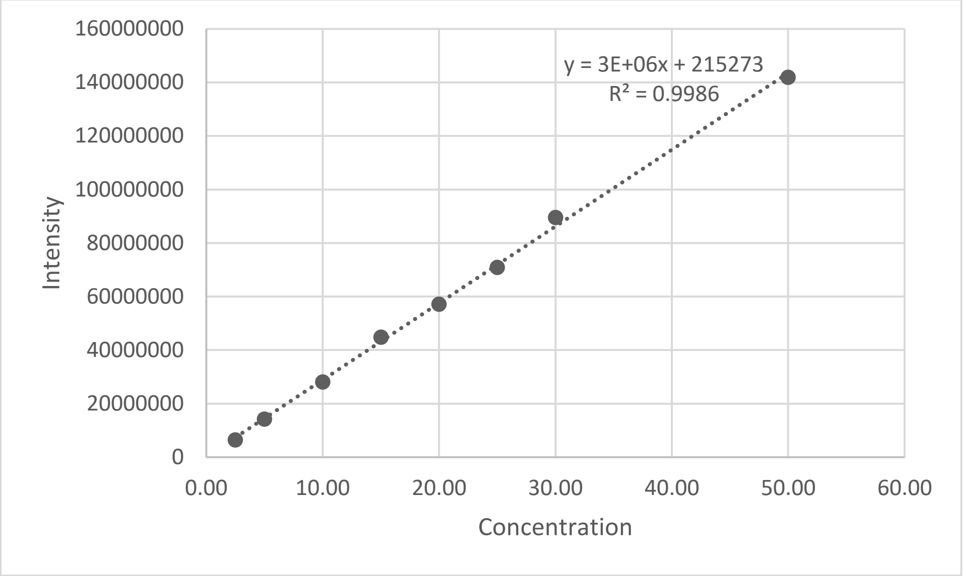


Figure 85 2,4 – DNT Calibration

Table 53 1,3,5-TNB

Concentration	1	2	3	mean	SD	RSD
50.00	1.26E+08	1.28E+08	1.45E+08	133124149	10624844	8
30.00	82436376	83868780	78965301	81756819	2521384	3
25.00	67352196	66651796	66760065	66921352	377028	1
20.00	52663102	53044519	53697108	53134910	522896	1
15.00	40279103	40547457	41758773	40861778	788323	2
10.00	27968606	25676212	25460627	26368482	1389934	5
5.00	13534985	13481226	13731772	13582661	131902	1
2.50	7376013	5771784	6375132	6507643	810282	12

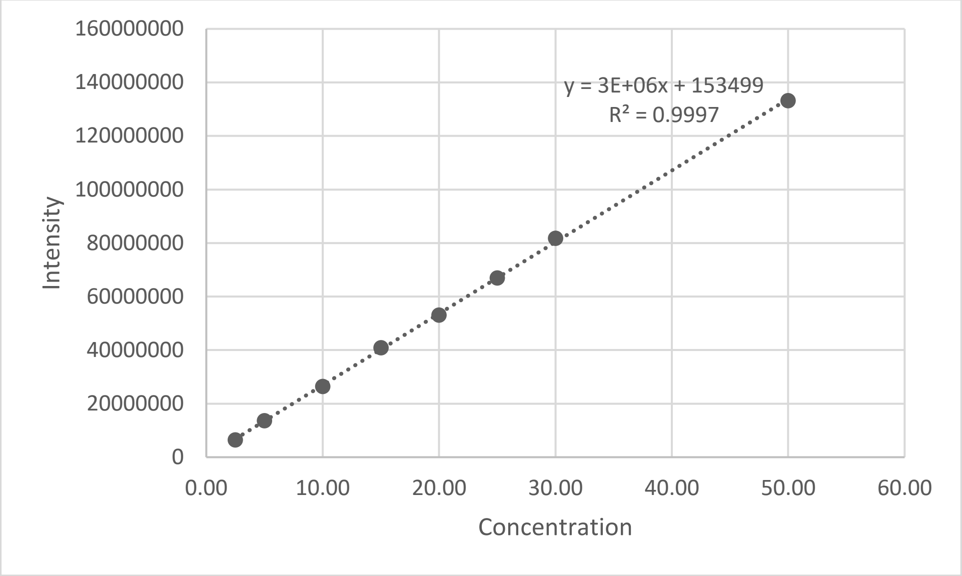


Figure 86 1,3,5-TNB

Table 54 2,4,6-TNT

Concentration	1	2	3	mean	SD	RSD
50	77839214	83548059	89116886	83501386	5638981	7
30	51237548	49817452	47924662	49659887	1662054	3
25	44086774	45032660	39501826	42873753	2958225	7
20	36045886	32661581	34328834	34345434	1692214	5
15	26734790	28139133	28209773	27694565	831940	3
10	20682781	20388762	20747875	20606473	191332	1
5	11695179	12183434		11939307	345248	3

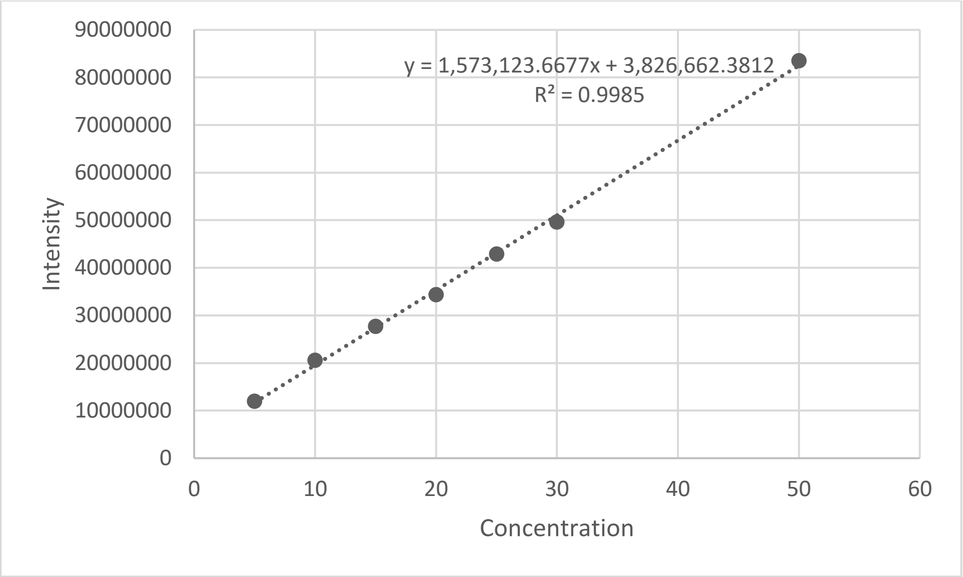


Figure 87 2,4,6-TNT Calibration

II. Matlab Code R2015a

```
function varargout = duitest8(varargin)
% DUITEST8 MATLAB code for duitest8.fig
%      DUITEST8, by itself, creates a new DUITEST8 or raises the existing
%      singleton*.
%
%      H = DUITEST8 returns the handle to a new DUITEST8 or the handle to
%      the existing singleton*.
%
%      DUITEST8('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in DUITEST8.M with the given input
arguments.
%
%      DUITEST8('Property','Value',...) creates a new DUITEST8 or raises
the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before duitest8_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to duitest8_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help duitest8

% Last Modified by GUIDE v2.5 20-Jul-2018 14:08:27

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @duitest8_OpeningFcn, ...
                   'gui_OutputFcn',  @duitest8_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

format compact
dbstop if error
% End initialization code - DO NOT EDIT



% --- Executes just before duitest8 is made visible.
function duitest8_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to duitest8 (see VARARGIN)

% Choose default command line output for duitest8
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes duitest8 wait for user response (see UIRESUME)
% uiwait(handles.figure1);



% --- Outputs from this function are returned to the command line.
function varargout = duitest8_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



% --- Executes on selection change in listbox1.
function listbox1_Callback(hObject, eventdata, handles)
% hObject    handle to listbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns listbox1
contents as cell array
%        contents{get(hObject,'Value')} returns selected item from listbox1



% --- Executes during object creation, after setting all properties.
function listbox1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end




function [p] = FindCentroidLimit(c,x1,x2,y1,y2)
% Find where the point is between two limits in x and y.
xi = intersect(find(c(:,1) < x2),find(c(:,1) > x1));
yi = intersect(find(c(:,2) < y2),find(c(:,2) > y1));
p = intersect(xi,yi);

% --- Executes on button press in pushbutton1.
```

```
function pushbutton1_Callback(hObject, eventdata, handles)
% Imports the image.

% Prompts user for image using file extensions.
[filename,filepath] = uigetfile({'*.*';'*.jpg';'*.png';'*.bmp'}, 'Search
Image To Be Displayed');
fullname=[filepath filename];

% Read the image.
ImageFile = imread(fullname);

%threshold = 0.5;

%graypic = rgb2gray(ImageFile);
%bwpic = im2bw(graypic,threshold);
%bwpic2 = imcomplement(bwpic);

ud = struct([]);
ud(1).ImageFile = ImageFile;
set(handles.figure1,'userdata',ud);

% Display the image.
axes(handles.axes2)
imagesc(ImageFile);

% clear axes scale
axis off



% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)



% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)

% This is the image 'detect' button on the GUI.

ud = get(handles.figure1,'userdata');
ImageFile = ud(1).ImageFile;
clear ud
threshold=0.5
picture = (ImageFile);
graypic = rgb2gray(picture);

bwpic = im2bw(graypic,threshold);
bwpic2 = imcomplement(bwpic);

props = regionprops(bwpic2, 'centroid');
centroids = cat(1,props.Centroid);



xf = [120 125; 120 125;310 315;295 310;360 365;422 427;660 663;720 725];
yf = [1 9; 400 450;35 45;340 350;200 244;207 270;340 350;1 40];
nf = char('G agent','G agent','G agent','G agent','Blister','G agent','V
agent','V agent');
```

```
for i=1:size(xf,1)
    [p] = FindCentroidLimit(centroids,xf(i,1),xf(i,2),yf(i,1),yf(i,2));
    if isempty(p)
        thistext= ['Not Found '  strtrim(nf(i,:))];
    else
        thistext =['Found '  strtrim(nf(i,:))];
    end
    if i==1;
        mtext = thistext;
    else
        mtext = char(mtext,thistext);
    end
end
set(findobj('tag','listbox1'),'string',mtext,'value',1)



% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)



% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% This is the image 'detect' button on the GUI.

ud = get(handles.figure1,'userdata');
ImageFile = ud(1).ImageFile;
clear ud
threshold=0.5
picture = (ImageFile);
graypic = rgb2gray(picture);

bwpic = im2bw(graypic,threshold);
bwpic2 = imcomplement(bwpic);

props = regionprops(bwpic2, 'centroid');
centroids = cat(1,props.Centroid);



xf = [120 125; 120 125;310 315;295 310;360 365;422 427;660 663;720 725];
yf = [1 9; 400 450;35 45;340 350;200 244;207 270;340 350;1 40];
nf = char('GB','GB','GA','GD','H','GF','VM','VX');

for i=1:size(xf,1)
    [p] = FindCentroidLimit(centroids,xf(i,1),xf(i,2),yf(i,1),yf(i,2));
    if isempty(p)
        thistext= ['Not Found '  strtrim(nf(i,:))];
    else
        thistext =['Found '  strtrim(nf(i,:))];
    end
    if i==1;
        mtext = thistext;
    else
        mtext = char(mtext,thistext);
    end
end
set(findobj('tag','listbox1'),'string',mtext,'value',1)

% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

III. CWA Matrices Chromatograms

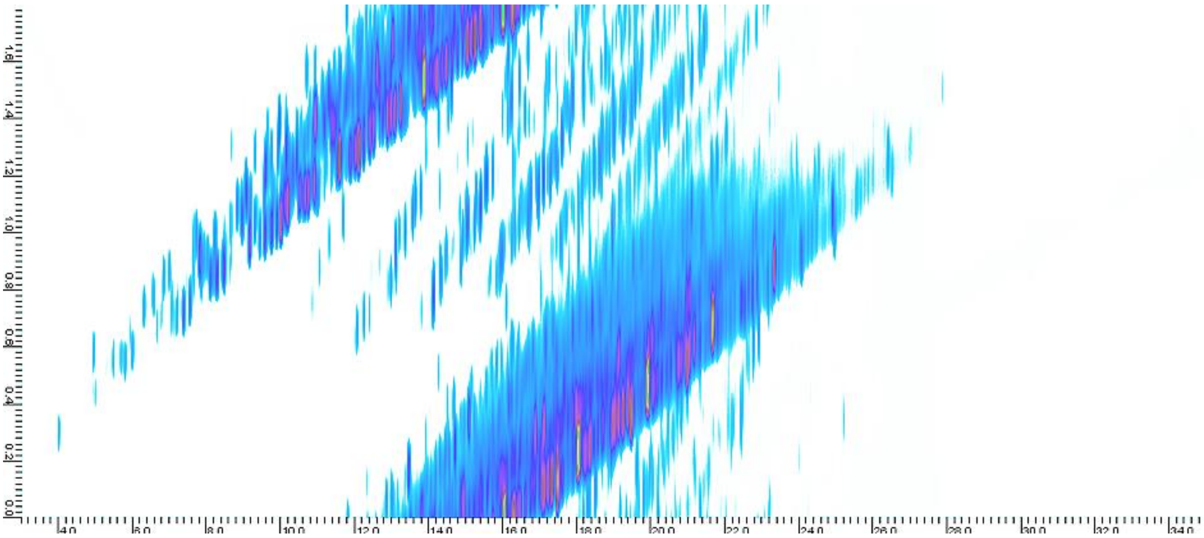


Figure 88 Aviation fuel spiked with CWA

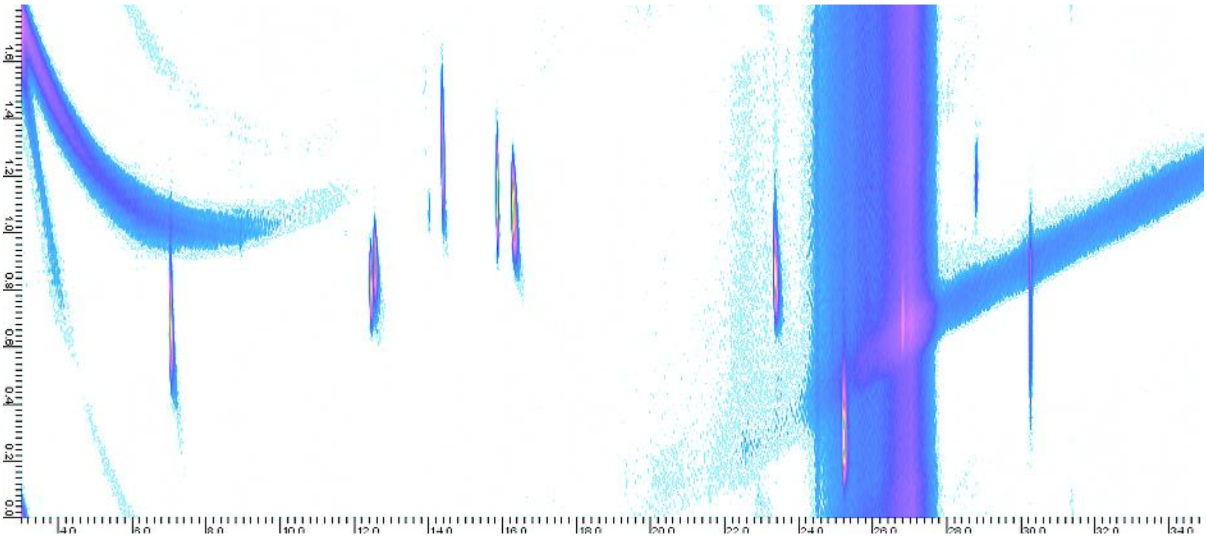


Figure 89 Swab of clothing spiked with CWA

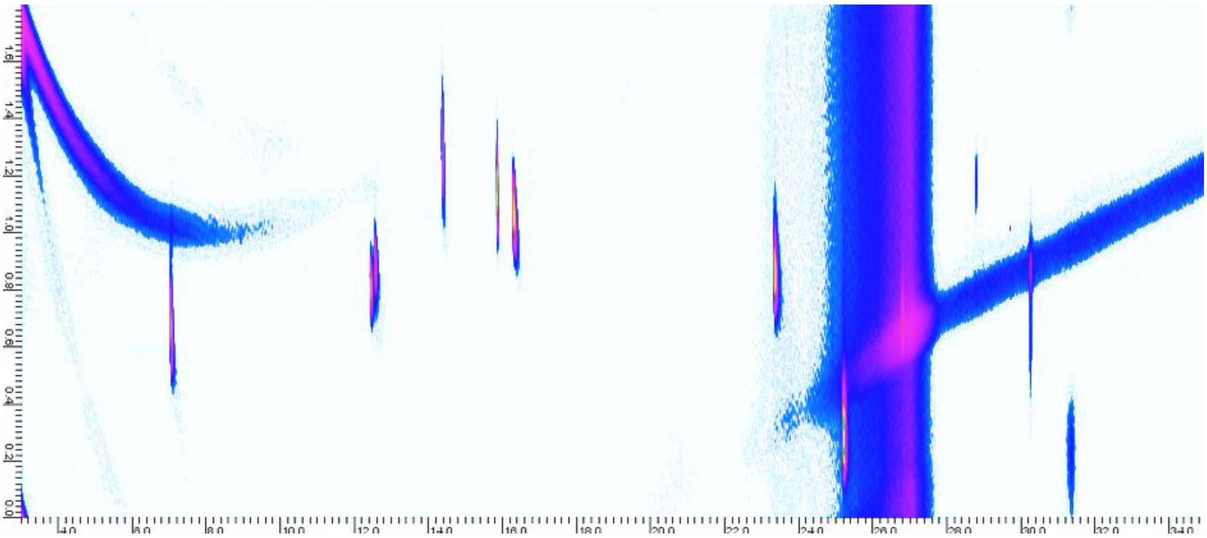


Figure 90 Control swab spiked with CWA

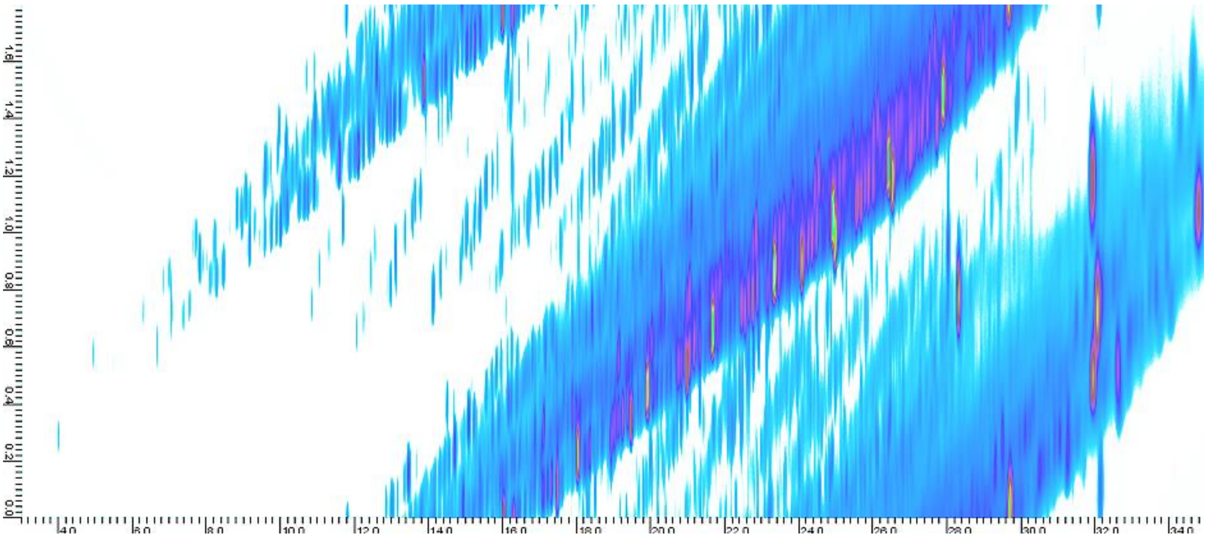


Figure 91 Diesel spiked with CWA

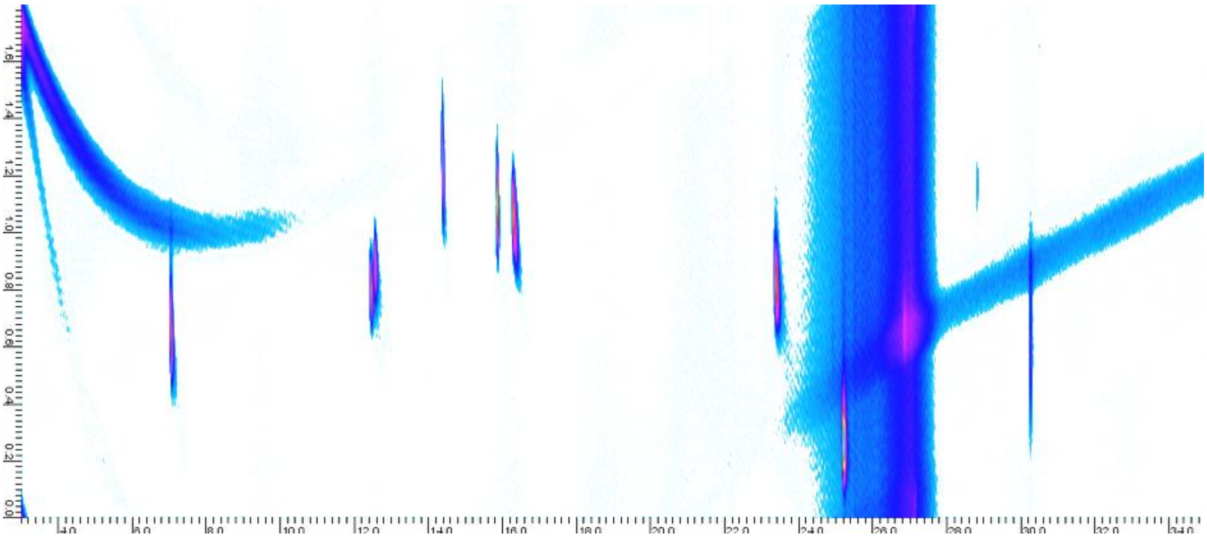


Figure 92 Sahara dust sample spiked with explosives

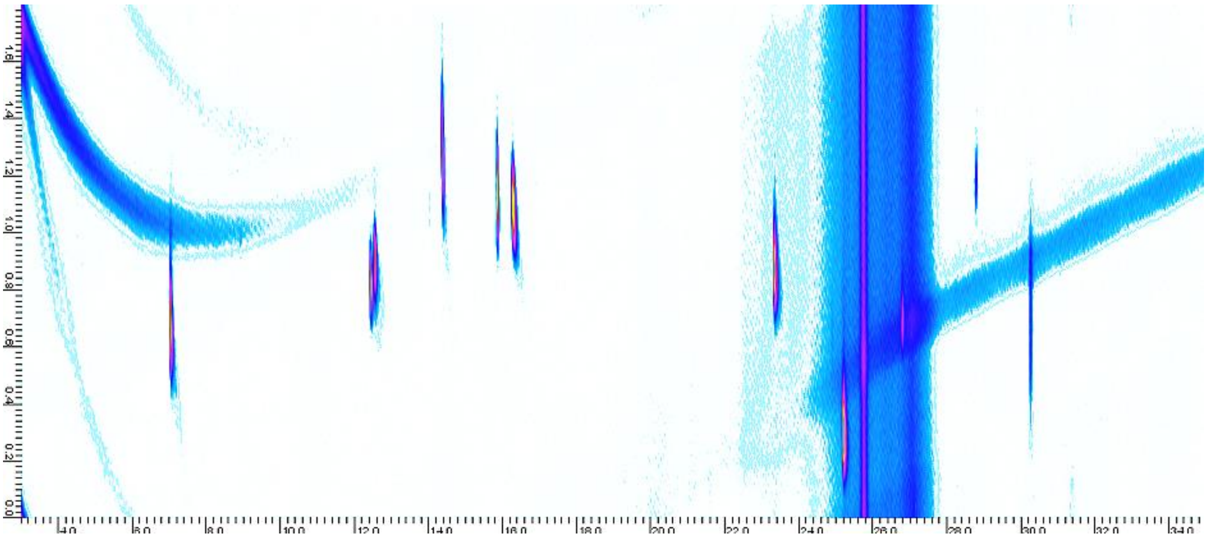


Figure 93 Swab of a kitchen floor spiked with CWA

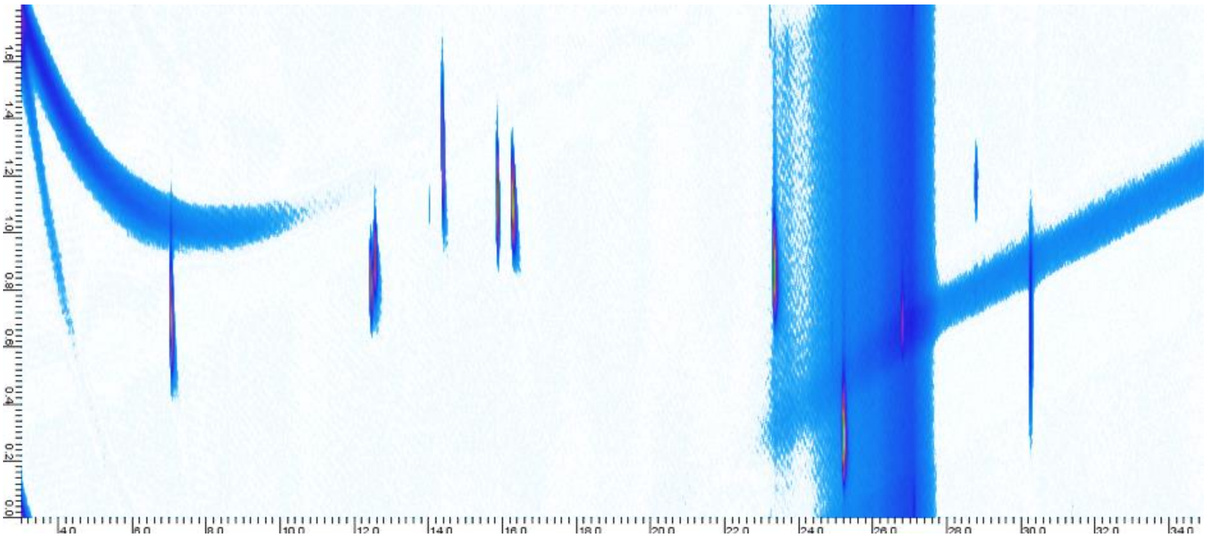


Figure 94 Hotel swab 1 spiked with CWA

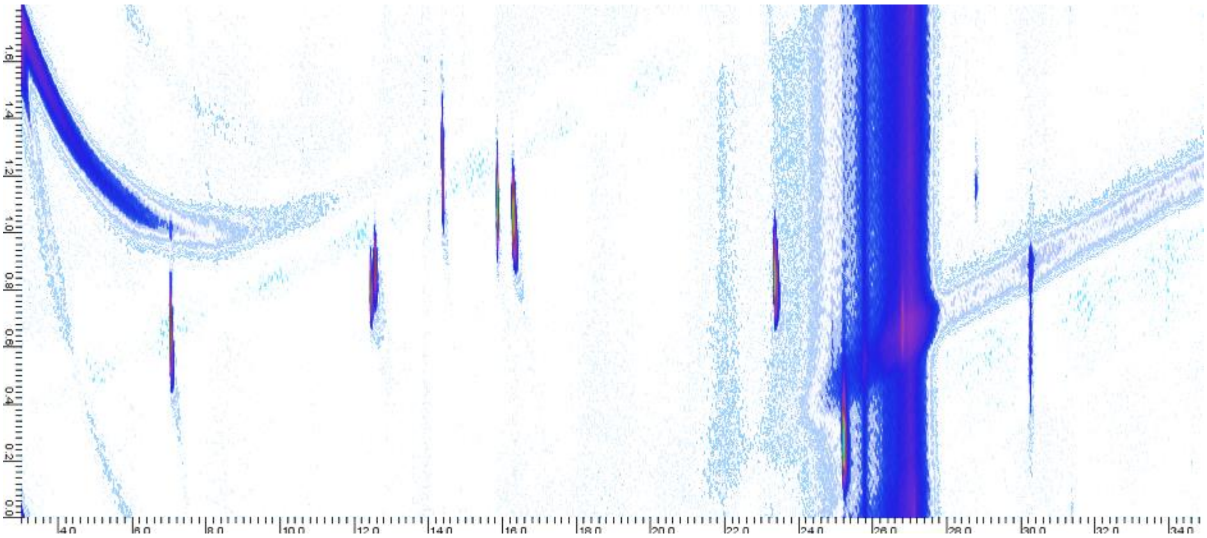


Figure 95 Hotel swab 2 spiked with CWA

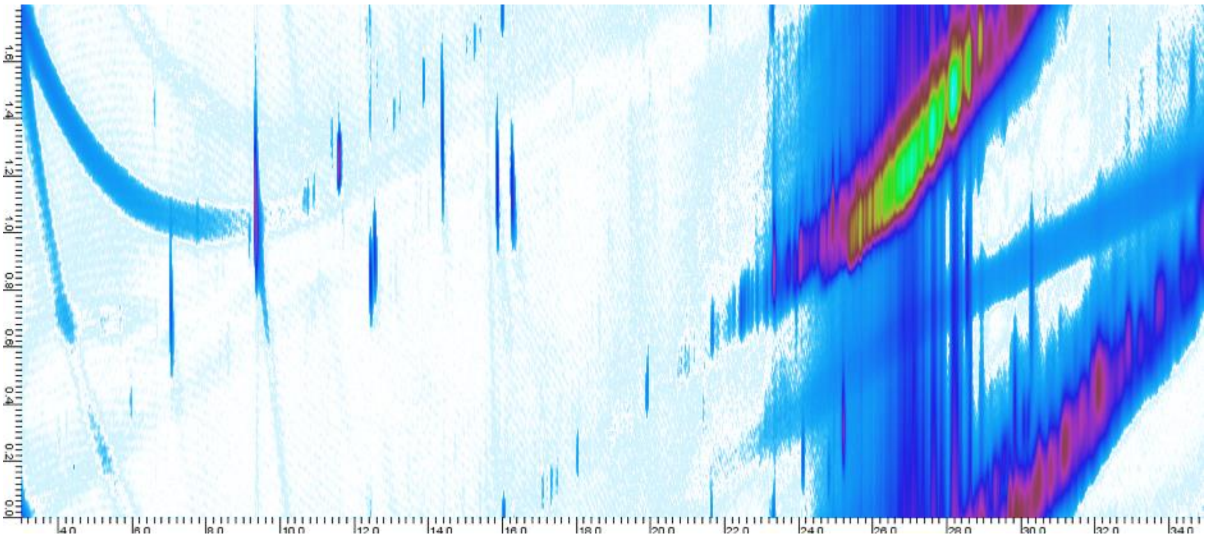


Figure 96 Light weight oil spiked with explosives

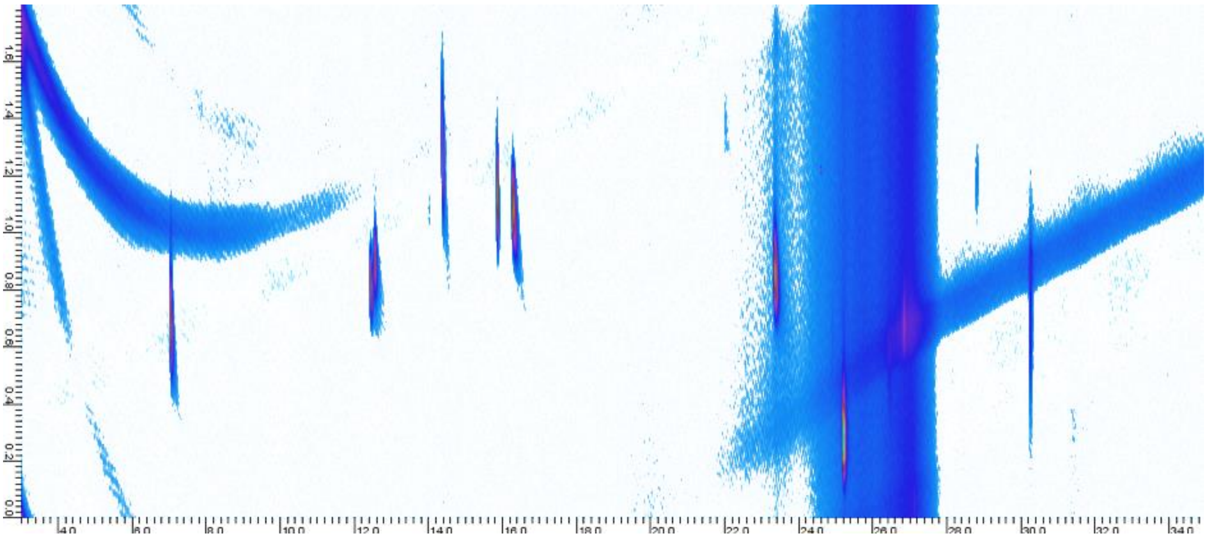


Figure 97 Porton Down aerosol sample spiked with CWA

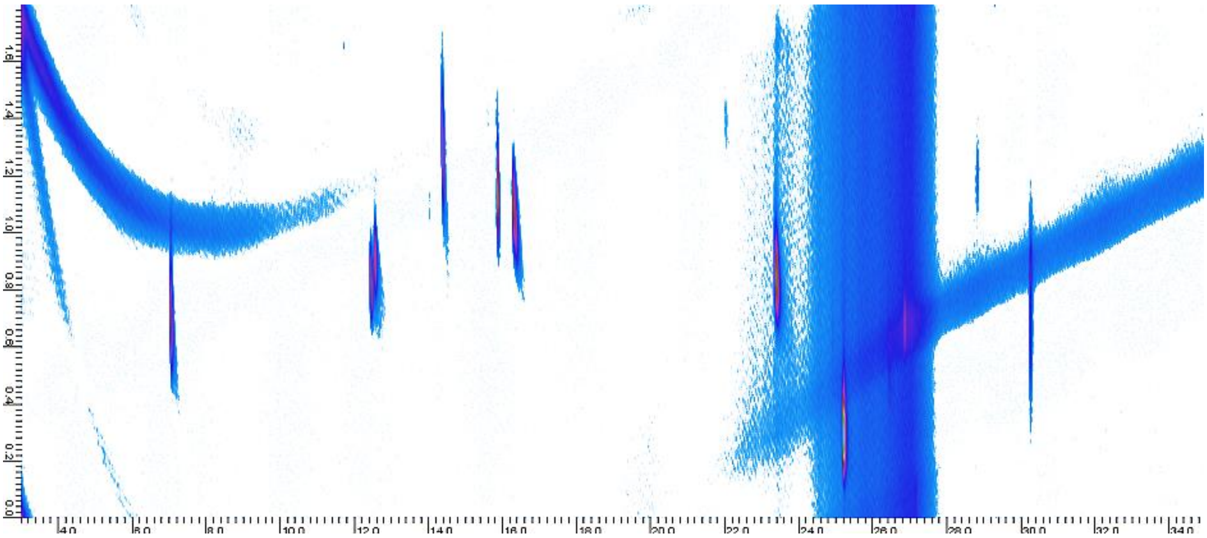


Figure 98 Porton Down aerosol sample 2 spiked with CWA

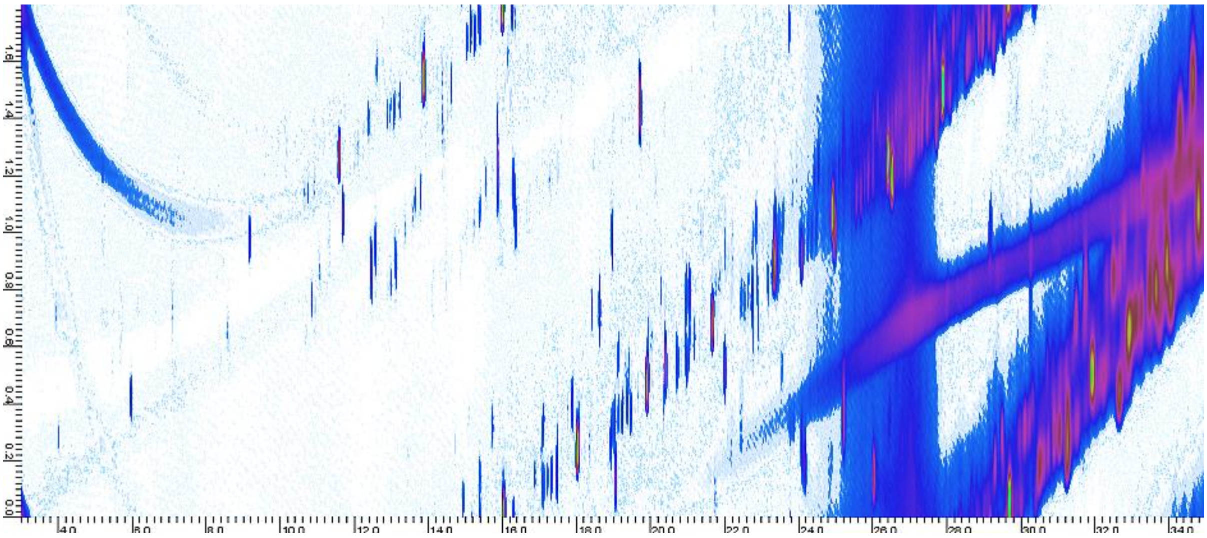


Figure 99 OMD-90 spiked with CWA

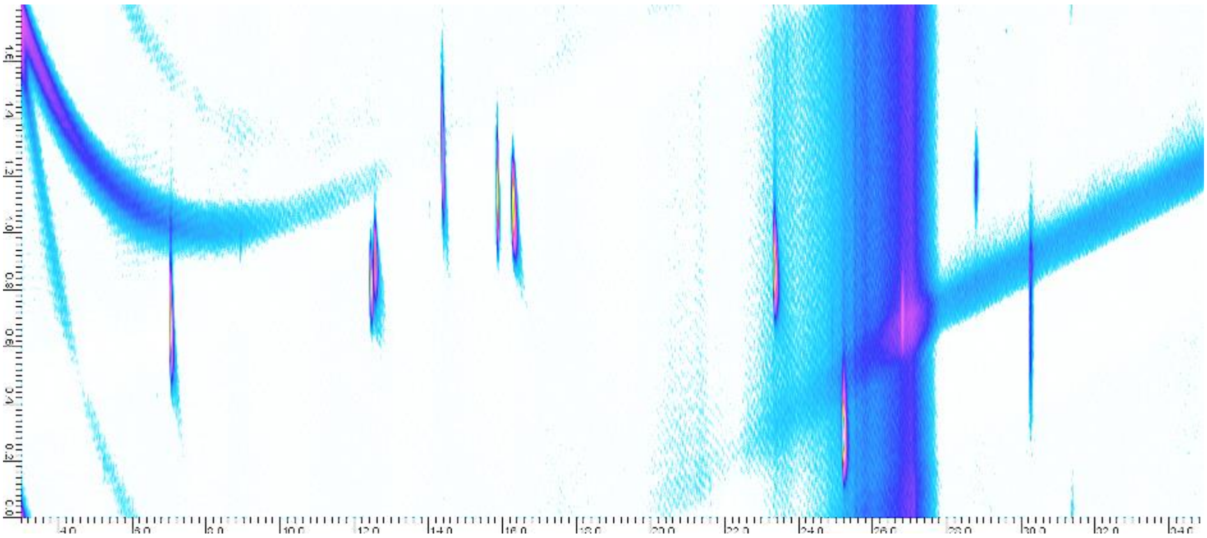


Figure 100 Swab of the inside of an oven spiked with CWA

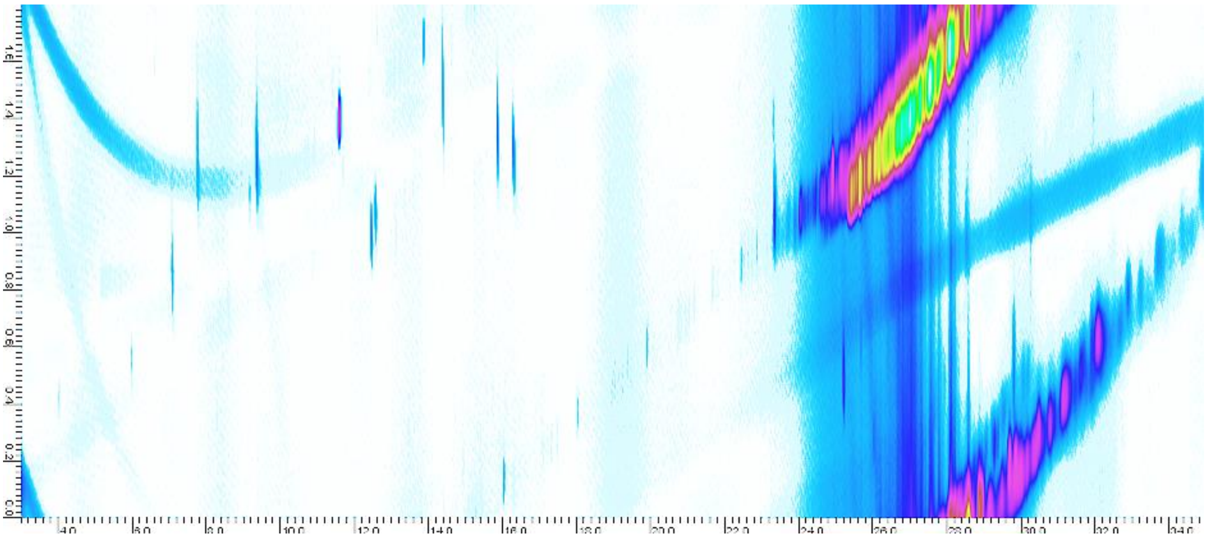


Figure 101 OX – 24 spiked with CWA

IV. Explosive Matrices Chromatograms

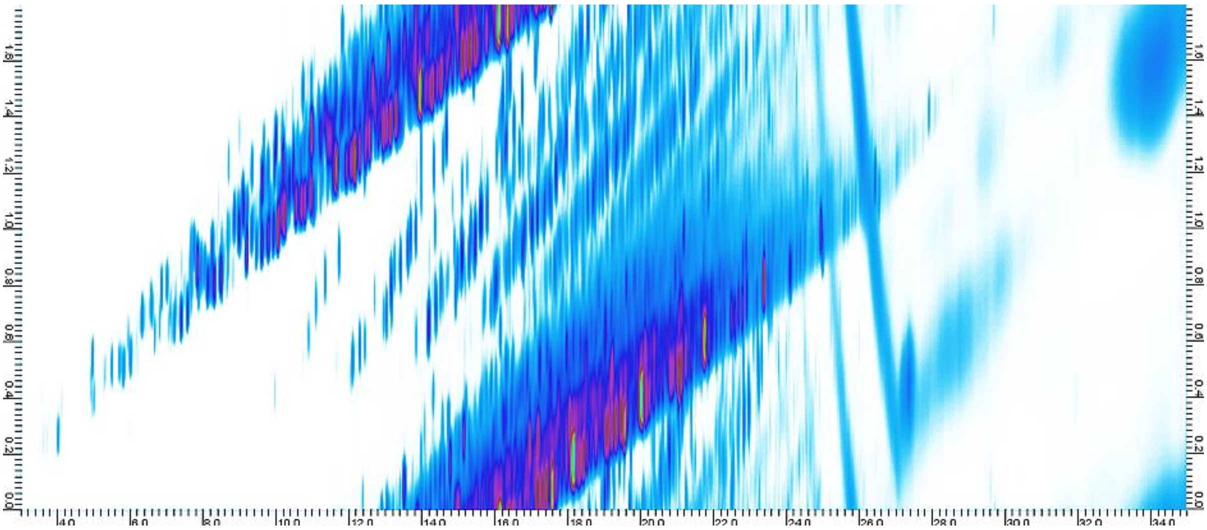


Figure 102 Aviation Fuel Spiked with 1 ppm of explosives

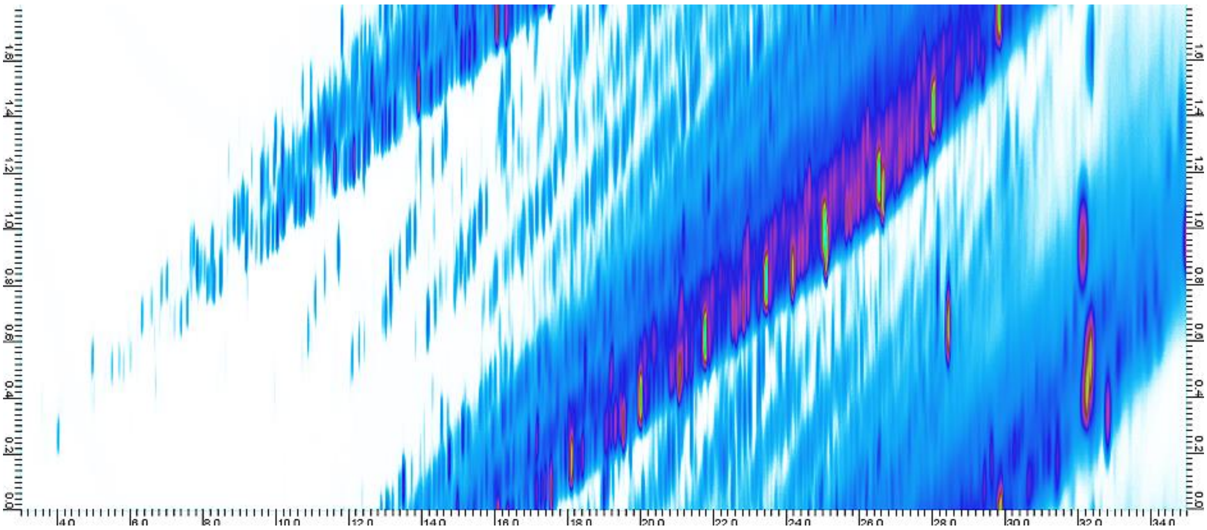


Figure 103 Diesel spiked with 1 ppm of explosives

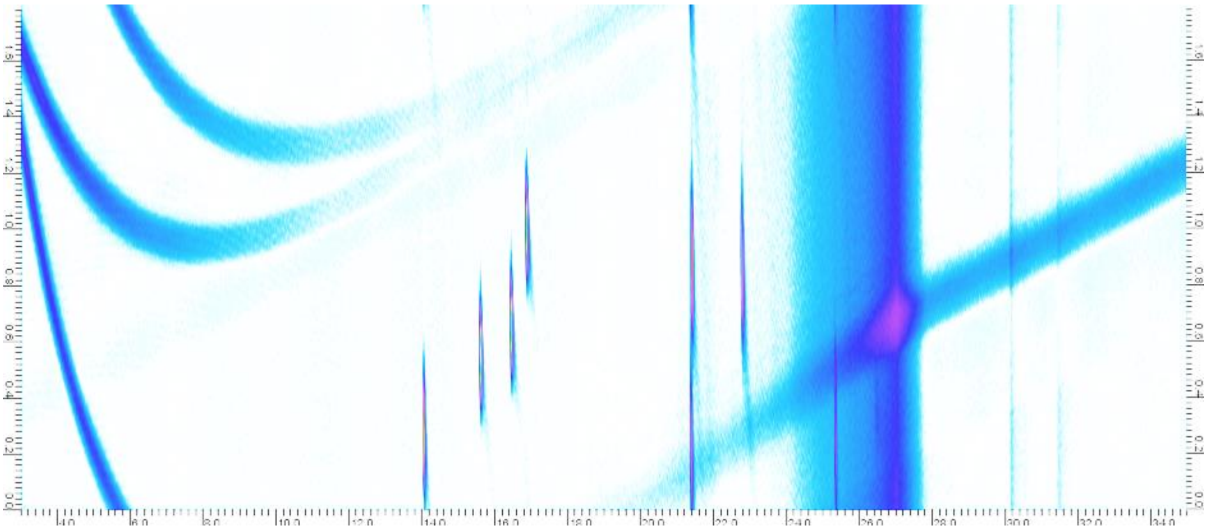


Figure 104 Sahara dust sample spiked with 1 ppm of explosives

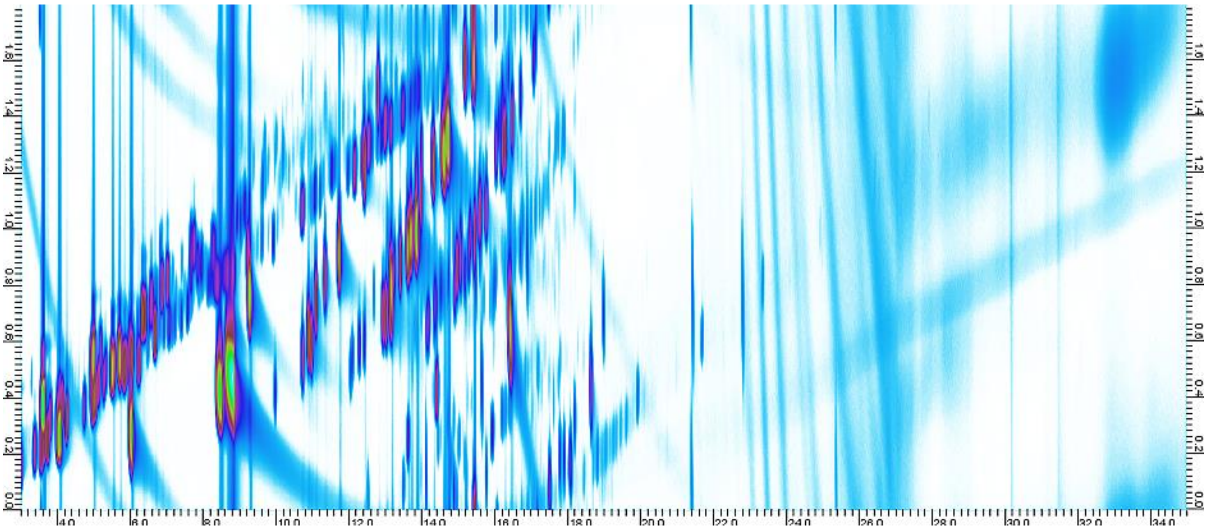


Figure 105 Gasoline spiked with 1 ppm of explosives

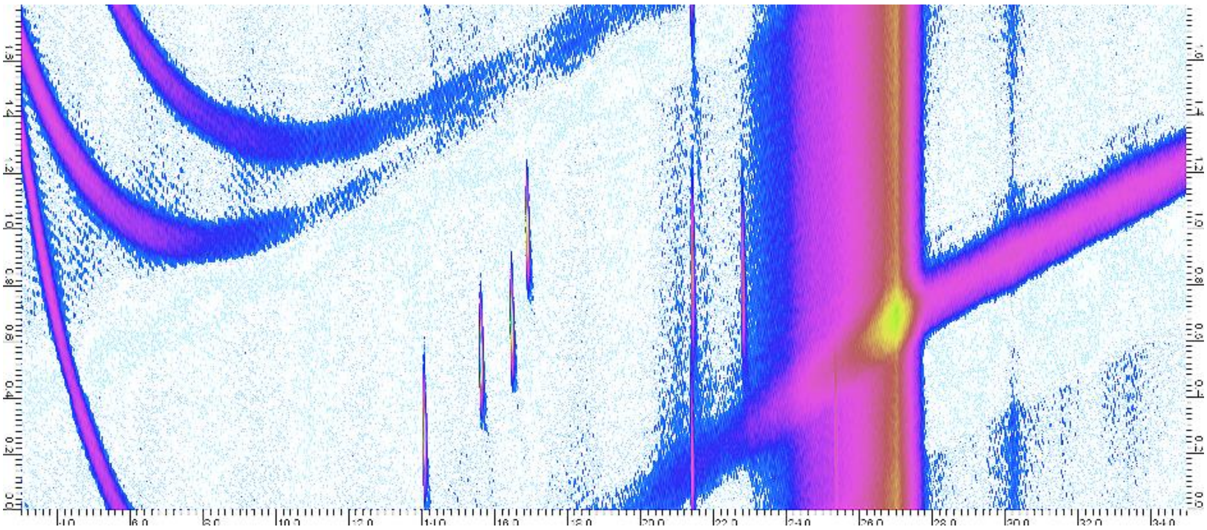


Figure 106 Swab sample from a Hotel with known interferents spiked with 1 ppm of explosives

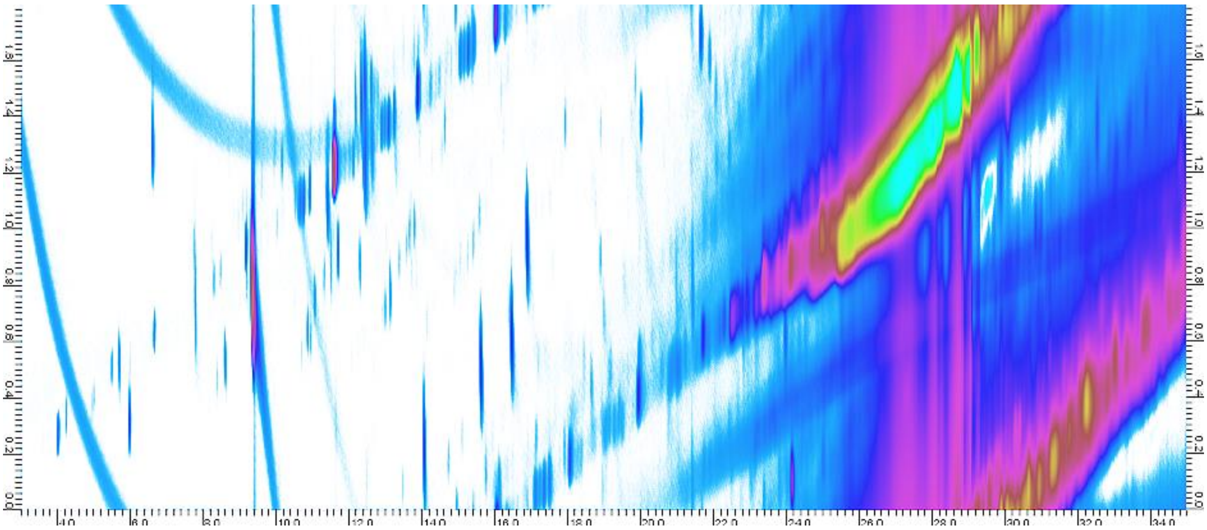


Figure 107 Light weight oil – used on guns – spiked with 1 ppm of explosives

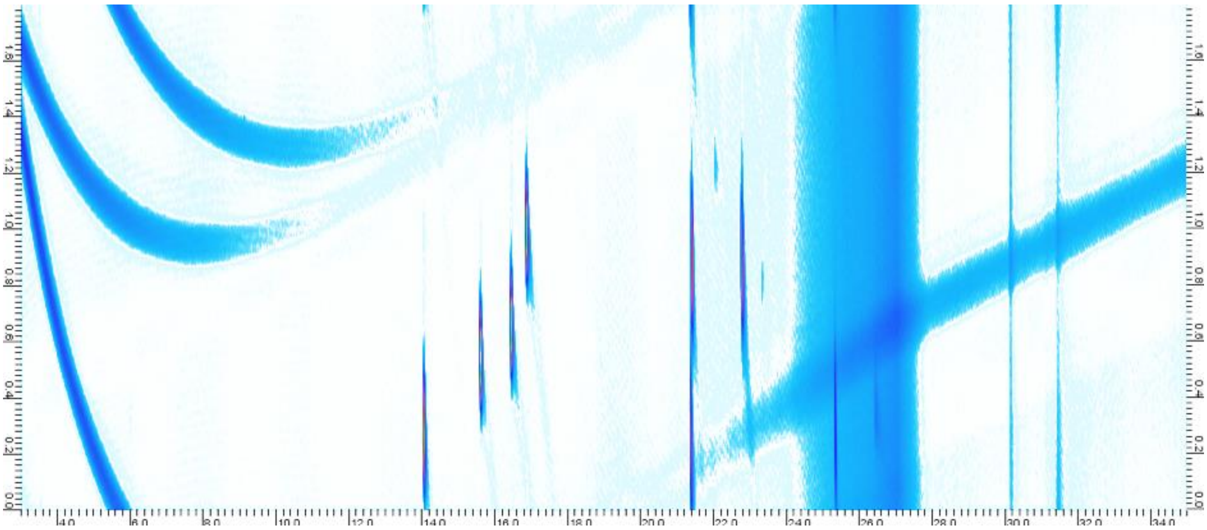


Figure 108 Porton range aerosol sample spike with 1 ppm of explosives

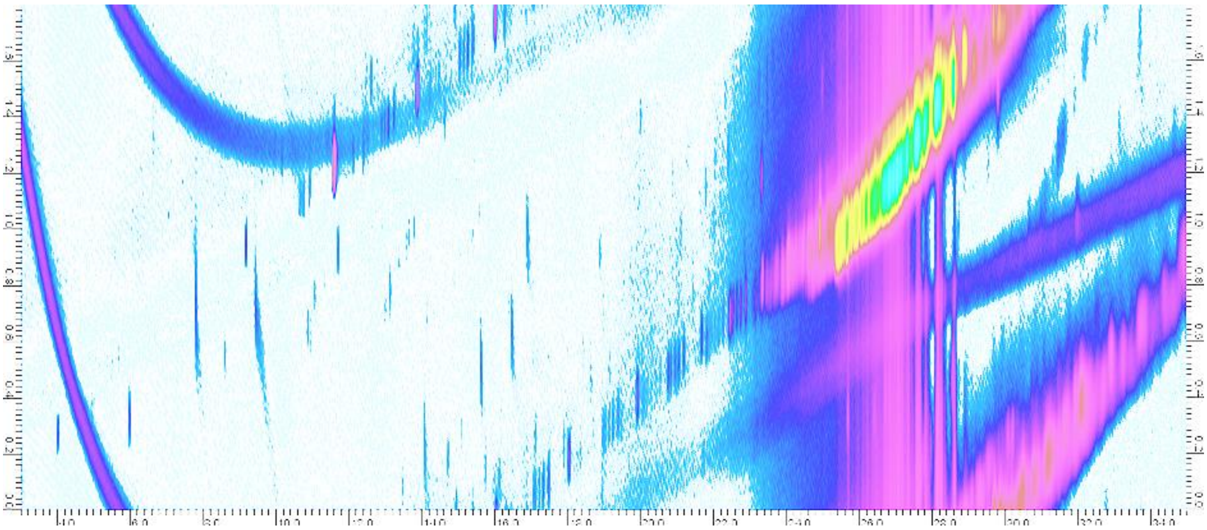


Figure 109 OX- 24 spiked with 1 ppm of explosives

V. CWA Matrices with Secondary Heater

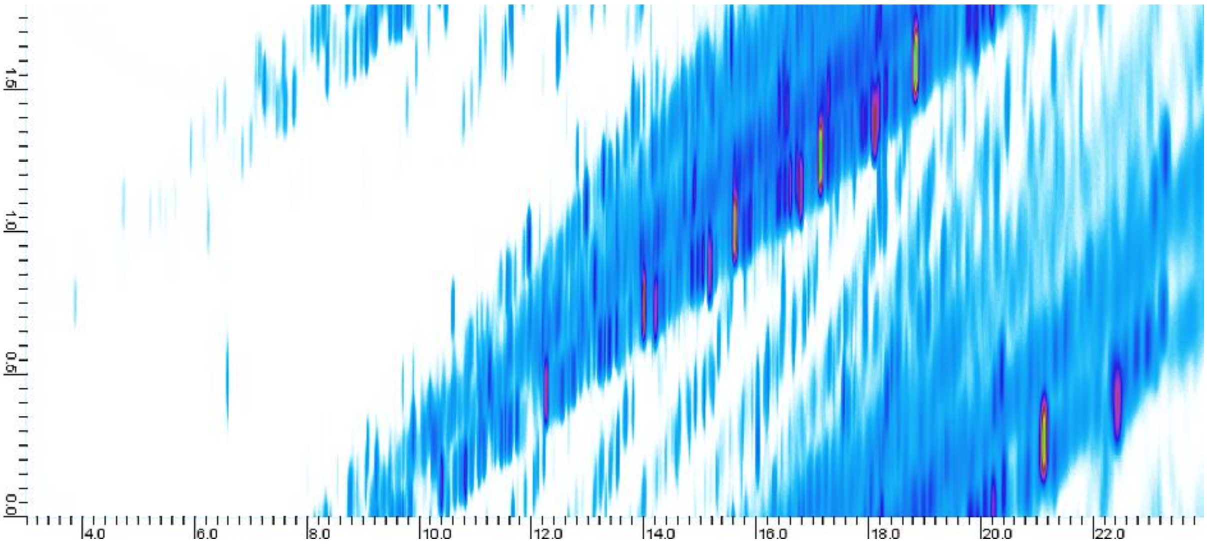


Figure 110 Diesel spiked with CWA standard of nerve agents

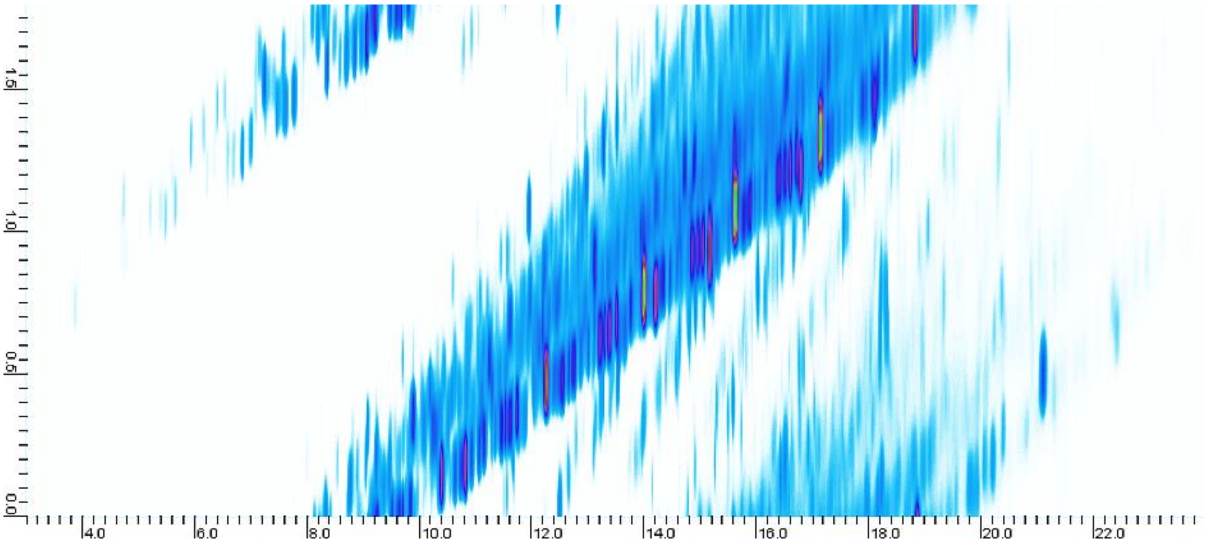


Figure 111 AV fuel spiked with CWA standard of nerve agents

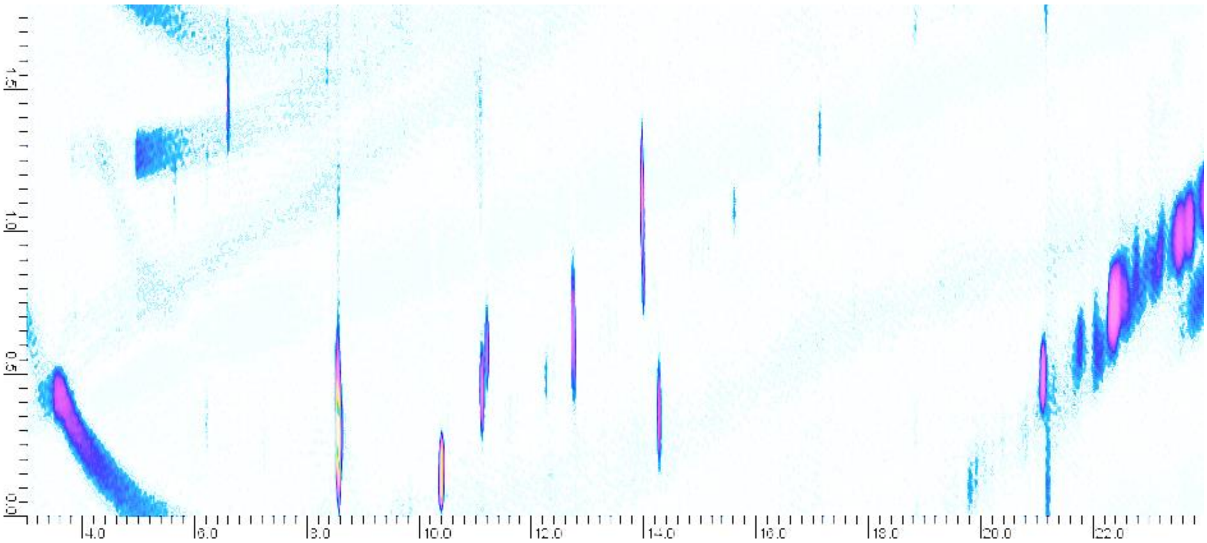


Figure 112 LWO spiked with CWA standard of nerve agents

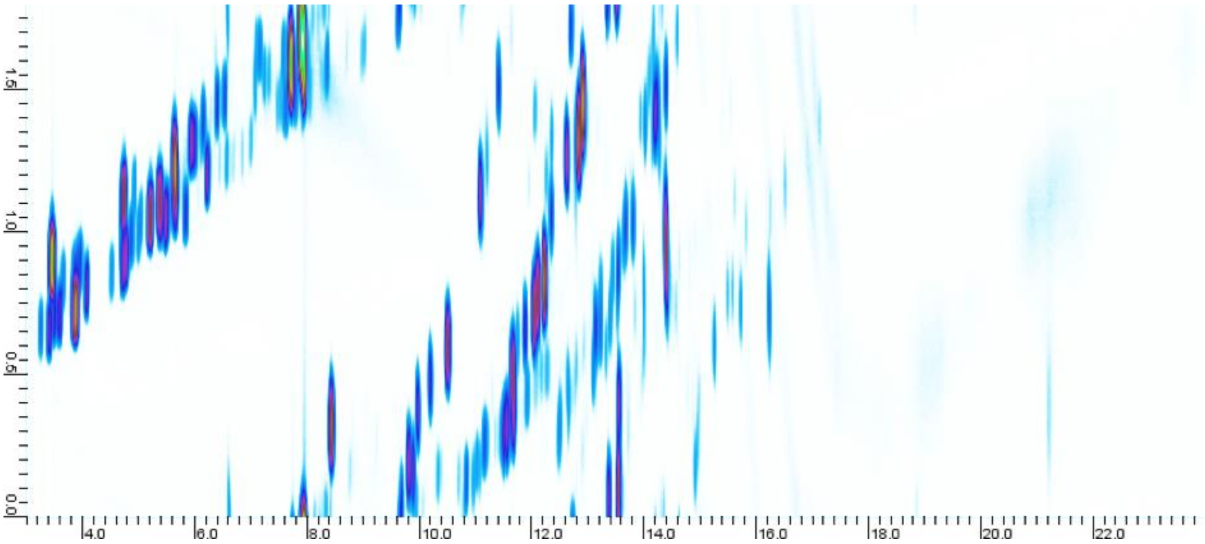


Figure 113 Gasoline Spiked with CWA standard of nerve agents

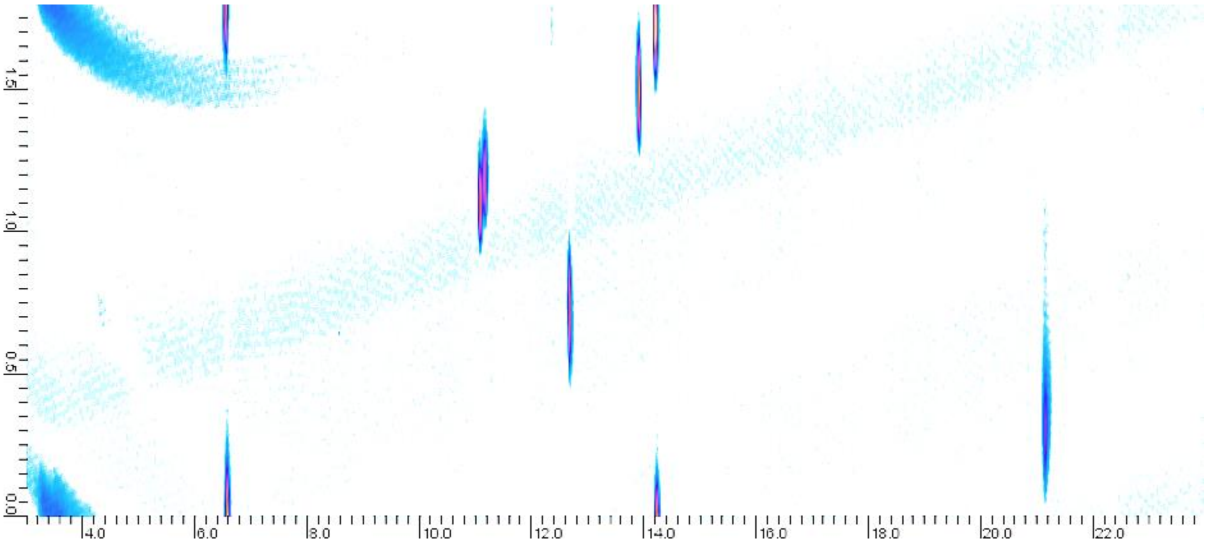


Figure 114 Swab of a hotel room spiked with CWA standard of nerve agents

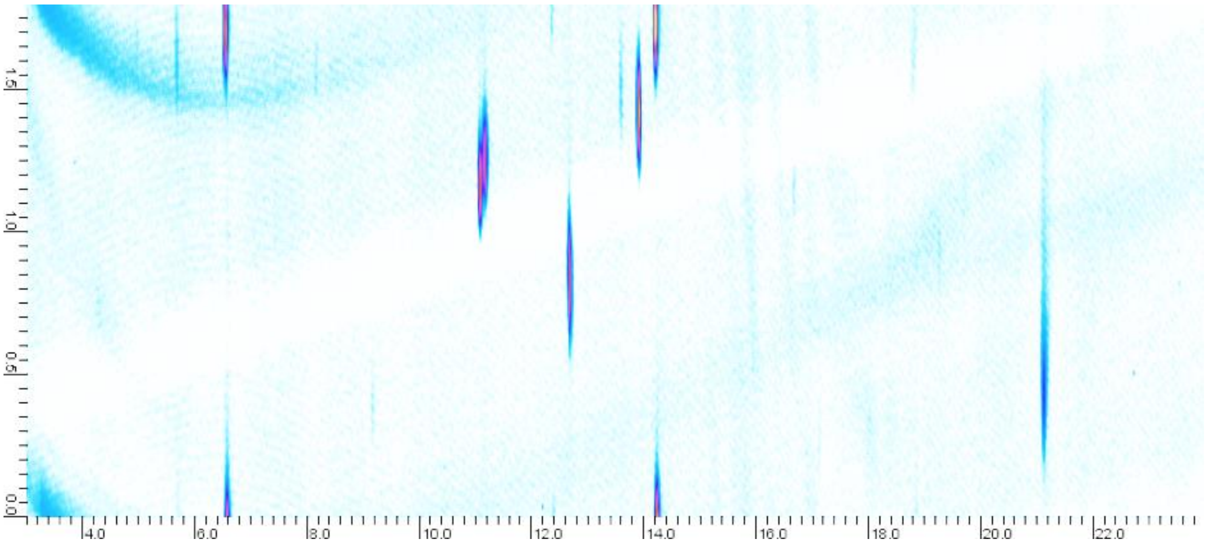


Figure 115 Swab of inside an oven spiked with CWA standard of nerve agents

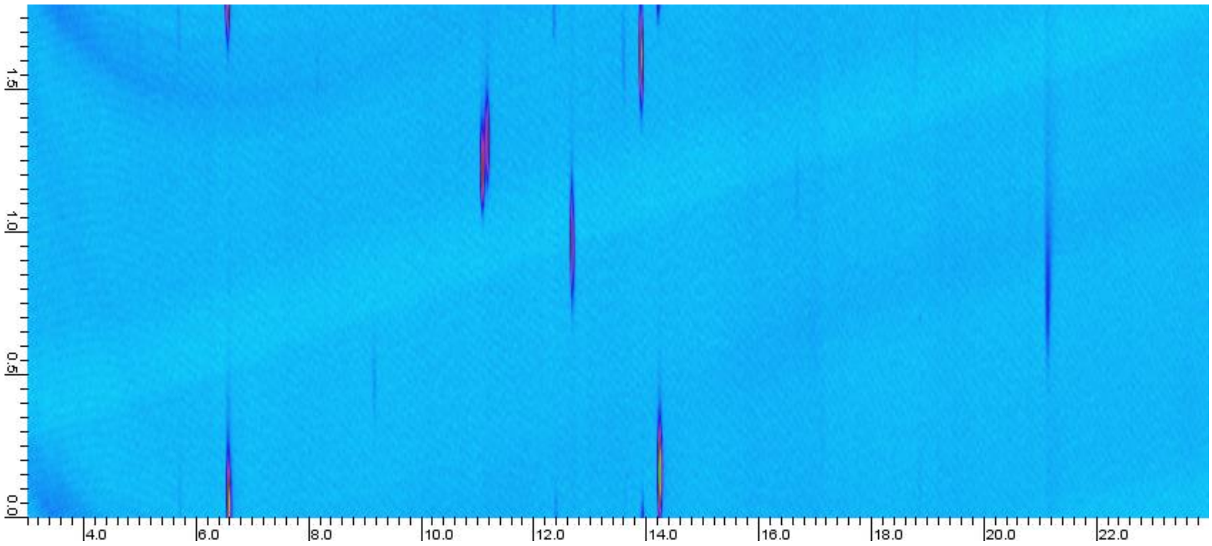


Figure 116 Swab of Clothing spiked with CWA standard of nerve agents

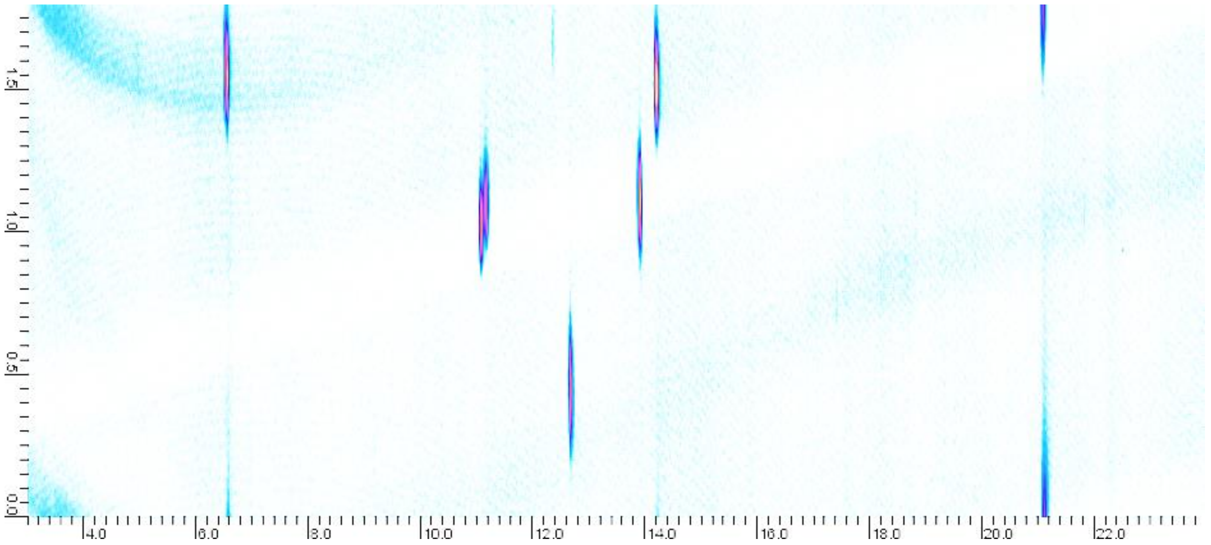


Figure 117 Aerosol sample from Porton Down spiked with CWA standard of nerve agents

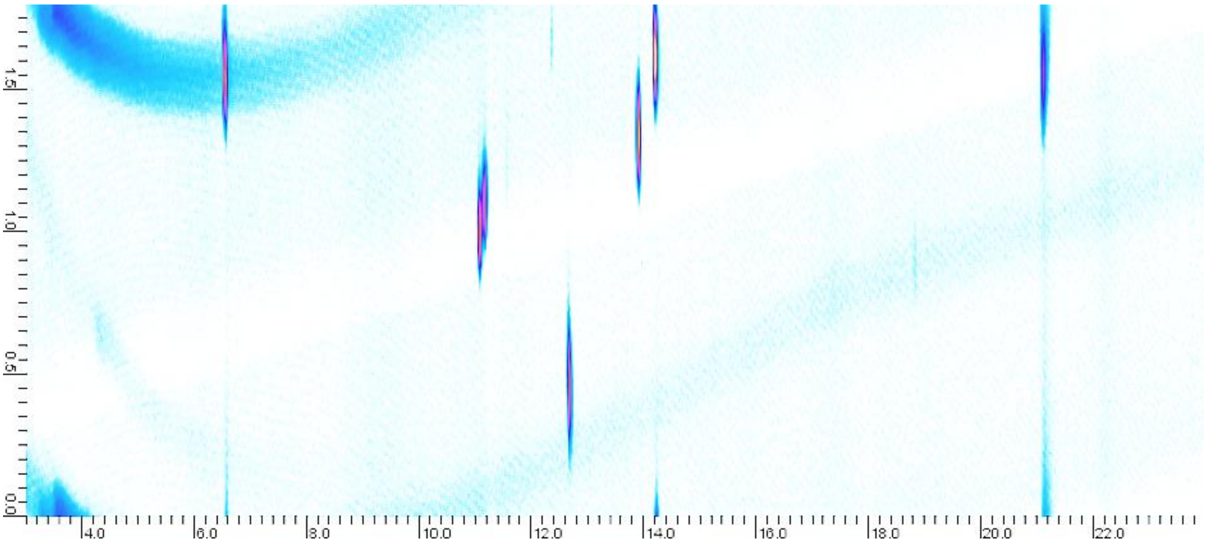


Figure 118 Sahara Dust aerosol sample spiked with CWA standard of nerve agents

VI. Secondary Oven data with Explosives Spiked into Matrices

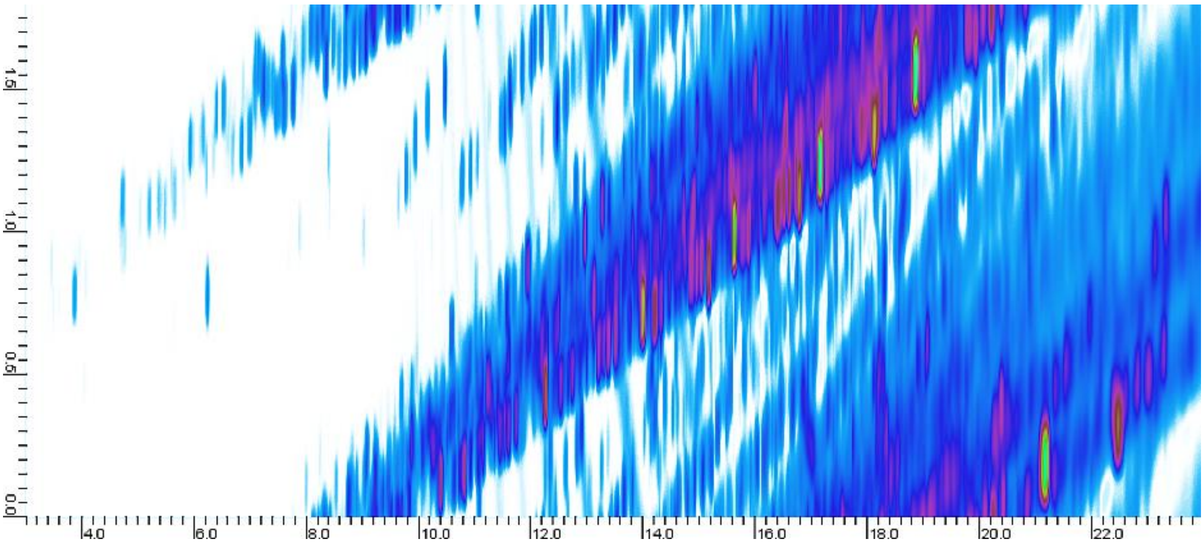


Figure 119 Diesel spiked with 8330 standard of explosives

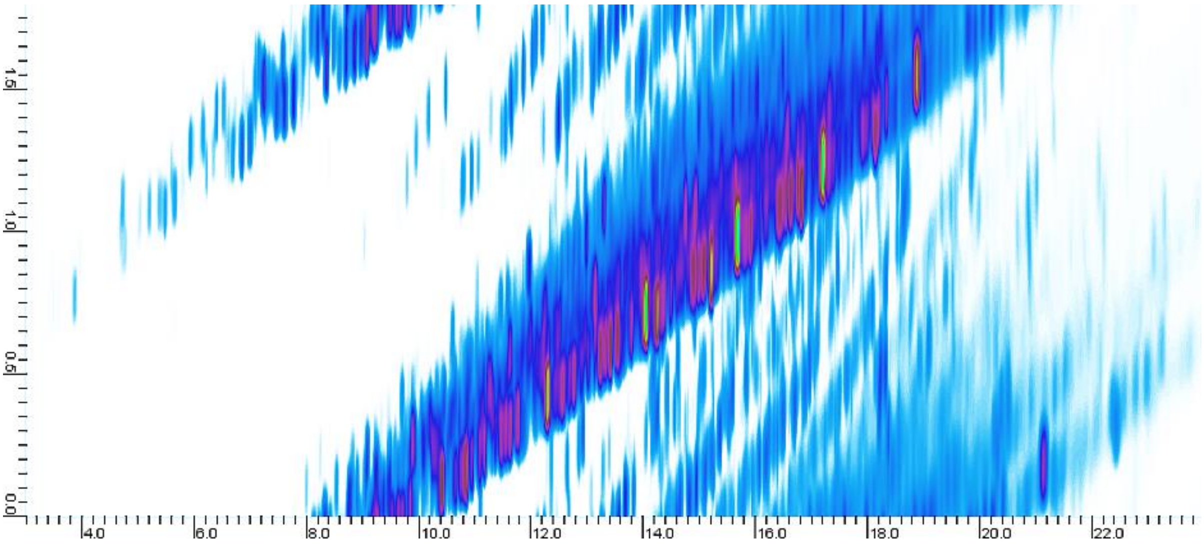


Figure 120 AV fuel spiked with 8330 standard of explosives

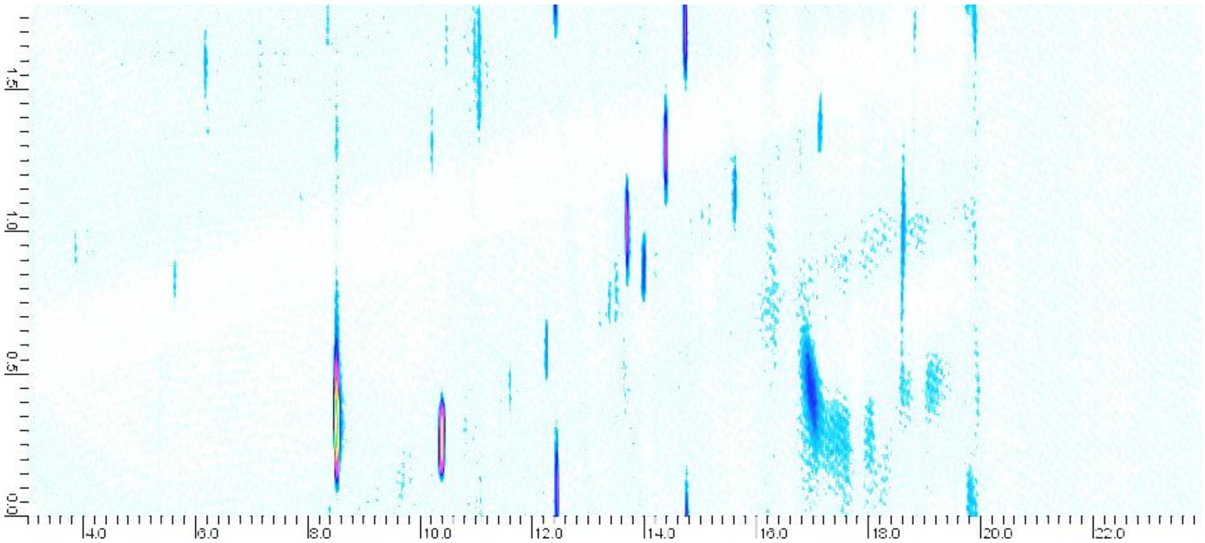


Figure 121 LWO spiked with 8330 standard of explosives

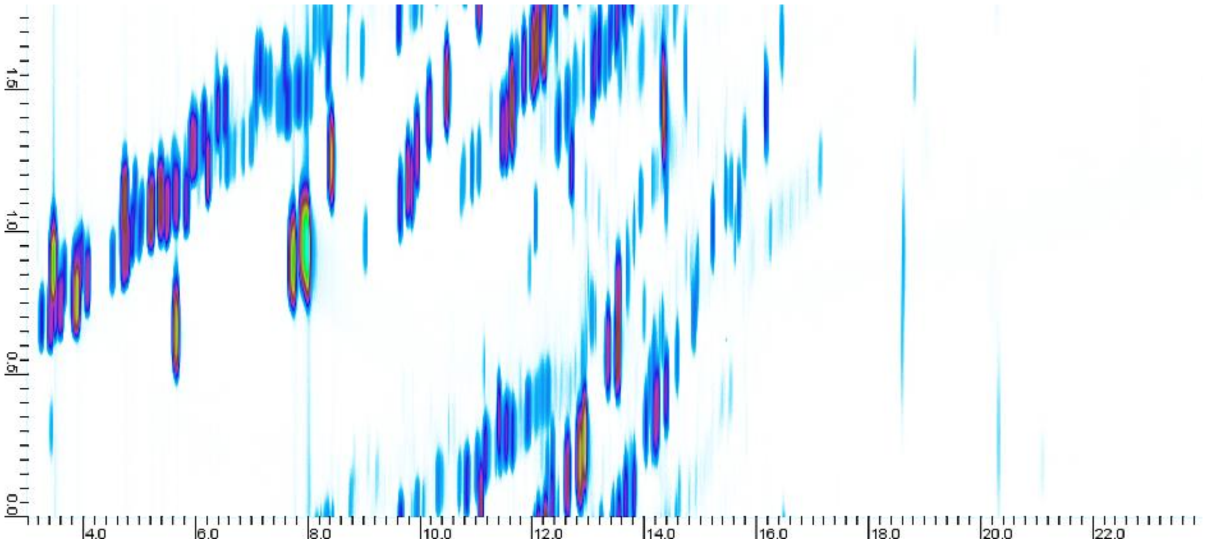


Figure 122 Gasoline Spiked with 8330 standard of explosives

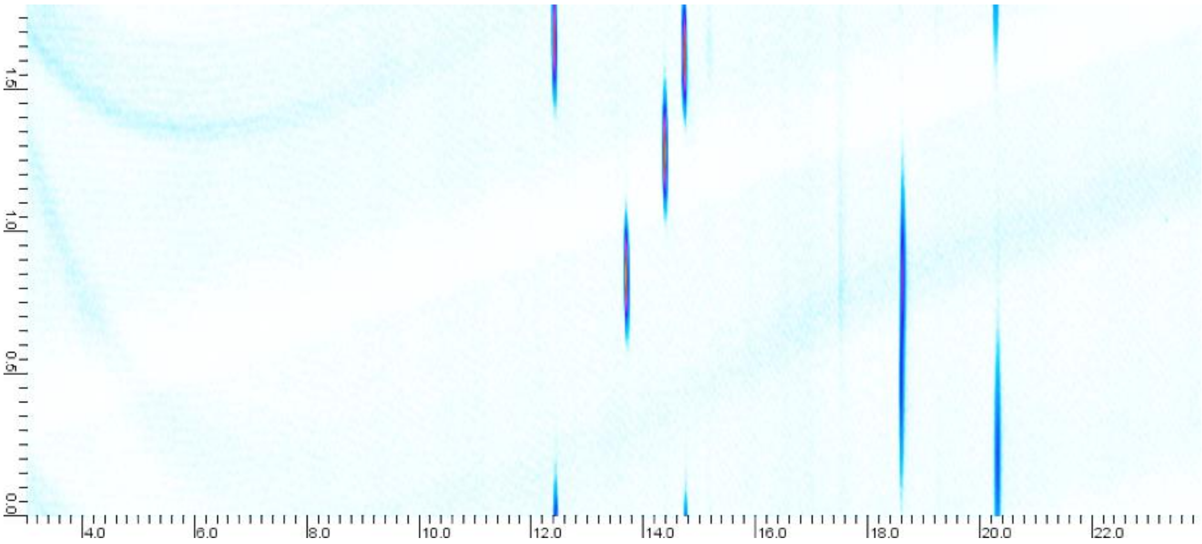


Figure 123 Swab of a hotel room spiked with 8330 standard of explosives

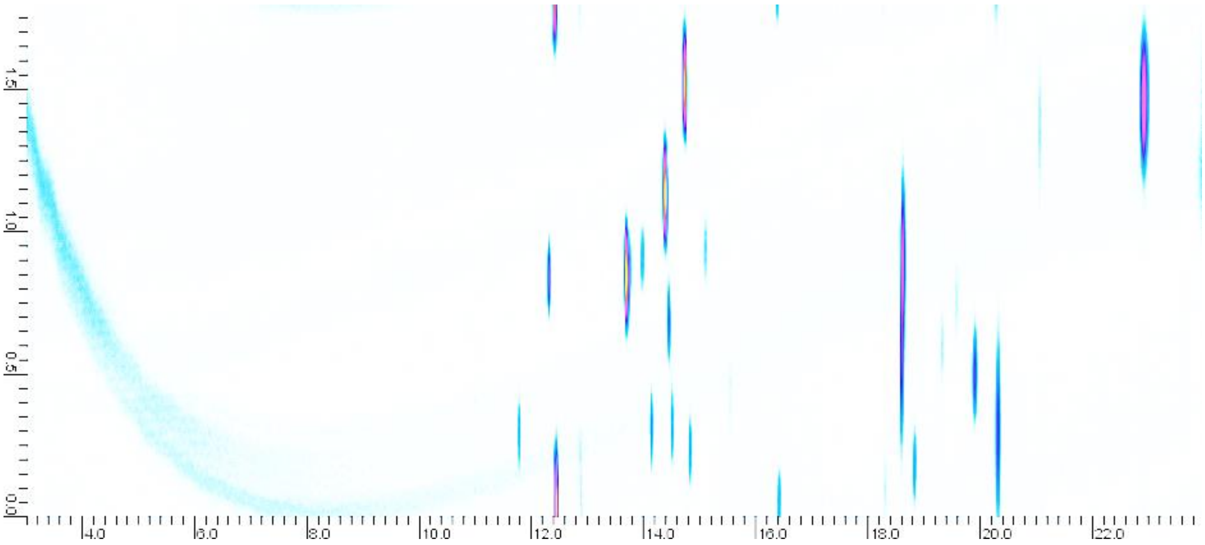


Figure 124 Perfume spiked with 8330 standard of explosives

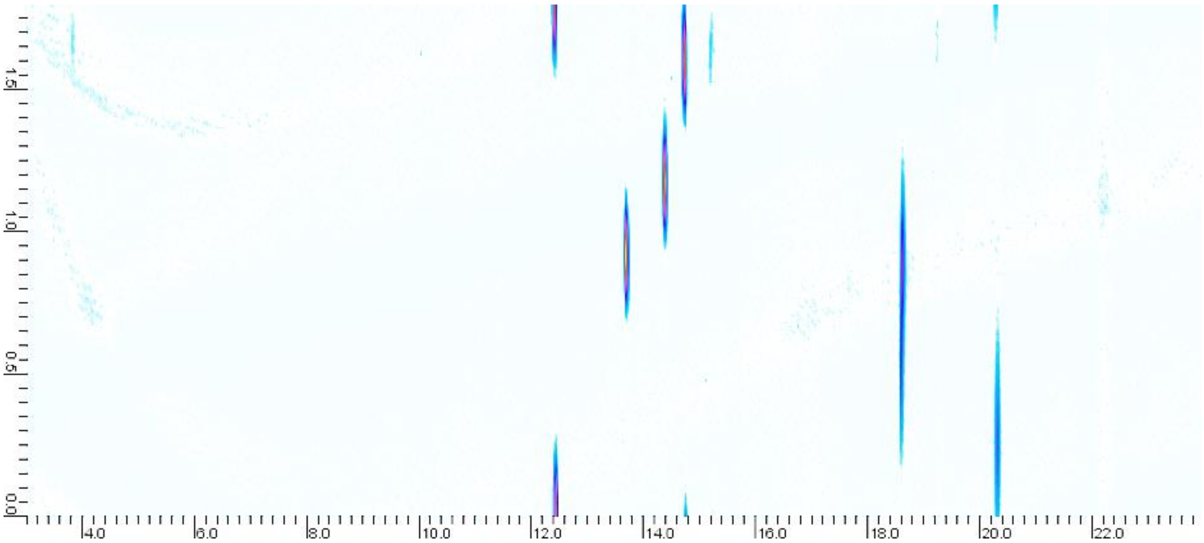


Figure 125 Aerosol sample from Porton Down spiked with 8330 standard of explosives

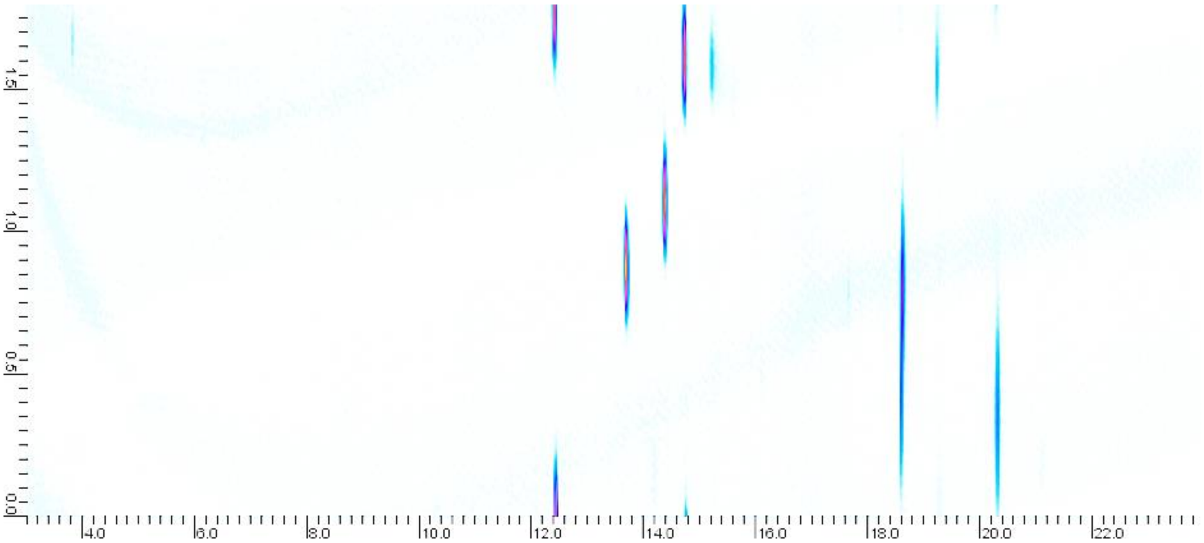


Figure 126 Sahara Dust aerosol sample spiked with 8330 standard of explosives

Annex 3

I. GC Slave

```
GC Slave
/**********************************************************************
#include <Wire.h>
#include <Adafruit_MAX31856.h>              // thermocouple library

#define               rx  0                 // receiving pin
#define               tx  1                 // transmission pin
#define              LED  2                 // LED pin
#define               H2  3                 // heater output pin
#define              SSR  4                 // heater output pin
#define               H1  5                 // heater output pin
#define               H4  6                 // heater output pin
//#define               ??  7                 // unused pin
//#define               ??  8                 // unused pin
#define               H3  9                 // heater output pin
#define               CS  10                // pin 10, chip select
#define               DI  11                // pin 12, data out of Arduino
#define               DO  12                // pin 12, data in to Arduino
#define              CLK  13                // pin 13, clock

#define              ID0  5                 // binary identifier analog pin
#define              ID1  6                 // binary identifier analog pin
#define              ID2  7                 // binary identifier analog pin

#define         maxPower  250               // maximum allowed power output
#define        heaterNum  1                 // number of heaters
#define    transInterval  150               // interval between data transmission
#define     calcInterval  1000              // milliseconds between calculations
#define      pulsePeriod  1024              // milliseconds between SSR ouput
pulses
#define           packet  12                // number of bytes in the message
stream
#define       tempOffset  16                // temperature offset for serial
transmission

#define       inletPower  30                // PWM power (of 255)
#define      outletPower  240               // PWM power (of 255)

byte myByte             = 0;                // the byte that this slave reads
byte valvePower         = 0;                // logging requested power going to the
valve
bool dataIsGood         = LOW;              // flag if checksum is good
byte dataByte[packet]   = {0};              // define array of output bytes
byte checksum           = 0;                // for caulcuating checksum of
transmission
long lastSerial         = 0;                // time stamp of last good serial
communication
long lastRead           = 0;                // time stamp of last temperature read
long lastCalc           = 0;                // time stamp of last power calculation
long nextPulse          = 0;                // time stamp of next SSR pulse
long onTime             = 0;                // time that the SSR has been on
long offTime            = 0;                // time that the SSR will turn off
bool pulse              = LOW;              // SSR pulse state

int  ID                 = 1;                // calculated in setup() but always
starts from 1
int  setPoint           = 0;                // setPoint for this device
int  lastSP             = 0;                // storing the last setpoint
int  error              = 0;                // for passing on errors

float temperature[heaterNum];               // array of sensor temperatures, in °C
float lastTemp[heaterNum];                  // array of previous sensor
temperatures, in °C
float power[4];                             // array of power outputs
float deltaTemp[heaterNum];                 // rate of change of temperature

Adafruit_MAX31856 max = Adafruit_MAX31856(CS);

// *******************************************************
void setup() {

  Serial.begin(115200);                     // serial comm baud rate
  max.begin();                              // start thermocouple board
  pinMode(rx, OUTPUT);                      // serial transmit pin
  pinMode(tx, INPUT);                       // serial transmit pin
  pinMode(LED, OUTPUT);                     // for LED blink
  pinMode(SSR, OUTPUT);                     // for SSR output
  pinMode(H1, OUTPUT);                      // for PWM output
  pinMode(H2, OUTPUT);                      // for PWM output
  pinMode(H3, OUTPUT);                      // for PWM output
  pinMode(H4, OUTPUT);                      // for PWM output

  max.setThermocoupleType(MAX31856_TCTYPE_K); // set thermocouple type

  // which slave am I? determined by binary input on analog inputs
  if (analogRead(ID0) > 512){ ID += 1; }    // add 1
  if (analogRead(ID1) > 512){ ID += 2; }    // add 2
  if (analogRead(ID2) > 512){ ID += 4; }    // add 4

  blink_LED(ID, 50, 250);                   // blink the LED to confirm slave ID
  myByte = 2 * ID - 1;                      // calculate which byte to read based
on ID
  read_temperature();                       // read the thermocouple
temperature/error

  while (millis() <= 2500) {}               // wait until timer is at 2 seconds
  nextPulse = 5000;                         // update nextPulse (not needed?)
  lastSerial = millis();
}

// *******************************************************
void loop() {

  // check for incoming serial data
  receive_serial();                                 // always check for incoming
data

  // compute and resend data
  if (dataIsGood){                                  // if serial has been received
and is good
    compute_data();                                 // read setPoint and update the
data
    transmit_serial();                              // send serial data
    dataIsGood = LOW;                               // clear flag
    blink_LED(1, 10, 0);                            // one blink, 10ms on, 0ms off
  }
  if (millis() - lastRead >= calcInterval/2){       // read temperature twice per
calculation
    read_temperature();                             // read temperatures
    lastRead = millis();                            // update last calculation time
  }

  // calculate power factor
  if (millis() - lastCalc >= calcInterval){         // only change power at set
times
    lastCalc = millis();                            // update last calculation time
    read_temperature();                             // read temperatures
    if (ID == 1)           { valve_power();}        // calculate valve power factor
    if (ID == 2)           { heater_power();}       // calculate valve heater power
factor
    if (ID == 3 || ID == 4){ column_power();}       // calculate column power
factor
    if (ID == 5)           { transfer_power();}     // calculate valve power factor
  }

  // set outputs
  set_pulse();                                      // set SSR pulse length
  set_outputs();                                    // write the output power

  // check for errors

  if (millis() - lastSerial > 2500){
    setPoint = 0;                                   // turn off heaters
    digitalWrite(LED, HIGH);                        // set LED high
    delay(10);
  }

}

// *******************************************************
void receive_serial() {
  if (Serial.available() >= packet){                // if there is enough data int
the buffer
    delay(1);                                       // this delay seems to help!
    checksum = 0;                                   // clear the checksum
    dataIsGood = LOW;                               // clear data good flag
    for (int i = 0; i < packet-1; i++){             // loop through each byte
      dataByte[i] = Serial.read();                  // store to array of inputs
      checksum = checksum ^ dataByte[i];            // calculate checksum on the
fly
    }
    dataByte[packet-1] = Serial.read();             // read in the last byte
    // compare checksums && check first byte is a header
    if (dataByte[packet-1] == checksum && dataByte[0] == 1){
      lastSerial = millis();                        // log the last time good data
was received
      dataIsGood = HIGH;                            // flag it as good data to
process
    }
    while (Serial.available() > 0){                 // if there is still data in
the buffer
      byte trash = Serial.read();                   // read it to trash
    }
  }
}

// ***************************************************
void compute_data() {
    byte statusByte = dataByte[myByte];             // read my status byte to
variable
    if (statusByte == 5){                           // status 5: confirms that this
is incoming data
      if (!error){                                  // assuming there are no
internal errors
        setPoint = dataByte[myByte+1] + tempOffset; // update the local setPoint
from my data byte
        if (setPoint){                              // if heating is on (i.e.
setPoint not zero)
          int p = map(power[0], 0, maxPower, 0, 100); // convert power to
percentage
          dataByte[myByte] = 128 + p;               // send power percentage
        }
        else { dataByte[myByte] = 6; }              // return an acknowledgement
but heater is off
      }
      else { dataByte[myByte] = error; }            // return an error code
    }
    else { dataByte[myByte] = 21; }                 // return error that the
incoming data is not correct

    dataByte[myByte+1] = temperature[0] - tempOffset; // always return live
temperature data

    // read valve temperature for
    if (dataByte[1] >= 128){                        // read the power requested
from the valve block slave
      valvePower = map(dataByte[1], 128, 228, 0, maxPower);  // convert to one byte
power factor
      valvePower = constrain(valvePower, 0, maxPower);
    }
    error = 0;                                      // clear errors in case they
are fixed
}

// ***************************************************
void transmit_serial() {
  checksum = 0;                                     // clear the checksum
  for (int i = 0; i < packet -1; i++){              // for header and all data
bytes
    Serial.write(dataByte[i]);                      // send serial data
    checksum = checksum ^ dataByte[i];              // calculate checksum on the
fly
  }
  dataByte[packet-1] = checksum;                    // last byte is the checksum
  Serial.write(dataByte[packet-1]);                 // send the last byte
}

// *****************************************************
void read_temperature(){
  if (ID == 5) {                                    // for the transfer lines
    if (setPoint == 0){
      temperature[0] = 32;
    }
    else {
      temperature[0] = constrain(setPoint, 22, 254);  // just return the setpoint
    }
    delay(50);
  }
  else {
    for (byte h = 0; h < heaterNum; h++){           // for each of the heaters
      temperature[h] = max.readThermocoupleTemperature(); // read in live
temperature
      if (temperature[h] <= 5)            { error = 38; }
    }
    int fault = max.readFault();                    // check for thermocouple
errors
    if (fault) {
      if (fault & MAX31856_FAULT_CJRANGE) { error = 30; }
      if (fault & MAX31856_FAULT_TCRANGE) { error = 31; }
      if (fault & MAX31856_FAULT_CJHIGH)  { error = 32; }
      if (fault & MAX31856_FAULT_CJLOW)   { error = 33; }
      if (fault & MAX31856_FAULT_TCHIGH)  { error = 34; }
      if (fault & MAX31856_FAULT_TCLOW)   { error = 35; }
      if (fault & MAX31856_FAULT_OVUV)    { error = 36; }
      if (fault & MAX31856_FAULT_OPEN)    { error = 37; }
    }
  }
}

// *****************************************************
void column_power(){
  for (byte h = 0; h < heaterNum; h++){             // for each of the heaters
    if (setPoint){                                  // if heaters are turned on
      float tempToSet = setPoint - temperature[h];  // distance to setPoint
      float deltaTemp = temperature[h] - lastTemp[h]; // change in temperature
      float deltaSP   = setPoint - lastSP;          // change in setpoint
      double newPower = 0;

      // newPower = [oldPower] + [temp to setPoint} + [change in setpoint] -
[change in temp]
      // **************************************** // more than 1° below setpoint
      if (tempToSet > 1){
        newPower = power[h] + 0.5 * tempToSet + 70 * (deltaSP - deltaTemp);
      }
      // **************************************** // less than 0.5° above
setpoint
      else if (tempToSet > -0.5){
        newPower = power[h] + 0.5 * tempToSet + 50 * (deltaSP - deltaTemp);
      }
      // ****************************************** // more than 0.3° above
setpoint
      else {
        newPower = power[h] + 3.0 * tempToSet + 60 * (deltaSP - deltaTemp);
      }

      power[h] = constrain(newPower, 0, maxPower);  // limit power output

      if (deltaTemp < 0.1 && power[h] == 100){      // check for slow temp change
        blink_LED(1, 100, 0);
      //  error = 40;                               // set heater error
      }
      lastSP = setPoint;                            // backup last setpoint
    }

    else {                                          // if heaters are turned off
      power[h] = 0;                                 // set power to zero
      lastSP = 250;                                 // set lastSP high to stop
initial spike
    }
    lastTemp[h] = temperature[h];                   // backup last temperature
  }

}

// *******************************************************
void valve_power(){
  for (byte h = 0; h < heaterNum; h++){             // for each of the heaters
    if (setPoint){                                  // if heaters are turned on
      float tempToSet = setPoint - temperature[h];  // distance to setPoint
      float deltaTemp = temperature[h] - lastTemp[h]; // change in temperature
      double newPower = 0;

      // newPower = [oldPower] + [temp to setPoint] - [change in temp] + [change in
setpoint]
      // ****************************************** // more than 10° below setpoint
      if (tempToSet > 10){
        newPower = power[h] + 1.0 * tempToSet - 200 * deltaTemp;
      }
      // ****************************************** // less than 0.5° above
setpoint
      else if (tempToSet > -0.5){
        newPower = power[h] + 0.25 * tempToSet - 60 * deltaTemp;
      }
      // ****************************************** // more than 0.5° above
setpoint
      else {
        newPower = power[h] + 8.0 * tempToSet - 20 * deltaTemp;
      }

      power[h] = constrain(newPower, 0, maxPower);  // limit power output

    }

    else {                                          // if heaters are turned off
```

```
      power[h] = 0;                                 // set power to zero
      lastSP = 250;                                 // set lastSP high to stop
initial spike
    }
    lastTemp[h] = temperature[h];                   // backup last temperature
  }

}
// *******************************************************
void heater_power(){
  for (byte h = 0; h < heaterNum; h++){             // for each of the heaters
    if (setPoint){                                  // if heaters are turned on
      float tempToSet = setPoint - temperature[h];  // distance to setPoint
      float deltaTemp = temperature[h] - lastTemp[h]; // change in temperature
      double newPower = 0;

      // newPower = [oldPower] + [temp to setPoint] - [change in temp] + [change in
setpoint]
      // **************************************** // more than 10° below setpoint
      if (tempToSet > 10){
        newPower = power[h] + 2.0 * tempToSet - 120 * deltaTemp;
      }
      // **************************************** // less than 0.5° above
setpoint
      else if (tempToSet > -0.5){
        newPower = power[h] + 1 * tempToSet - 60 * deltaTemp;
      }
      // **************************************** // more than 0.5° above
setpoint
      else {
        newPower = power[h] + 8.0 * tempToSet - 20 * deltaTemp;
      }

      power[h] = constrain(newPower, 0, valvePower); // limit power to that of the
valve

      if (deltaTemp < 0.1 && power[h] == 100){      // check for slow temp change
        blink_LED(1, 100, 0);
      //  error = 40;                               // set heater error
      }
    }

    else {                                          // if heaters are turned off
      power[h] = 0;                                 // set power to zero
      lastSP = 250;                                 // set lastSP high to stop
initial spike
    }
    lastTemp[h] = temperature[h];                   // backup last temperature
  }
}

// *******************************************************
void transfer_power(){
  if (setPoint){                                    // if heaters are turned on
    power[0] = map(setPoint, 32, 270, 0, 178);      // line between col2 and
detector
```

```
    power[1] = inletPower;                          // line between injector and
col1
    power[2] = 0;                                   // line between col1 and valve
    power[3] = 0;                                   // line between valve and col2
    delay(1);
  }
  else {                                            // if heaters are turned off
    for (byte h = 0; h < 4; h++){                   // for each of the heaters
      power[h] = 0;                                 // set power to zero
    }
  }
}

// *******************************************************
void set_pulse(){
  // set SSR output
  if (millis() >= nextPulse){                       // when time for next pulse
    onTime = millis();                              // note the time that SSR
turned on
    offTime = onTime + (4 * power[0]);              // calculate the off time
    if (power[0] > 1){                              // if the power is more than 1
(1 is too low!)
      pulse = HIGH;                                 // set the SSR pin high
    }
    else{                                           // otherwise if power is zero
      pulse = LOW;                                  // set the SSR pin low
    }
    nextPulse += pulsePeriod;                       // using += in case of delays
elsewhere in code
  }
  if (millis() >= offTime){                         // if it is time to turn off
SSR
    pulse = LOW;                                    // set the SSR pin LOW
  }
}

// *******************************************************
void set_outputs(){
  // set PWM power
  analogWrite(H1, power[0]);                        // set the power output
  analogWrite(H2, power[1]);                        // set the power output
  analogWrite(H3, power[2]);                        // set the power output
  analogWrite(H4, power[3]);                        // set the power output
  // set SSR output
  digitalWrite(SSR, pulse);                         // set the SSR pin LOW
  // set LEDs?
}

// *******************************************************
void blink_LED(int count, int on, int off) {
  while(count > 0){
    count--;
    digitalWrite(LED, HIGH);
    delay(on);
    digitalWrite(LED, LOW);
    delay(off);
  }
```

```
}
```

II. GC Master

```
/***********************************************************************

  0   1   2   3   4   5   6   7   8   9   10  11
  Hd  S1  D1  S2  D2  S3  D3  S4  D4  S5  D5  Ck

Header byte:
01        Start of data packet
Status byte:
05        Enquiry - master sent data
06        Acknowledge - slave response, but heaters are off
07        Error - slave returning an error
21        Negative - acknowlegde receipt but data is incorrect
30        Cold Junction Range Fault
31        Thermocouple Range Fault
32        Cold Junction High Fault
33        Cold Junction Low Fault
34        Thermocouple High Fault
35        Thermocouple Low Fault
36        Over/Under Voltage Fault
37        Thermocouple Open Fault
99        heater too slow responding
128-228   Power - if heating, this is the % power being applied (-128)

*******************************************************/

#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_ILI9340.h>
#include <Fonts/FreeSans9pt7b.h>
#if defined(__SAM3X8E__)
  #undef __FlashStringHelper::F(string_literal)
  #define F(string_literal) string_literal
#endif

#define                rx 0                 // receiving pin
#define                tx 1                 // transmission pin
#define              encA 2                 // encoder A pin
#define              encB 3                 // encoder B pin
#define            button 4                 // jog dial button pin
#define             relay 5                 // relay output
#define               fan 6                 // PWM fan output
#define            shtdwn 7                 // stop button input
#define              _rst 8                 // define screen pins
#define               _dc 9                 // define screen pins
#define               _cs 10                // define screen pins
#define             _mosi 11                // define screen pins
#define             _miso 12                // define screen pins
#define             _sclk 13                // define screen pins

#define            packet 12                // number of bytes in the message
stream
#define            slaves 5                 // number of slave devices in the
system
#define              tMax 265               // maximum temperature system can read
#define              tMin 40                // minimum temperature system can read
```

```
#define         screenInt 1000              // interval between screen prints
#define          transInt 500               // interval between sending data
#define        tempOffset 16                // temperature offset for serial
transmission

Adafruit_ILI9340 tft    = Adafruit_ILI9340(_cs, _dc, _rst);

#define               RED 0xF800
#define              BLUE 0x041F
#define             BLUE2 0x020C
#define             WHITE 0xFFFF
#define             GREY1 0x31A6
#define             GREY2 0x9492
#define             GREY3 0xD69A
#define             BLACK 0x0000
#define             gYmin 40                // top of the graph area
#define             gYmax 170               // bottom of the graph area
#define             gXmin 30                // left of the graph area
#define             gXmax 225               // right of the graph area


// define array of graph line colours
const int line[2 * slaves] = {0x8400, 0x0400, 0x0410, 0xA014, 0x8180,  // setpoints
are darker
                              0xFFE0, 0x07E0, 0x07FF, 0xFA1F, 0xFB00}; //
temperatures are brighter
char* slaveName[slaves] = {"Valve Body", "Valve Heater", "Primary Column",
"Secondary Column", "Transfer Lines"};

int  menu0[] = {170, 220, 250, 200};        // valve target, modulator heater,
modulator time, transfer lines
int  menu1[] = {50, 240, 20};               // primary start, & end, secondary
offset
int  menu2[] = {40, 1, 0};                  // ramp rate, hold time, run time

bool dataIsGood         = LOW;              // flag if checksum is good
bool printOnce          = LOW;              // for printing backgrounds only one
bool lastButton         = LOW;              // flagging last button state
bool buttonPress        = LOW;              // flagging when button is pressed
bool jogTurn            = LOW;              // flagging movement in the jog
dial/button
bool heatersOn          = LOW;              // flag for relay controlling power
bool heatersReady       = LOW;              // flag for heaters at temperature
int dataByte[packet];                      // define array of output bytes
int targetSetpoint[slaves];                // ramping target setpoint
int startSetpoint[slaves];                 // ramping start setpoint
int slaveSetpoint[slaves];                 // array of live slave setpoint
int slaveStatus[slaves];                   // array of slave returned status
int slaveTemp[slaves];                      // array of slave returned temperatures
int lastTemp[slaves];                       // array of last slave temperatures
(for graph only)
int lastSet[slaves];                       // array of last setpoints (for graph
only)
int slavePower[slaves];                    // array of slave power percentages
byte checksum           = 0;                // for caulcuating checksum of
transmission
int  errorCount         = 0;                // for counting errors
int  column             = 0;                // graph column
int  menu               = 0;                // program number
int  jog                = 0;                // multi-use variable when the jog dial
is turned
int  editVal            = 0;                // flag for editting a value
int  lineNo             = 0;                // line number for printing menus
int  overTemp           = 0;

long startTime          = 0;                // start time of GC run
long runTime            = 0;                // calculated run time
long estRunTime         = 0;                // estimated run time from settings
long lastTransmit       = 0;                // time stamp for last transmition
long lastSerial         = 0;                // time since last good serial
communication
long lastScreen         = 0;                // time of last screen print
long lastGraphic        = 0;                // time of last graphic display
long buttonTime         = 0;                // counting long button presses
float rampRate          = 0;                // degs per minute to ramp


// *******************************************************
void setup() {

  Serial.begin(115200);                     // serial comm baud rate
  pinMode(rx, OUTPUT);                      // serial transmit pin
  pinMode(tx, INPUT);                       // serial transmit pin
  pinMode(encA,   INPUT_PULLUP);            // rotary signal A
  pinMode(encB,   INPUT_PULLUP);            // rotary signal B
  pinMode(button, INPUT_PULLUP);            // button pin
  pinMode(shtdwn, INPUT_PULLUP);            // stop button pin
  pinMode(relay,  OUTPUT);                  // relay driving pin
  pinMode(fan,    OUTPUT);                  // fan driving pin

  tft.begin();
  tft.setRotation(2);                       // rotate screen by 180°
  tft.fillScreen(BLACK);                    // print black background
  print_heading(String F("System Startup"), 0);
  delay(250);
  tft.setFont();
  tft.setTextSize(1);
  tft.setCursor(0, 36);

  // check all slaves are present
  int systemReady = 0;                      // system ready flag
  tft.print((char)16); tft.println(F(" Starting DSTL GC concept program"));
  tft.print((char)16); tft.print(F(" Testing serial"));
  while (!dataIsGood){                      // keep transmitting until all slaves
are ready
    prepare_serial();                       // prepare serial data to be sent
    transmit_serial();                      // transmit serial
    tft.print(F("."));                      // print a dot to indicate transmision
    delay(250);                             // short delay
    receive_serial();
  }
  compute_data();                           // read what was returned and look for
errors
  dataIsGood = LOW;                         // clear flag
  tft.println(F("."));                      // one last dot and new line

  for (int sl = 0; sl < slaves; sl++){      // for each slave (slaves are zero
indexed)
    tft.print("   ");
    tft.print(slaveName[sl]);               // print the slave name
    if (slaveStatus[sl] == 5){              // serial not picked up
      tft.println(F(" is missing")); }
    else if (slaveStatus[sl] == 6){         // serial returned good data
      tft.println(F(" is ready")); systemReady++; }
    else {                                  // anything else is an error
      tft.print(F(" returned error "));
      tft.println(slaveStatus[sl]);}
  }
  delay(250);
  tft.print((char)16); tft.print(F(" "));
  tft.print(systemReady); tft.print(F("/"));
  tft.print(slaves); tft.println(F(" ready"));
  tft.print((char)16); tft.println(F(" Testing fan"));
  digitalWrite(fan, HIGH);                  // turn on fan
  delay(1000);
  tft.print((char)16); tft.print(F(" Starting GUI"));
  digitalWrite(fan, LOW);                   // turn off fan
  delay(1000);
  tft.setTextWrap(false);

  menu = 0;                                 // go to first menu
  printOnce = HIGH;                         // set print once flag
  tft.fillScreen(BLACK);                    // print black background
  temp_background();                        // print the overlay
  print_temps();                            // print the current temperatures

  attachInterrupt(digitalPinToInterrupt(encA), encoderA_change, FALLING); //
encoder interupt function
  lastSerial = millis();
}

// *******************************************************
void loop() {

  // ********* if heaters are off
  if (heatersOn) {                                        // if heaters are on
    digitalWrite(relay, HIGH);                            // turn on the relay
  }
  else {                                                  // for heaters off
    clear_setpoints();                                    // clear all setpoints
    digitalWrite(relay, LOW);                             // turn off relay
  }

  // ********* set fan speed
  fan_control();                                          // should the fan be on

  // ********* read input buttons
  read_button();                                          // read digital buttons

  // ********* check for incoming serial data
  receive_serial();                                       // check and read
incomming serial

  if (dataIsGood){                                        // if serial has been
received and is good
    compute_data();                                       // read what was returned
and look for errors
    dataIsGood = LOW;                                     // clear flag
  }

  // ********* send serial data at set interval
  if (millis() - lastTransmit >= transInt){               // at set transmission
intervals
    lastTransmit = millis();                              // update last
transmission
    prepare_serial();                                     // prepare serial data to
be sent
    transmit_serial();                                    // transmit serial
  }

  // ********* print live temperatures
  if (millis() - lastScreen >= screenInt){                // at set screen
intervals
    lastScreen = millis();                                // update screen interval
    print_temps();                                        // print bottom
temperatures
  }

  // ********* check for errors
  if (millis() - lastSerial > 5000 || errorCount > 10){    // if no serial for more
than 5 seconds
    if (menu != 99){
      menu = 99;                                          // move to error program
      printOnce = HIGH;                                   // for printing new
screen
    }
  }

  errorCount--;                                           // reduce error count
  errorCount = constrain(errorCount, 0, 128);             // constrain error count

  switch (menu) {
    // ********************** MENU 0 ***********************
    case 0:
    // ******************** PRINT ONCE (0)******************
      if (printOnce){                                       // print this once
        lineNo = 50;                                        // set row starting
point
        print_heading(F("Preheat & Modulation"), 1);        // clear background and
print heading line
        print_line(F("Valve Body"),  menu0[0], 1, "C", WHITE);
        print_line(F("Valve Heater"), menu0[1], 1, "C", WHITE);
        print_line(F("Modulator Pulse"),  menu0[2], 0, "ms", WHITE);
        print_line(F("Transfer Lines"),   menu0[3], 1, "C", WHITE);
        print_button(170, lineNo, 70, F("NEXT"), 1);
        tft.fillTriangle(230, lineNo-2, 230, lineNo-10, 234, lineNo-6, WHITE);
        heatersOn = LOW;                                    // make sure heaters
are off
        printOnce = LOW;                                    // clear print once
        jog = 0;                                            // set pointer to first
line
        editVal = -1;                                       // not editing a line
        jogTurn = HIGH;                                     // force jog turn high
      }
    // ************** SCROLLING THROUGH MENU (0)************
      if (editVal < 0){                                     // if NOT editing a
line
        if (jogTurn){                                       // if dial has been
turned
          jog = constrain(jog, 0, 4);                       // limit jog to number
of options on a menu
          lineNo = jog * 25 + 50;                           // calculate line
number
          tft.fillRect(5, 40, 5, 130, BLACK);               // clear old triangles
          tft.fillTriangle(5, lineNo-2, 5, lineNo-10, 9, lineNo-6, WHITE); // draw
a pointer
          jogTurn = LOW;                                    // clear the change
flag
        }
        if (buttonPress){                                   // if the button has
been pressed
          editVal = jog;                                    // move to edit
setpoint
          tft.drawRoundRect(170, lineNo-16, 70, 21, 4, RED);// draw red box
          if (editVal >= 4){                                // if moving on to next
menu
            menu++;                                         // move to next menu
            printOnce = HIGH;                               // set printonce
          }
          else { jog = menu0[editVal]; }                    // move menu setting to
jog
          buttonPress = LOW;                                // clear button press
          jogTurn = HIGH;                                   // force setpoint to be
re-written
        }
      }
    // **************** EDITING  VALUES (0)*****************
      else {                                                // so if edit val is
greater than 0
        lineNo = editVal * 25 + 50;                         // calculate line
number
        if (jogTurn){                                       // if dial has been
turned
          if (editVal == 2) { jog = constrain(jog, 50, 750); } // limit pulse width
          else { jog = constrain(jog, tMin, tMax - 15); }   // limit jog to max/min
temps
          print_value(lineNo, jog, WHITE);                  // update the value
bine edited
          jogTurn = LOW;                                    // clear the change
flag
        }
        if (buttonPress){                                   // if the button has
been pressed
```

```
          menu0[editVal] = jog;                             // update value from
jog

          tft.drawRoundRect(170, lineNo-16, 70, 21, 4, BLACK);  // draw black box
          jog = editVal;
          editVal = -1;                                     // clear editVal
          buttonPress = LOW;                                // clear button press
        }
      }
    break;

    // ********************** MENU 1 **********************
    case 1:
    // ******************** PRINT ONCE (1)*****************
      if (printOnce){                                       // print this once
        lineNo = 50;                                        // set row starting
point
        print_heading(F("Column Temperatures"), 1);         // clear background and
print heading line
        print_line(F("Primary Initial"),  menu1[0], 1, "C", WHITE);
        print_line(F("Primary Final"),    menu1[1], 1, "C", WHITE);
        print_line(F("Secondary Offset"), menu1[2], 1, "C", WHITE);
        print_button(170, lineNo, 70, F("  BACK"), 1);
        tft.fillTriangle(180, lineNo-2, 180, lineNo-10, 176, lineNo-6, WHITE);
        lineNo += 25;
        print_button(170, lineNo, 70, F("NEXT"), 1);
        tft.fillTriangle(229, lineNo-2, 229, lineNo-10, 233, lineNo-6, WHITE);
        heatersOn = LOW;                                    // make sure heaters
are off
        printOnce = LOW;                                    // clear print once
        jog = 0;                                            // set pointer to first
line
        editVal = -1;                                       // not editing a line
        jogTurn = HIGH;                                     // force jog turn high
      }
    // ************** SCROLLING THROUGH MENU (1)************
      if (editVal < 0){                                     // if NOT editing a
line
        if (jogTurn){                                       // if dial has been
turned
          jog = constrain(jog, 0, 4);                       // limit jog to number
of options on a menu
          lineNo = jog * 25 + 50;                           // calculate line
number
          tft.fillRect(5, 40, 5, 130, BLACK);               // clear old triangles
          tft.fillTriangle(5, lineNo-2, 5, lineNo-10, 9, lineNo-6, WHITE); // draw
a pointer
          jogTurn = LOW;                                    // clear the change
flag
        }
        if (buttonPress){                                   // if the button has
been pressed
          editVal = jog;                                    // move to edit
setpoint
          tft.drawRoundRect(170, lineNo-16, 70, 21, 4, RED);// draw red box
          if (editVal == 3){                                // if moving on to next
menu
```

```
            menu--;                                          // move back a menu
            printOnce = HIGH;                                // set printonce
          }
          else if (editVal >= 4){
            menu++;                                          // move to next menu
            printOnce = HIGH;                                // set printonce
          }
          else { jog = menu1[editVal]; }                     // move menu setting to
jog
          buttonPress = LOW;                                 // clear button press
          jogTurn = HIGH;                                    // force setpoint to be
re-written
        }
      }
    // **************** EDITING  VALUES (1)******************
      else {                                                 // so if edit val is
greater than 0
        lineNo = editVal * 25 + 50;                          // calculate line
number
        if (jogTurn){                                        // if dial has been
turned
          if (editVal == 2){ jog = constrain(jog, 0, 50); } // limit jog to column
offset
          else { jog = constrain(jog, tMin, tMax-15); }      // limit jog to max/min
temps
          print_value(lineNo, jog, WHITE);                   // update the value
being edited
          jogTurn = LOW;                                     // clear the change
flag
        }
        if (buttonPress){                                    // if the button has
been pressed
          menu1[editVal] = jog;                              // update value from
jog
          tft.drawRoundRect(170, lineNo-16, 70, 21, 4, BLACK);  // draw black box
          jog = editVal;
          editVal = -1;                                      // clear editVal
          buttonPress = LOW;                                 // clear button press
        }
      }
    break;

    // ********************** MENU 2 ***********************
    case 2:
    // ******************** PRINT ONCE (2)******************
      if (printOnce){                                        // print this once
        lineNo = 50;                                         // set row starting
point
        menu2[2] = menu2[1] + (menu1[1] - menu1[0]) / menu2[0];
        print_heading(F("Ramp Settings"), 1);                // clear background and
print heading line
        print_line(F("Ramp Rate"),     menu2[0], 1, "/m", WHITE);
        print_line(F("Hold Time"),     menu2[1], 0, "min", WHITE);
        print_line(F("Est. Runtime"),  menu2[2], 0, "min", GREY2);

        print_button(170, lineNo, 70, F("  BACK"), 1);
        tft.fillTriangle(180, lineNo-2, 180, lineNo-10, 176, lineNo-6, WHITE);
        lineNo += 25;
        print_button(170, lineNo, 70, F("NEXT"), 1);
        tft.fillTriangle(229, lineNo-2, 229, lineNo-10, 233, lineNo-6, WHITE);
        heatersOn = LOW;                                    // make sure heaters
are off
        printOnce = LOW;                                    // clear print once
        jog = 0;                                            // set pointer to first
line
        editVal = -1;                                       // not editing a line
        jogTurn = HIGH;                                     // force jog turn high
      }
    // ************** SCROLLING THROUGH MENU (2)************
      if (editVal < 0){                                     // if NOT editing a
line
        if (jogTurn){                                       // if dial has been
turned
          jog = constrain(jog, 0, 4);                       // limit jog to number
of options on a menu
          lineNo = jog * 25 + 50;                           // calculate line
number
          tft.fillRect(5, 40, 5, 130, BLACK);               // clear old triangles
          tft.fillTriangle(5, lineNo-2, 5, lineNo-10, 9, lineNo-6, WHITE); // draw
a pointer
          jogTurn = LOW;                                    // clear the change
flag
        }
        if (buttonPress && (jog != 2)){                     // if the button has
been pressed (except option 2)
          editVal = jog;                                    // move to edit
setpoint
          tft.drawRoundRect(170, lineNo-16, 70, 21, 4, RED);// draw red box
          if (editVal == 3){                                // if moving on to next
menu
            menu--;                                         // move back a menu
            printOnce = HIGH;                               // set printonce
          }
          else if (editVal >= 4){
            menu++;                                         // move to next menu
            printOnce = HIGH;                               // set printonce
          }
          else { jog = menu2[editVal]; }                    // move menu setting to
jog
          buttonPress = LOW;                                // clear button press
          jogTurn = HIGH;                                   // force setpoint to be
re-written
        }
        else if (buttonPress){                              // so in when selecting
an uniditable item
          jogTurn = HIGH;                                   // fake a jog turn
          jog--;                                            // move arrow to
previous item
          buttonPress = LOW;                                // clear button press
        }
      }
    // **************** EDITING  VALUES (2)****************
      else {                                                // so if edit val is
greater than 0
```

```
        lineNo = editVal * 25 + 50;                         // calculate line
number
        if (jogTurn){                                       // if dial has been
turned
          jog = constrain(jog, 1, 60);                      // limit jog to ramp
rate and hold time
          print_value(lineNo, jog, WHITE);                  // update the value
being edited
          jogTurn = LOW;                                    // clear the change
flag
        }
        if (buttonPress){                                   // if the button has
been pressed
          menu2[editVal] = jog;                             // save the value from
jog
          menu2[2] = menu2[1] + (menu1[1] - menu1[0]) / menu2[0]; // calc new
runtime
          if (menu2[2] >= 60){                              // for long or no
runtime
            print_value(100, menu2[2], RED);                // print the runtime in
RED!
            }
          else {
            print_value(100, menu2[2], GREY2);              // print the run time
          }

          tft.drawRoundRect(170, lineNo-16, 70, 21, 4, BLACK);  // draw black box
over red edit box
          jog = editVal;
          editVal = -1;                                     // clear editVal
          buttonPress = LOW;                                // clear button press
        }
      }
    break;

    // ********************** MENU 3 ***********************
    case 3:
    // ******************* PRINT ONCE (3)*****************
      if (printOnce){                                       // print this once
        lineNo = 50;                                        // set row starting
point
        print_heading(F("Preheating - NOT READY"), 1);      // clear background and
print heading line
        print_button(20, lineNo, 70, F("  BACK"), 1);
        tft.fillTriangle(29, lineNo-2, 29, lineNo-10, 25, lineNo-6, WHITE);
        print_button(170, lineNo, 70, F("START"), 0);
        tft.fillTriangle(5, 48, 5, 40, 9, 44, WHITE);       // draw a pointer

        // move menu setting to setpoints
        slaveSetpoint[0] = menu0[0];                        // injector setpoint
        slaveSetpoint[1] = menu0[1];                        // modulator setpoint
        slaveSetpoint[2] = menu1[0];                        // primary column
        slaveSetpoint[3] = menu1[0] + menu1[2];             // secondary column
(calc from offet)
        slaveSetpoint[4] = menu0[3];                        // transfer lines (not
sure how this will work yet)
```

```
        targetSetpoint[2] = menu1[1];                       // primary column ramp
target
        targetSetpoint[3] = menu1[1] + menu1[2];            // secondary column
(calc from offet)
        startSetpoint[2] = slaveSetpoint[2];                // note the start
setpoint of primary
        startSetpoint[3] = slaveSetpoint[3];                // note the start
setpoint of secondary

        thermometer_background();                           // print the
thermometer background

        // turn on power relay   <<<<<<
        heatersOn = HIGH;                                   // set heaters on flag
        printOnce = LOW;                                    // clear print once
        jog = 0;                                            // set pointer to first
line
        editVal = -1;                                       // not editing a line
        jogTurn = HIGH;                                     // force jog turn high

        heatersReady = HIGH;              // <<<<<<<<< for testing only

      }

      if (millis() - lastGraphic >= screenInt){             // at set intervals
        lastGraphic = millis();                             // update last graph
print
        print_thermometer();                                // print the live
thermometer
      }

    // ************** SCROLLING THROUGH MENU (3)************
      if (buttonPress){                                     // if the button has
been pressed
        tft.drawRoundRect(20, 34, 70, 21, 4, RED);          // draw red box
        menu--;                                             // move back a menu
        printOnce = HIGH;                                   // set printOnce flag
        buttonPress = LOW;                                  // clear button press
      }

     // check setpoints are at temperature, if so set heatersReady flag
      if (heatersReady){
        menu++;
        printOnce = HIGH;
      }
    break;

    // ********************** MENU 4 ***********************
    case 4:
    // ******************** PRINT ONCE (4)******************
      if (printOnce){                                       // print this once
        print_heading(F("Heaters are READY"), 0);           // clear background and
print heading line
        print_button(170, lineNo, 70, F("START"), 1);
        printOnce = LOW;
      }
      // ************** SCROLLING THROUGH MENU (4)***********
      if (jogTurn){                                         // if dial has been
turned
        jog = constrain(jog, 0, 1);                         // limit jog to number
of options on a menu
        int col = 5 + (jog * 150);                          // arrow moves by
column
        tft.fillRect(5, 40, 5, 9, BLACK);                   // clear old triangles
        tft.fillRect(155, 40, 5, 9, BLACK);                 // clear old triangles
        tft.fillTriangle(col, 48, col, 40, col+4, 44, WHITE); // draw a pointer
        jogTurn = LOW;                                      // clear the change
flag
      }
      if (buttonPress){                                     // if the button has
been pressed (except option 2)
        editVal = jog;                                      // move to edit
setpoint
        if (editVal == 0){                                  // if moving on to next
menu
          tft.drawRoundRect(20, 34, 70, 21, 4, RED);        // draw blue box
          menu-=2;                                          // move back 2 menus
          printOnce = HIGH;                                 // set printOnce flag
        }
        else if (editVal >= 1){
          tft.drawRoundRect(170, lineNo-16, 70, 21, 4, RED); // draw RED box
          menu++;                                           // move to next menu
          printOnce = HIGH;                                 // set printonce
        }
        buttonPress = LOW;                                  // clear button press
      }

      if (millis() - lastGraphic >= screenInt){             // at set intervals
        lastGraphic = millis();                             // update last graph
print
        print_thermometer();                                // print the live
thermometer
      }
    break;

    // ********************** MENU 5 ***********************
    case 5:
    // ******************* PRINT ONCE (5)*****************
      if (printOnce){                                       // print this once
        lineNo = 50;                                        // set row starting
point
        print_heading(F("Temperature Ramp"), 1);            // clear background and
print heading line
        rampRate = 60000 / menu2[0];                        // calculate ramp rate
        estRunTime = (60000 * menu2[1]) + (rampRate * (menu1[1] - menu1[0])); //
calc run time
        graph_background();                                 // print the graph
background
        heatersOn = HIGH;                                   // check that heaters
on is enabled
        printOnce = LOW;                                    // clear print once
        column = gXmin + 1;                                 // set first column
        startTime = millis();                               // note the start time
      }
      // update setpoints
      runTime = millis() - startTime;                       // convert to time
since start
      for (int sl = 2; sl <=3 ; sl++){                      // for each of the two
heaters
        if (slaveSetpoint[sl] >= targetSetpoint[sl]) {      // at the top of the
ramp
          slaveSetpoint[sl] = targetSetpoint[sl];           // hold at temp....
        }
        else{                                               // else keep ramping
          slaveSetpoint[sl] = startSetpoint[sl] + (runTime / rampRate);
        }
      }
      if (runTime >= estRunTime){                           // if run time has
reached the end
        print_graph();                                      // print the graph
        menu++;                                             // move to next stage
        printOnce = HIGH;                                   // print new screen
      }
      if (millis() - buttonTime >= 3000 && buttonPress){
        print_graph();                                      // print the graph
        menu++;                                             // move to next stage
        printOnce = HIGH;                                   // print new screen
        buttonPress = LOW;                                  // clear button press
before next screen
      }
      if (millis() - lastGraphic >= screenInt){             // at set intervals
        lastGraphic = millis();                             // update last graph
print time
        print_graph();                                      // print the graph
        print_serial();
      }
    break;

    // ********************** MENU 6 ***********************
    case 6:
    // ******************* PRINT ONCE (6)*****************
      if (printOnce){                                       // print this once
        print_heading(F("Ramp Complete"), 0);               // clear background and
print heading line
        print_button(75, 75, 90, F("  Repeat"), 1);         // to repeat with same
settings
        print_button(75, 100, 90, F(" Change"), 1);         // to change the
settings
        print_button(75, 125, 90, F("Shutdown"), 1);        // to shut down the
system
        // move menu setting to setpoints
        slaveSetpoint[2] = menu1[0];                        // primary column
        slaveSetpoint[3] = menu1[0] + menu1[2];             // secondary column
(calc from offet)
        heatersOn = HIGH;                                   // heaters should still
be on
        digitalWrite(fan, HIGH);                            // turn on fan
        tft.fillTriangle(64, 73, 64, 65, 68, 69, WHITE);    // draw a pointer
        jog = 0;                                            // set jog to 0
        lineNo = 75;                                        // preset line number
        printOnce = LOW;                                    // clear print once
      }

      // ************** SCROLLING THROUGH MENU (6)************
      if (jogTurn){                                         // if dial has been
turned
        jog = constrain(jog, 0, 2);                         // limit jog to number
of options on a menu
        lineNo = jog * 25 + 75;                             // calculate line
number
        tft.fillRect(64, 65, 5, 60, BLACK);                 // clear old triangles
        tft.fillTriangle(64, lineNo-2, 64, lineNo-10, 68, lineNo-6, WHITE); // draw
a pointer
        jogTurn = LOW;                                      // clear the change
flag
      }
      if (buttonPress){                                     // if the button has
been pressed
        tft.drawRoundRect(75, lineNo-16, 90, 21, 4, RED);   // draw red box
        if (jog == 0){        menu = 3; }                   // menu 3 for same
settings
        else if (jog == 1){   menu = 0; }                   // menu 0 to change
settings
        else {                menu++; }                     // move on to shutdown
        printOnce = HIGH;                                   // set printonce
        buttonPress = LOW;                                  // clear button press
      }

    break;

    // ******************* MENU "default" ******************
    default:
    // ******************** PRINT ONCE (d)******************
      if (printOnce){                                       // print this once
        print_heading(F("Shutdown"), 1);                    // clear background and
print heading line
        tft.setCursor(4, 50);                               // set position
        tft.print("System is cooling");                     // print line
        clear_setpoints();                                  // set all setpoints
low
        thermometer_background();                           // print the
thermometer background
        digitalWrite(fan, HIGH);                            // turn on fan
        printOnce = LOW;                                    // clear print once
      }
      if (millis() - buttonTime >= 3000 && buttonPress){    // to escape from
shutdown
        menu = 0;                                           // move to menu 0
        printOnce = HIGH;                                   // print new screen
        buttonPress = LOW;                                  // clear button press
before next screen
      }
      if (millis() - lastGraphic >= screenInt){             // at set intervals
        lastGraphic = millis();                             // update last graph
print
        print_thermometer();                                // print the live
thermometer
```

```
      }
    break;
  }
  // ******************** END OF MENU *********************

}

// *******************************************************
void receive_serial() {
  if (Serial.available() >= packet){                      // if there is enough
data int the buffer
    delay(1);                                             // wait in case of extra
bytes
    checksum = 0;                                         // clear the checksum
    dataIsGood = LOW;                                     // clear data good flag
    for (int i = 0; i < packet-1; i++){                   // loop through each byte
      dataByte[i] = Serial.read();                        // store to array of
inputs
      checksum = checksum ^ dataByte[i];                  // calculate checksum on
the fly
    }
    dataByte[packet-1] = Serial.read();                   // read in the last byte
    // compare checksums && check first byte is a header
    if (dataByte[packet-1] == checksum && dataByte[0] == 1){
      dataIsGood = HIGH;                                  // flag it as good data
to process
      lastSerial = millis();                              // log the last time good
data was received
    }

    while (Serial.available() > 0){                       // if there is still data
in the buffer
      char trash = Serial.read();                         // read it to trash
    }
  }
}

// *******************************************************
void compute_data() {
  for (int sl = 0; sl < slaves; sl++){                    // for each slave (slaves
are zero indexed)
    slaveStatus[sl] = dataByte[2 * sl + 1];               // move slave status from
serial to local variable
    if (slaveStatus[sl] == 6){                            // slave is present but
heating is off
      slaveTemp[sl] = dataByte[2 * sl + 2] + tempOffset;  // then update the live
temperature (note temperature offset)
      slavePower[sl] = 0;                                 // note that slave is off
    }
    else if (slaveStatus[sl] == 5){                       // no slave[sl] found
      slaveTemp[sl] = 0;                                  // slave was not found
and so set nul valve
    }
    else if (slaveStatus[sl] >= 128 && slaveStatus[sl] <= 228){ // slave returned a
power factor
      slaveTemp[sl] = dataByte[2 * sl + 2] + tempOffset;  // update the live
temperature (note temperature offset)
```

```
      slavePower[sl] = slaveStatus[sl] - 128;             // update the slaves
power percentage
    }
    else {                                                // if slave status
returned a nul value
      slaveTemp[sl] = 0;                                  // slave was not found
and so set nul valve
      errorCount++;                                       // increment error count
    }
  }
}

// *******************************************************
void prepare_serial() {
  dataByte[0] = 1;                                        // set byte zero to be a
header
  for (int sl = 0; sl < slaves; sl++){                    // for each slave
    int b = 2 + (2 * sl);                                 // calculate the serial
byte number
    dataByte[b - 1] = 5;                                  // set status
    if (heatersOn){ dataByte[b] = slaveSetpoint[sl] - tempOffset; }    // if the
heaters are on transmit the setpoint
    else { dataByte[b] = tempOffset; }                    // otherwise transmit
zero (note offset!)
  }
  //dataByte[11] is the checksum byte
}

// *******************************************************
void transmit_serial() {
  checksum = 0;                                           // clear the checksum
  for (int i = 0; i < packet -1; i++){                    // for header and all
data bytes
    Serial.write(dataByte[i]);                            // send serial data
    checksum = checksum ^ dataByte[i];                    // calculate checksum on
the fly
  }
  dataByte[packet-1] = checksum;                          // last byte is the
checksum
  Serial.write(dataByte[packet-1]);                       // send the last byte
}

// *******************************************************
void print_serial() {
  Serial.println();                                       // start new line
  Serial.print(runTime);                                  // print time
  for (int sl = 2; sl <= 3; sl++){                        // for each slave
    Serial.print("\t"); Serial.print(slaveTemp[sl]);      // print temperature
  }
  Serial.println();                                       // start new line
}

// *******************************************************
void temp_background() {                                  // print the temperature
background
  tft.setFont();
  tft.setTextSize(1);
  for (int sl = 0; sl < slaves; sl++){                    // for each slave
    int row = 27 * sl + 186;                              // calculate a TFT line
position
    // print slave name
    tft.setTextColor(WHITE, BLACK);
    tft.drawLine(0, row, 239, row, GREY1);                // draw grey line
    tft.setCursor(4, row + 4);                            // set first line
    tft.print(slaveName[sl]);                             // print slave name
    //print "setpoint" and "temperature"
    tft.drawLine(115, row + 7, 147, row + 7, line[sl]);   // draw setpoint line
    tft.drawLine(115, row + 19, 129, row + 19, line[sl+5]); // draw temperature
line
    tft.setCursor(152, row + 4);                          // top right
    tft.setTextColor(GREY3, BLACK);                       // change to grey on
black
    tft.print(F("Setpoint"));                             // print "setpoint"
    tft.setCursor(134, row + 16);                         // bottom right
    tft.print(F("Temperature"));                          // print "Temperature"
  }
}

// *****************************************************
void print_temps() {
  tft.setFont();                                          // set basic font
  tft.setTextSize(1);                                     // make it small
  /*
  debug printing
  tft.setTextColor(WHITE, BLACK);
  tft.setCursor(4, 165);
  tft.print(errorCount);
  tft.print("  ");
  */
  for (int sl = 0; sl < slaves; sl++){                    // for each slave
    int row = 27 * sl + 186;                              // calculate a row offset
    tft.setCursor(4, row + 16);                           // move position to below
slave name
    if (slaveStatus[sl] == 6){                            // 6 acknowledges receipt
only
      tft.setTextColor(GREY2, BLACK);                     // grey on black for
heaters off
      tft.print(F("Power Off  ")); }                      // print "heaters off"
    else if (slaveStatus[sl] >= 128 && slaveStatus[sl] <= 228){
      tft.setTextColor(BLUE, BLACK);                      // blue on black for
power %
      tft.print(F("Power ")); tft.print(slavePower[sl]); tft.print(F("% ")); }
    else if (slaveStatus[sl] == 5){                       // when serial is
returned un-seen
      tft.setTextColor(RED, BLACK);                       // red on black for no
slave
      tft.print(F("Offline   ")); }
    else {
      tft.setTextColor(RED, BLACK);                       // red on black for error
      tft.print(F("Error ")); tft.print(slaveStatus[sl]); tft.print(F("  ")); }
    //print slave setpoint and temperature
    tft.setTextColor(WHITE, BLACK);                       // white on black for
numbers
    tft.setCursor(206, row + 4);                          // move to top right
    if (!heatersOn){                                      // if the heaters are off
      tft.print(F("Off    "));  }                         // print off
    else {                                                // else print the temp
      tft.print(slaveSetpoint[sl]); tft.print((char)247); tft.print("C  "); }
    tft.setCursor(206, row + 16);                         // move to bottom right
    tft.print(slaveTemp[sl]); tft.print((char)247); tft.print("C  ");
    read_button();                                        // read button mid print
    if (buttonPress || jogTurn){                          // and exit print is
something has changed
      break;
    }
  }
}

// *******************************************************
void thermometer_background() {                           // print the temperature
background
  tft.setFont();                                          // set basic font
  tft.setTextSize(1);                                     // make it small
  for (int sl = 0; sl < slaves; sl++){                    // for each slave
    int col = 16 + sl*47;                                 // the column ID for each
thermometer
    int sp  = 67 + map(slaveSetpoint[sl], 0, 255, 98, 0); // calculate setpoint
level
    tft.setTextColor(line[sl]);                           // setpoint colour
    if (slaveSetpoint[sl]){                               // if the setpoint is
greater than 0
      tft.setCursor(col + 12, sp - 3);                    // set cursor setpoint
      tft.print(slaveSetpoint[sl]);                       // print the setpoint
      tft.drawLine(col-2, sp, col+8, sp, line[sl]);       // draw a setpoint line
    }
    tft.drawRoundRect(col, 66, 7, 100, 3, WHITE);         // draw thermometer
    tft.drawCircle(col+3, 170, 7, WHITE);                 // draw bulb
    tft.fillCircle(col+3, 170, 6, BLACK);                 // clear bottom of rod
    tft.fillRect(col+1, 162, 5, 3, BLACK);                // clear top of bulb
    tft.fillCircle(col+3, 170, 5, line[sl+5]);            // draw liquid in bulb
    tft.setRotation(1);                                   // turn the screen
    tft.setCursor(158, col - 11);                         // cursor to side of
thermometer
    tft.print(slaveName[sl]);                             // print the name of the
heaters
    tft.setRotation(2);                                   // return screen rotation
  }
}
// *******************************************************
void print_thermometer() {                                // print the temperature
background
  tft.setFont();                                          // set basic font
  tft.setTextSize(1);                                     // make it small
  for (int sl = 0; sl < slaves; sl++){                    // for each slave
    int col = 16 + sl*47;                                 // the column ID for each
thermometer
    int top = 67 + map(slaveTemp[sl], 0, 255, 98, 0);     // calculate top of level
    int bot = 67 + 98 - top;                              // bottom of thermometer
    int sp  = 67 + map(slaveSetpoint[sl], 0, 255, 98, 0); // calculate setpoint
level
    tft.fillRect(col+2, 67, 3, top-67, BLACK);            // clear thermometer
    if (slaveSetpoint[sl]){
      // if the setpoint is greater than 0
      tft.drawLine(col+1, sp, col+5, sp, line[sl]);       // draw a setpoint line
    }
    tft.fillRect(col+2, top, 3, bot, line[sl+5]);         // draw level
    tft.setTextColor(line[sl+5], BLACK);                  // print the live
temperature
    tft.setCursor(col + 14, 167);                         // move cursor
    tft.print(slaveTemp[sl]);                             // print live temperature
    if (slaveTemp[sl] < 100){ tft.print(" "); }           // print blank spaces
    if (slaveTemp[sl] <  10){ tft.print(" "); }           // print blank spaces
  }
}

// *******************************************************
void graph_background() {
  tft.drawLine(gXmin-1, gYmin-2, gXmin-1, gYmax+1, GREY2);// draw Y Axis
  tft.drawLine(gXmin-1, gYmax+1, gXmax+2, gYmax+1, GREY2);// draw X Axis
  tft.setFont();                                          // set basic font
  tft.setTextSize(1);                                     // make it small
  int spMax = targetSetpoint[3] + 10;                     // note the max
temperature
  for (int sl = 2; sl <= 3; sl++){                        // for each slave that we
want to draw
    tft.setTextColor(line[sl]);                           // print the setpoint
    int mark = map(slaveSetpoint[sl], tMin-10, spMax, gYmax, gYmin);  // convert
setpoint to position
    tft.drawLine(gXmin-5, mark, gXmin-2, mark, line[sl]); // draw low setpoint mark
    tft.setCursor(4, mark + 17 - (8 * sl));               // set cursor abover or
below mark
    if (slaveSetpoint[sl] < 100){ tft.print(" "); }       // print leading space
    tft.print(slaveSetpoint[sl]);                         // print start setpoint
    mark = map(targetSetpoint[sl], tMin-10, spMax, gYmax, gYmin); // convert target
setpoint to position
    tft.drawLine(gXmin-5, mark, gXmin-2, mark, line[sl]); // traw target setpoint
    tft.setCursor(4, mark + 17 - (8 * sl));               // set cursor abover or
below mark
    if (targetSetpoint[sl] < 100){ tft.print(" "); }      // print leading space
    tft.print(targetSetpoint[sl]);                        // print end setpoint
    lastSet[sl] = slaveSetpoint[sl];                      // pre-fill last setpoint
    lastTemp[sl] = slaveTemp[sl];                         // pre-fill last temp
  }
  print_time(210, gYmax+4, estRunTime, GREY2, 0);
}

// *******************************************************
void print_graph() {
  column++;

  if (column > gXmax){                                    // at end of graph
    column = gXmin + 1;                                   // reset column number
(+1)
    tft.drawLine(gXmax + 1, gYmin-2, gXmax+1, gYmax, BLACK); // clear end line
    tft.drawLine(gXmin, gYmin, gXmin, gYmax, BLACK);      // clear first line
before printing graph
  }
  tft.drawLine(column, gYmin-2, column, gYmax, BLACK);    // clear colum
  tft.drawLine(column + 1, gYmin, column + 1, gYmax, GREY1); // grey column before
lines
  int spMax = targetSetpoint[3] + 10;                     // note the max
temperature
  for (int sl = 2; sl <= 3; sl++){                        // for each slave that we
want to draw
    int yNew = map(slaveSetpoint[sl], tMin-10, spMax, gYmax, gYmin); // scale
setpoint
    yNew = constrain(yNew, gYmin, gYmax);
    int yOld = map(lastSet[sl], tMin-10, spMax, gYmax, gYmin); // scale last
setpoint
    yOld = constrain(yOld, gYmin, gYmax);
    tft.drawLine(column-1, yOld, column, yNew, line[sl]); // draw setpoint point
    lastSet[sl] = slaveSetpoint[sl];                      // copy setpoint to last
setpoint
  }
  for (int sl = 2; sl <= 3; sl++){                        // for each slave
    int yNew = map(slaveTemp[sl], tMin-10, spMax, gYmax, gYmin); // scale temp
    yNew = constrain(yNew, gYmin, gYmax);
    int yOld = map(lastTemp[sl], tMin-10, spMax, gYmax, gYmin);  // scale last temp
    yOld = constrain(yOld, gYmin, gYmax);
    tft.drawLine(column-1, yOld, column, yNew, line[sl+5]); // draw temp line
    lastTemp[sl] = slaveTemp[sl];                         // copy current temp to
last temp
  }
  print_time(175, gYmax+4, runTime, GREY3, 0);
}

// *******************************************************
void print_time(int x, int y, long t, int c, bool b){     // for converting and
printing a time
  tft.setFont();                                          // set basic font
  tft.setTextSize(1);                                     // make it small
  tft.setTextColor(c, BLACK);                             // colour on black
  tft.setCursor(x, y);                                    // set cursor below time
axis
  if (b) { tft.print("("); }                              // print bracket
  int minutes = t / 60000 ;                               // calc minutes
  int seconds = (t % 60000)/ 1000;                        // calc seconds
  tft.print(minutes);                                     // print minutes
  tft.print(":");                                         // print colon
  if (seconds < 10) {tft.print("0"); }                    // print leading zero
  tft.print(seconds);                                     // print seconds
  if (b) { tft.print(")"); }                              // print bracket


}

// *******************************************************
void print_heading(String heading, bool blank){           // clear the top screen
  if (blank) { tft.fillRect(0, 0, 240, 185, BLACK); }     // draw rectangle over
top screen
  tft.fillRect(0, 0, 240, 25, BLUE2);                     // dark blue heading
background
  tft.setFont(&FreeSans9pt7b);                            // set font
  tft.setTextColor(WHITE);                                // set text colour
  tft.setCursor(4, 17);                                   // set position
  tft.print(heading);                                     // print heading
  tft.drawLine(0, 25, 239, 25, WHITE);                    // draw grey line under
heading
}

// *******************************************************
void print_line(String text, int val, int degs, String unit, int colour){
  tft.setTextColor(colour);                               // change text colour
  tft.setCursor(15, lineNo);                              // set cursor for each
line
  tft.print(text);                                        // print the name
  tft.setCursor(175, lineNo);                             // set cursor for
setpoint
  if (val < 10){ tft.print(F("  ")); }                    // add a spaces to shift
low numbers
  if (val < 100){ tft.print(F("  ")); }                   // add a spaces to shift
low numbers
  tft.print(val);                                         // print the current
setpoint
  if (degs){ tft.setCursor(215, lineNo);                  // if degrees move next
unit
    tft.drawCircle(210, lineNo-10, 2, colour);}           // draw ° symbol
  else { tft.setCursor(206, lineNo);}                     // not degrees, unit
straight after value
  tft.print(unit);                                        // print the current
setpoint
  lineNo += 25;
}

// *******************************************************
void print_value(int y, int val, int c){                  // setting large font for
menus
  tft.fillRect(175, y-12, 30, 13, BLACK);                 // clear old value
  tft.setFont(&FreeSans9pt7b);                            // set font
  tft.setTextColor(c);                                    // set text colour
  tft.setCursor(175, y);                                  // set cursor last (it
works better that way!)
  if (val < 10){ tft.print(F("  ")); }                    // add a spaces to shift
low numbers
  if (val < 100){ tft.print(F("  ")); }                   // add a spaces to shift
low numbers
  tft.print(val);                                         // print the current
setpoint
}

// *******************************************************
void print_button(int x, int y, int w, String but, int c){        // drawing buttons
  tft.setTextColor(WHITE);                                // change text colour
  if (c) {                                                // if C, blue buttons
    tft.fillRoundRect(x, y - 16, w, 21, 4, BLUE2);        // fill blue box
    tft.drawRoundRect(x, y - 16, w, 21, 4, BLUE);         // draw blue box
  }
  else {                                                  // if not C, grey
    tft.fillRoundRect(170, y - 16, w, 21, 4, GREY1);      // fill grey box
    tft.drawRoundRect(170, y - 16, w, 21, 4, GREY2);      // draw draw box
  }
  tft.setCursor(x + 6, y); tft.print(but);                // print button option
```

```
}

// *****************************************************
void encoderA_change(){
  cli();                                            // stop interupts
  delayMicroseconds(10);                            // delay helps, don't
know why
  bool encoderA = digitalRead(encA);                // read encoderA
  bool encoderB = digitalRead(encB);                // read encoderA
  delayMicroseconds(500);                           // delay before moving on
  if (!encoderA && encoderB){                       // A low, B high
    jog--;                                          // CCW
    jogTurn = HIGH;                                 // flag that there has
been a change
  }
  else if(!encoderA && !encoderB){                  // A low, B low
    jog++;                                          // CW
    jogTurn = HIGH;                                 // flag that there has
been a change
  }
  sei();                                            // resume interupts
}

// *****************************************************
void fan_control(){
  overTemp = 0;                                     // for cumulative
temperature measurement
  int fanSP = 0;                                    // fan setpoint
  for (int sl = 2; sl <= 3; sl++) {                 // for each column
    if (heatersOn){ fanSP = slaveSetpoint[sl]; }
    else          { fanSP = 40; }

    if (slaveTemp[sl] == 0) { overTemp += 100; }    // if there is a heater
error
    else if (slaveTemp[sl] > fanSP){                // if temp above setpoint
      overTemp += slaveTemp[sl] - fanSP;            // add the difference
    }
  }
  if (overTemp > 40){ digitalWrite(fan, HIGH); }    // turn on fan
  if (overTemp < 5){ digitalWrite(fan, LOW); }      // turn on fan
}

// *****************************************************
void clear_setpoints(){
  for (int sl = 0; sl < slaves; sl++){              // for each slave (slaves
are zero indexed)
    slaveSetpoint[sl] = 0;                          // make sure the
setpoints are zero
  }
  heatersOn = LOW;                                  // clear heaters on flag
}
// *****************************************************
void read_button(){
  if (!digitalRead(shtdwn)){                        // if shutdown button is
low
    heatersOn = LOW;                                // turn off heaters
    if (menu < 50){                                 // if in a normal menu
```

```
      printOnce = HIGH;                                   // print new screen
      menu = 100;                                         // move to shutdown menu
    }
  }
  if (digitalRead(button) && lastButton){                 // on releasing button
    delay(15);                                            // add a debounce delay
    lastButton = LOW;                                     // clear button state
  }                                                       // if button is not
pressed, flag low
  if (!digitalRead(button) && !lastButton){               // on pressing the button
    delay(5);                                             // add a delay
    buttonPress = HIGH;                                   // flag if this is a +ve
edge for elsewhere in code
    buttonTime = millis();                                // log time of button
press
    lastButton = HIGH;                                    // update the last button
state
  }
}
// *******************************************************
void blink_LED(int count, int on, int off) {
  while(count > 0){
    count--;
    digitalWrite(13, HIGH);
    delay(on);
    digitalWrite(13, LOW);
    delay(off);
  }
}
```

III. Portable GCxGC – other data

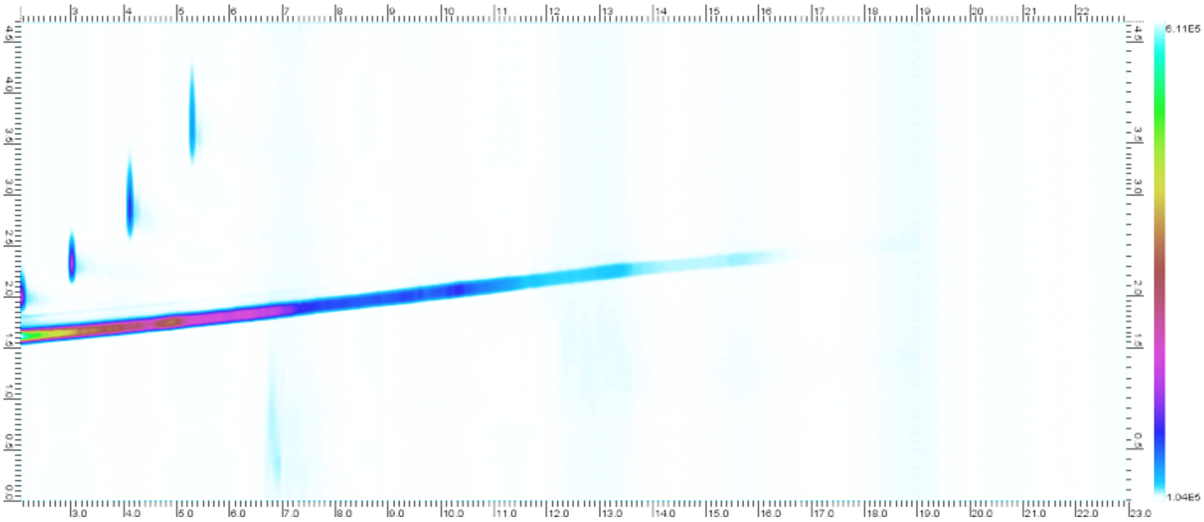


Figure 127 Separation of a C7 to C40 standard using method 3

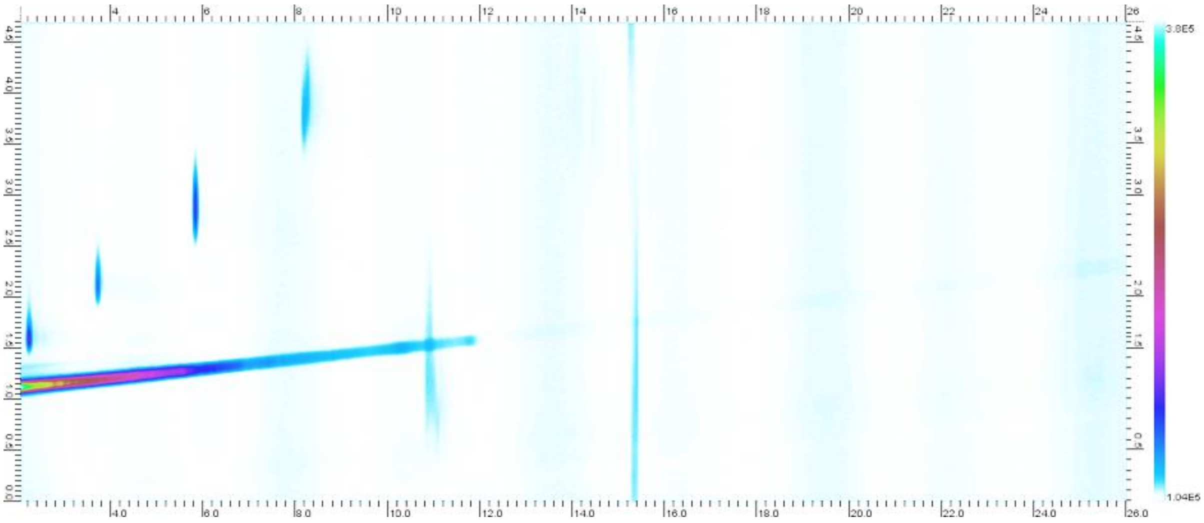


Figure 128 Separation of a C7 to C40 standard using method 4.

IV. Secondary Heater

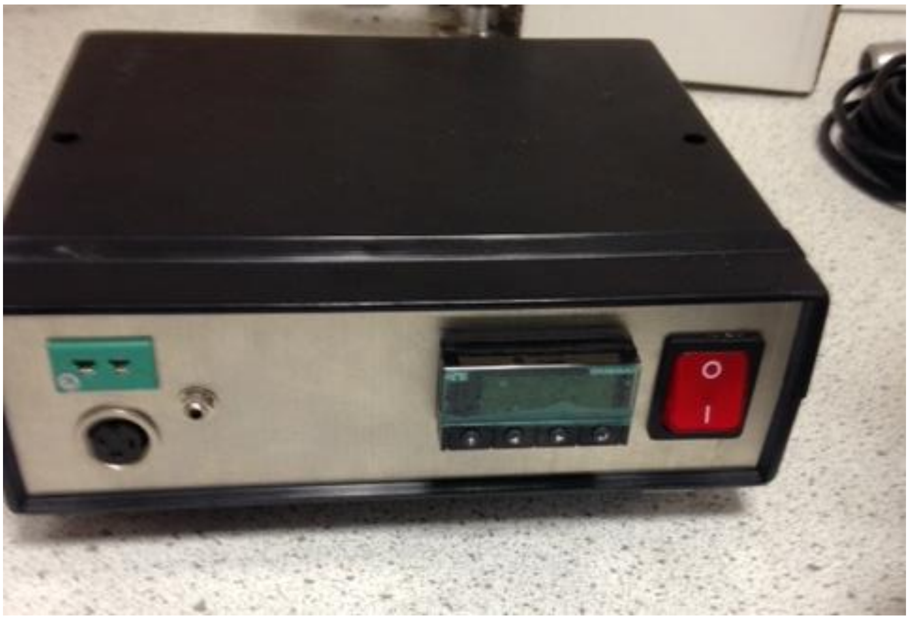


Figure 129 OMEGA fast PID used to control secondary heater

V. C7-C30 Analysed Using Secondary Heater Unit

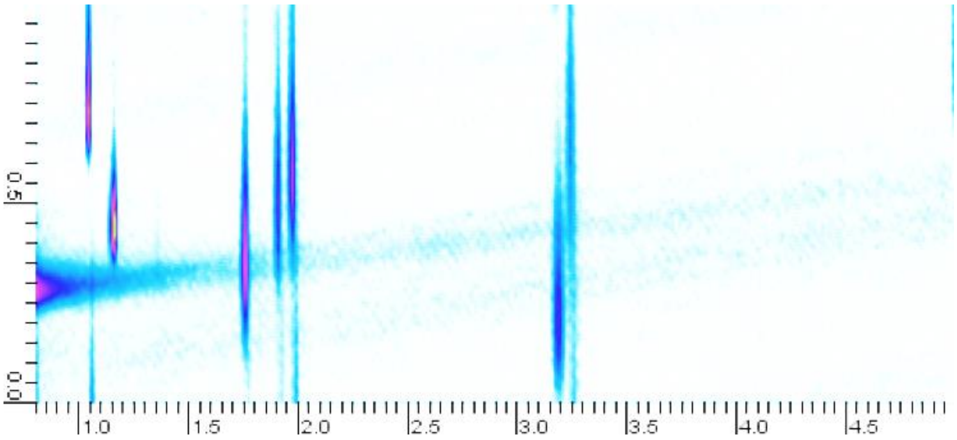


Figure 130 Chromatogram of C7 – C30 at 1 ug/mL analysed on the micro GCxGC unit with a secondary heater held at 200 degrees

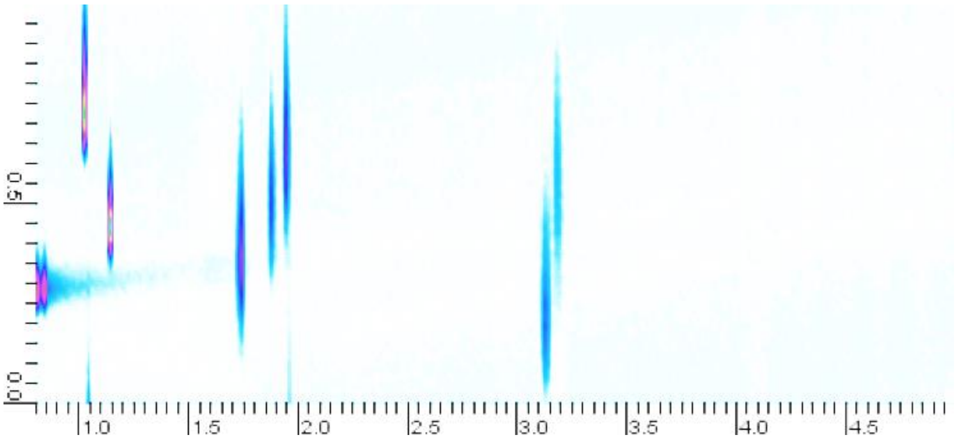


Figure 131 Chromatogram of C7 – C30 at 1 ug/mL spiked with aromatic species analysed on the micro GCxGC unit with a secondary heater held at 200 degrees

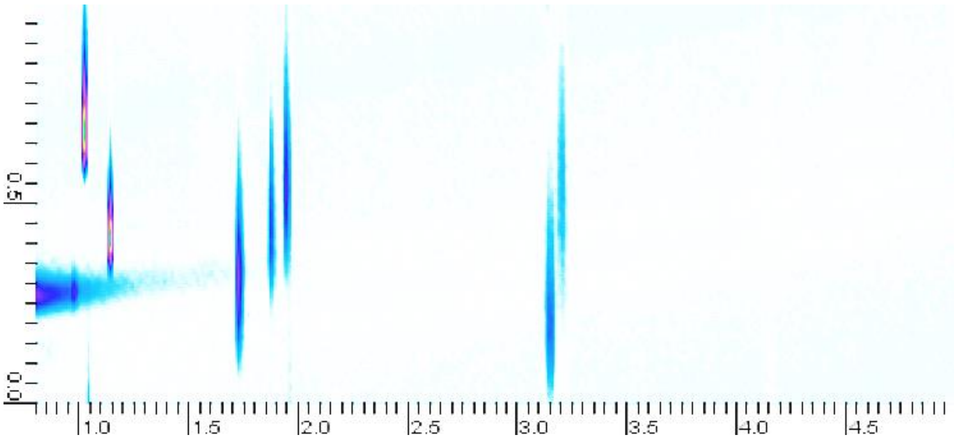


Figure 132 Chromatogram of C7 – C30 at 1 ug/mL spiked with aromatic species analysed on the micro GCxGC unit with a secondary heater held at 200 degrees

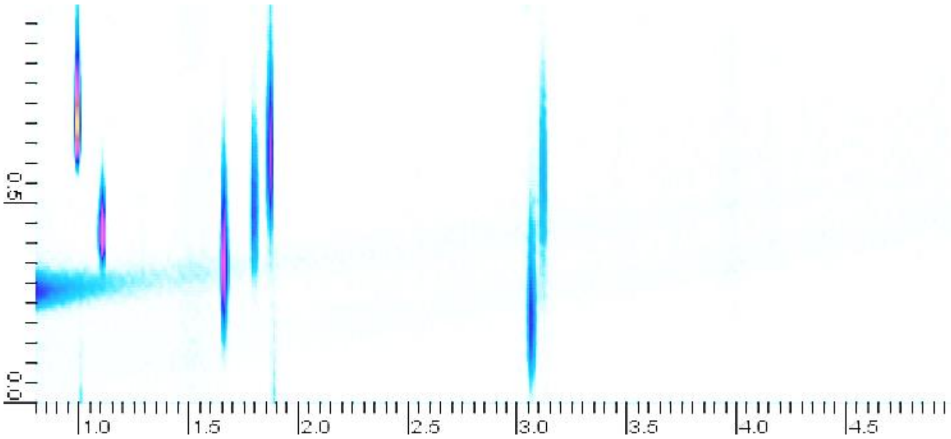


Figure 133 Chromatogram of C7 – C30 at 1 ug/mL spiked with aromatic species analysed on the micro GCxGC unit with a secondary heater held at 200 degrees

References

1. Z. P. O. David Sparkman, Fulton Kitson, *Gas Chromatography and Mass Spectrometry: A Practical Guide*, Academic Press, 2011.
2. L. S. Ettre and K. I. Sakodynskii, *Chromatographia*, 1993, **35**, 223-231.
3. A. J. P. Martin and R. L. M. Synge, *A theory of chromatography.*, 1941, **35**, 1358-1368.
4. A. T. James and A. J. P. Martin, *Biochemical Journal*, 1952, **50**, 679-690.
5. W. Skoog, Holler, *Fundamentals of Analytical Chemistry*, Saunders College Publishing, 1996.
6. Golay Curve, https://www.sigmaaldrich.com/content/dam/sigma-aldrich/docs/Supelco/General_Information/1/t411126h.pdf, (accessed 11/09, 2018).
7. M. C. Simmons and L. R. Snyder, *Analytical Chemistry*, 1958, **30**, 32-35.
8. M. R. Jacobs, R. Gras, P. N. Nesterenko, J. Luong and R. A. Shellie, *Journal of Chromatography A*, 2015, **1421**, 123-128.
9. C. J. Venkatramani and J. B. Phillips, *Journal of Microcolumn Separations*, 1993, **5**, 511-516.
10. D. Deans, *Journal of Chromatography A*, 1981, **203**, 19-28.
11. Z. Liu and J. B. Phillips, *Journal of Chromatographic Science*, 1991, **29**, 227-231.
12. J. B. Phillips and J. Beens, *Journal of Chromatography A*, 1999, **856**, 331-347.
13. J. B. Jens Dalluge, Udo A.Th. Brinkman, *Journal of Chromatography A*, 2003, **1000**, 39.
14. J. C. Giddings, *Journal of High Resolution Chromatography*, 1987, **10**, 319-323.
15. R. Lidster, *PhD Thesis University of York*, 2011.
16. C. J. Venkatramani and J. B. Phillips, *Journal of Microcolumn Separations*, 1993, **5**, 511-516.
17. Z. Liu and J. B. Phillips, *Journal of Microcolumn Separations*, 1994, **6**, 229-235.
18. J. B. Phillips and J. Xu, *Journal of Chromatography A*, 1995, **703**, 327-334.
19. H.-J. de Geus, J. de Boer and A. T. Udo, *Journal of Chromatography A*, 1997, **767**, 137-151.
20. J. B. Phillips, R. B. Gaines, J. Blomberg, F. W. van der Wielen, J. M. Dimandja, V. Green, J. Granger, D. Patterson, L. Racovalis and H. J. de Geus, *Journal of High Resolution Chromatography*, 1999, **22**, 3-10.
21. R. M. Kinghorn and P. J. Marriott, *Journal of Separation Science*, 1999, **22**, 235-238.
22. P. Marriott and R. Kinghorn, *TrAC Trends in Analytical Chemistry*, 1999, **18**, 114-125.
23. J. Beens, M. Adahchour, R. J. J. Vreuls, K. van Altena and U. A. Th. Brinkman, *Journal of Chromatography A*, 2001, **919**, 127-132.
24. M. Pursch, K. Sun, B. Winniford, H. Cortes, A. Weber, T. McCabe and J. Luong, *Analytical and Bioanalytical Chemistry*, 2002, **373**, 356-367.
25. J. V. Seeley, F. Kramp and C. J. Hicks, *Analytical Chemistry*, 2000, **72**, 4346-4352.
26. J. F. Hamilton, A. C. Lewis and K. D. Bartle, *Journal of Separation Science*, 2003, **26**, 578-584.
27. J. V. Seeley, N. J. Micyus, J. D. McCurry and S. K. Seeley, *American Laboratory*, 2006, **38**, 24-26.
28. J. V. Seeley, N. J. Micyus, S. V. Bandurski, S. K. Seeley and J. D. McCurry, *Analytical Chemistry*, 2007, **79**, 1840-1847.
29. Agilent, https://www.agilent.com/cs/library/brochures/5989-9384EN.pdf, (accessed 27/07/2017).
30. J. H. Richard Lidster, Alastair Lewis, *J. Sep. Sci*, 2011, **34**, 812 - 821.
31. R. E. Mohler, B. J. Prazen and R. E. Synovec, *Analytica Chimica Acta*, 2006, **555**, 68-74.
32. L. M. Chris E. Freye, Robert Synovec, *J. Chromatogr. A*, 2015, **1424**, 127-133.
33. BBC, London Bridge attack, http://www.bbc.co.uk/news/uk-40013040, (accessed 11/07/17).
34. T. Guardian, Syria chemical weapons attack toll rises to 70 as russian narrative is dismissed, https://www.theguardian.com/world/2017/apr/04/syria-chemical-attack-idlib-province, (accessed 11/07/17).
35. Independent, Isis could unleash car bombs and chemical weapons on Europe as new terror tactics employed, Europol warns, http://www.independent.co.uk/news/world/europe/isis-

terror-attacks-plots-europe-uk-britain-france-islamic-state-europol-report-car-bombs-chemical-a7451591.html, (accessed 11/07/17).

36. Independent, Manchester arena terror attack, http://www.independent.co.uk/news/uk/crime/manchester-arena-terror-attack-man-arrested-liverpool-john-lennon-airport-suicide-bombing-a7829461.html, (accessed 11/07/17).
37. W. P. W. J. A. Zukas, *Explosive Effects and Applications*, Springer, New York, 1998.
38. C. Kopp, *Defence Today*, 2008, **4649**, 46-49.
39. A. K. H. Schubert, *Detection of Liquid Explosives and Flammable Agents in Connection with Terrorism*, Springer, Netherlands, 2008.
40. M. A.-H. Lama Mokalled, Karim Y.Kabalan, Ali El-Hajj, *International Journal of Scientific & Engineering Reseach* 2017, **5**, 14-15.
41. C. Bruschini, *Subsurface Sensing Technologies and Applications*, 2001, **2**, 299-336.
42. R. G. Ewing, D. A. Atkinson, G. A. Eiceman and G. J. Ewing, *Talanta*, 2001, **54**, 515-529.
43. D. S. Moore, *Sensing and Imaging: An International Journal*, 2007, **8**, 9-38.
44. D. S. Moore and R. J. Scharff, *Analytical and Bioanalytical Chemistry*, 2009, **393**, 1571-1578.
45. UKAS, *Internal Report*, Porton Down, 2016.
46. FEL, *Internal Report*, Porton Down, 2016.
47. M. W. Robert Ewing, David Atkinson, Jay Grate, Peter Hotchkiss, *Trends in Analtical Chemistry*, 2013, **42**, 14.
48. S. W. Henric Ostmark, How Ghee Ang *Propellants Explos. Pyrotech*, 2012, **37**, 11.
49. J. Yinon, *TrAC Trends in Analytical Chemistry*, 2002, **21**, 292-301.
50. B. J. L. J. A. Romano, H. Salem, *Chemical warfare agents chemistry, pharmacology, toxicology and therapeutics*, CRC Press, 2nd edn., 2008.
51. OPCW, History of CW use, http://www.opcw.org/about-chemical-weapons/history-of-cw-use/, (accessed 11/07/17).
52. D. E. D. H. P. Price, *Regular Review: Chemical Weapons* 2002.
53. M. B. A. Richard, *Decontamination of Warfare Agents*, Wiley, 2008.
54. OPCW, Chemical Weapons Convention, https://www.opcw.org/chemical-weapons-convention/, (accessed 12/07/17).
55. A. C. Association, Chemical Weapons Convention Signatories and States https://www.armscontrol.org/factsheets/cwcsig, (accessed 12/07/17).
56. B. News, http://www.bbc.co.uk/news/world-middle-east-23927399, (accessed 22/01/17).
57. CIA, Intelligence Update: Chemical Warfare Agent Issues, https://www.cia.gov/library/reports/general-reports-1/gulfwar/cwagents/index.htm).
58. A. M. H. Mohtadi, University of Minnesota, 2006.
59. B. News, US Says Syria is Preparing Chemical Weapons Attack, http://www.bbc.co.uk/programmes/p056dl8r, (accessed 12/07/17).
60. J. J. F. Worek, H. Thiermann, *Chemical Warfare Toxicology*, RSC, 2016.
61. A. M. Prentiss, *Classification of Chemical Agents*, McGraw Hill, New York, 1937.
62. J. Matoušek, *Chemical Weapons Chemical Warfare Agents*, Prague Czech National Institute for NBC Protection, Prague, 2008.
63. R. L. Maynard, *The Physicochemical Properties and General Toxicology of Chemical Warfare Agents*, Wiley, Chichester, 2007.
64. J. F. Hamilton and A. C. Lewis, *Atmospheric Environment*, 2003, **37**, 589-602.
65. X. Xu, L. L. P. Stee, J. Williams, J. Beens, M. Adahchour, R. J. J. Vreuls, U. A. Brinkman and J. Lelieveld, *Atmos. Chem. Phys.*, 2003, **3**, 665-682.
66. T. H. M. Shimmo, M. Kallio, P. Antifa, M.-L. Riekkola, *LC.GC Eur*, 2004, **17**, 640.
67. R. B. L. Gaines, E. B.; Stuart, J. D., *J. Microcolumn*, 1998, **10**, 597-604.
68. T. Hyötyläinen, M. Kallio, K. Hartonen, M. Jussila, S. Palonen and M.-L. Riekkola, *Analytical Chemistry*, 2002, **74**, 4441-4446.

69. R. Ong, S. Lundstedt, P. Haglund and P. Marriott, *Journal of Chromatography A*, 2003, **1019**, 221-232.

70. S. E. Reichenbach, M. Ni, V. Kottapalli, A. Visvanathan, E. B. Ledford, J. Oostdijk and H. C. Trap, *Jr. Journal of Chromatography A*, 2003, **958**, 45-46.

71. M. R. Gravett, F. B. Hopkins, A. J. Self, A. J. Webb, C. M. Timperley and J. R. Riches, *Analytical and Bioanalytical Chemistry*, 2014, **406**, 5121-5135.

72. B. Gruber, B. A. Weggler, R. Jaramillo, K. A. Murrell, P. K. Piotrowski and F. L. Dorman, *TrAC Trends in Analytical Chemistry*, 2018, **105**, 292-301.

73. M. Claeys, B. Graham, G. Vas, W. Wang, R. Vermeylen, V. Pashynska, J. Cafmeyer, P. Guyon, M. O. Andreae, P. Artaxo and W. Maenhaut, *Science*, 2004, **303**, 1173-1176.

74. D. J. Eatough, R. W. Long, W. K. Modey and N. L. Eatough, *Atmospheric Environment*, 2003, **37**, 1277-1292.

75. H. Church, *CLOUD RISE FROM HIGH-EXPLOSIVES DETONATIONS*, Sandia Labs., Albuquerque, N. Mex., 1969.

76. A. L. Kuhl, *Dynamics of Detonations and Explosions*, American Institute of Aeronautics and Astronautics, Washington DC, 1989.

77. J. L. Thomas, C. C. Donnelly, E. W. Lloyd, R. F. Mothershead and M. L. Miller, *Forensic Science International*, 2018, **284**, 65-77.

78. Y. Seto, M. Kanamori-Kataoka, K. Tsuge, I. Ohsawa, K. Matsushita, H. Sekiguchi, T. Itoi, K. Iura, Y. Sano and S. Yamashiro, *Sensors and Actuators B: Chemical*, 2005, **108**, 193-197.

79. K. Kim, O. G. Tsay, D. A. Atwood and D. G. Churchill, *Chemical Reviews*, 2011, **111**, 5345-5403.

80. J. S. Caygill, F. Davis and S. P. J. Higson, *Talanta*, 2012, **88**, 14-29.

81. P. E. Leary, G. S. Dobson and J. A. Reffner, *Applied Spectroscopy*, 2016, **70**, 888-896.

82. R. J. Hopkins, S. H. Pelfrey and N. C. Shand, *Analyst*, 2012, **137**, 4408-4410.

83. C. Bohling, K. Hohmann, D. Scheel, C. Bauer, W. Schippers, J. Burgmeier, U. Willer, G. Holl and W. Schade, *Spectrochimica Acta Part B: Atomic Spectroscopy*, 2007, **62**, 1519-1527.

84. P. Lucena, A. Doña, L. Tobaria and J. Laserna, *Spectrochimica Acta Part B: Atomic Spectroscopy*, 2011, **66**, 12-20.

85. R. G. Ewing and M. J. Waltman, *International Journal for Ion Mobility Spectrometry*, 2009, **12**, 65-72.

86. T. Khayamian, M. Tabrizchi and M. Jafari, *Talanta*, 2003, **59**, 327-333.

87. M. Tabrizchi and V. ILbeigi, *Journal of Hazardous Materials*, 2010, **176**, 692-696.

88. J. Babis, R. Sperline, A. Knight, D. Jones, C. Gresham and M. Denton, *Analytical and Bioanalytical Chemistry*, 2009, **395**, 411-419.

89. S. Zimmermann, N. Abel, W. Baether and S. Barth, *Sensors and Actuators B: Chemical*, 2007, **125**, 428-434.

90. A. B. Kanu and H. H. Hill, *Journal of Chromatography A*, 2008, **1177**, 12-27.

91. M. Martin, M. Crain, K. Walsh, R. A. McGill, E. Houser, J. Stepnowski, S. Stepnowski, H.-D. Wu and S. Ross, *Sensors and Actuators B: Chemical*, 2007, **126**, 447-454.

92. G. A. Eiceman, H. Schmidt and A. A. Cagan, *Counterterrorist Detection Techniques of Explosives*, 2007, 61-90.

93. S. A. Kasten, S. Zulli, J. L. Jones, T. Dephillipo and D. M. Cerasoli, *Chirality*, 2014, **26**, 817-824.

94. J. E. Kolakowski, S. Harvey and L. P. Reiff, *Gas Chromatographic Analysis of the Stereoisomers of the Chemical Warfare Agent GF*, US Army ECBC, USA, 2002.

95. W. R. Collin, A. Bondy, D. Paul, K. Kurabayashi and E. T. Zellers, *Analytical Chemistry*, 2015, **87**, 1630-1637.

96. J. Lee, S. K. Sayler, M. Zhou, H. Zhu, R. J. Richardson, R. Neitzel, K. Kurabayashi and X. Fan, *Analytical Methods*, 2018, **10**, 237-244.

97. J. Lee, M. Zhou, H. Zhu, R. Nidetz, K. Kurabayashi and X. Fan, *Analytical Chemistry*, 2016, **88**, 10266-10274.

98. S. J. Edwards, A. C. Lewis, S. J. Andrews, R. T. Lidster, J. F. Hamilton and C. N. Rhodes, *Analytical Methods*, 2013, **5**, 141-150.