# 02223 Fundamentals of Modern Embedded Systems
# Group 4


Eleonora Girardini - s182556
Apostolos Lalos - s182452
Colin Buck - s181183
Tolga Tasbent - s173138

All members of the group contributed equally.

December 10, 2018

# Wireless Fire Detection Sensor Network

Eleonora Girardini
s182556@student.dtu.dk

Apostolos Lalos
s182452@student.dtu.dk

Colin Buck
s181183@student.dtu.dk

Tolga Tasbent
s173138@student.dtu.dk

## 1. ABSTRACT

Going from an idea to a real system is a complicated transition that could lead to risks and all sort of problems that should be analyzed beforehand. That is why modeling our system through a simulator is useful. We used an engineering approach to define a model for a forest fire detecting system and develop a simulator that helped us analyzing the different aspects of it, focusing our attention on deployment layouts for the sensors. We compare the connectivity, coverage, reliability, and cost efficiency of each of our defined sensor layouts. Ultimately, we come to the conclusion that arranging the sensors in an optimized grid format provides the best results when using the data parameters we chose.

## 2. INTRODUCTION

### 2.1 Background

Across the world, wild forest fires ravage the land. They create a wide array of problems globally, from harming biodiversity, to causing deforestation and releasing massive amounts of carbon dioxide into the atmosphere [1]. Teams of firefighters are usually quick to respond to these fires once they are found. However, all too often, these fires are not detected until the inferno is large enough that it becomes extraordinarily difficult to extinguish, sometimes even taking days to fully contain. This puts discovering these fires before they become untameable at a very high priority [2]. But this discovery process is a difficult one, since massive forests often contain large portions of remote area that is not watched over. There are many fire detection units used to solve this problem on a small scale such as in a home, but using these solutions in something as large as a forest, is infeasible. Our goal in this project is to efficiently solve this problem of fire detection in a large forest scenario. This is a broad issue, however, so first we must define exactly what aspects of this problem we would like to cover. This will allow us to narrow the scope of our overall question, and create a solution that we can confidently say answers this problem.

### 2.2 Approach

A good and well implemented fire detection system requires much forethought about the different components it will contain. There are many different ways to implement it depending on the plethora of varying questions that can be answered on this topic.

Our first step was been to define the scope of our analysis. We started by thinking about the different components of this system, and what specific questions and problems we wanted to focus on since our study is limited by our time frame. First, we had to decide whether we wanted to implement a system that prevents fire from spreading, or whether we wanted to focus on a system that detects fire. We decided to choose the latter, creating a system that is efficiently able to detect fires and warn fire fighters. This option was chosen because we wanted to focus on the network aspect of the problem, with an emphasis on how we should arrange the network nodes, as well as how many nodes we should deploy to achieve an efficient and reliable network that is able to detect the fire, while minimizing the cost.

The main part of our system is a node that consists of a fire detection sensor, and a communication transceiver that is able to transmit information to neighboring nodes, and eventually to the fire station, which we consider the end point of the network.

Because we decided on narrowing down our research scope to the connectivity and efficiency of the network by analyzing the number of sensors and the layout of the network, we decided to leave out some design considerations. Some of the problems we are leaving out of the scope of our question include energy consumption, implications related to data computation and the protocols used to transmit data, and the possibility of the nodes shifting location over time due to external factors such as rain or erosion.

The overall question we wish to answer here is: determine the sensor layout that is the most effective in terms of having a significant proportion of sensors connected as well as the highest area of forest covered by the sensors, while also being the most reliable.

## 3. SYSTEM DESCRIPTION

### 3.1 Fire Sensors

A sensor typically consists of a small micro controller, a radio transceiver and sensing instruments [3] [4]. A non-optical fire sensor usually measures one or multiple variables. The most common ones are concentrations of H2, CxHx gases, smoke and temperature [5] [6] [7]. Gas sensitive sensors are used to measure H2 and CxHx, while an aspirating system analyses in flowing air to detect fire smoke. Lastly, a thermometer is used to measure fast temperature fluctuations to detect fire. Typically, these 4 measurements are combined in some fashion to increase the accuracy of a system.

In addition to these measurement, some fire sensors also

have fire resistant shells that prolong their lifetime. They might still break if fire sweeps over them, but the improved resistance means that the sensor nodes can track the fire more accurately until they are destroyed, and thus providing more valuable information for fire fighters. In one example, the fire sensor is kept in a glass case to protect the sensor [3], and in another implementation thermal insulation materials is used to create a sock that the sensor is kept in [4].

## 3.2 Wireless Sensor Networks

The architecture of the sensor network is rather simple. Each sensor should have the ability to sense different factors in its environment using its fire detection unit, and compute and transmit data to other sensor nodes using its communication unit.

When building a wireless sensor network, there are many design considerations that must be addressed. Some of the most important issues in a Wireless Sensor Network are energy efficiency, accurate localization and robustness [8]. Without addressing these issues, the lifetime and usefulness of the system diminishes, affecting the overall effectiveness of the network.

In the context of forest fire detection, accuracy location and early detection are particularly important, since fires often becomes unmanageable when they reach a certain size [1] [2]. Fast and precise detection provides firefighters with much better odds at containing fires.

Other important factors in designing a sensor network are scalability, production costs, network topology and hardware constraints [4].

## 3.3 Deployment Strategies

The deployment of sensors can be done in various ways, as seen in the following reports [9] [10] [11] [12]. For a smaller target area, a deterministic deployment strategy is the preferred one, because the placement of each sensor is optimized before deployment. Dividing the area of interest in hexagons and placing a sensor in the middle of each hexagon has shown to be one of the most optimal deployment of sensors. It achieves maximum coverage, while having minimal number of sensors.

In inhospitable and vast environments, it is often preferable to randomly distribute sensors because manual deployment becomes cumbersome and expensive. This is usually done by either aerial vehicle such as drones or helicopters, with parachutes attached to sensors to soften the impact when they reach the ground [9]. An alternative for randomly deploying sensors is a centrifugal cannon-based sprinkler. The resulting network from a centrifugal cannon-based sprinkler performs slightly worse compared to using aerial vehicles to drop sensors randomly and is more difficult to use in a forest setting, but its cheaper and more time efficient for very large areas.

The random distribution can be improved by deciding a deployment path that an aerial vehicle takes when dropping sensor nodes[10]. By having a helicopter cover the entire area of interest and dropping sensors at select or random intervals, the sensors are spread more evenly. An even more specific option is using an aerial vehicle to drop sensors over predefined locations. However, the wind and terrain will affect where the sensor lands, so it will only be approximately around the desired location. Additionally, it will take more time to deploy each sensor at its predefined location.

The random deployment of sensors can also be optimized by allowing the sensor nodes to move. These types of sensors are part of what is known as mobile sensor networks [9]. They can relocate to more suitable positions that maximizes the overall coverage and connectivity.

In a survey[11] various stochastic node placement strategies were examined, where sensor nodes were spread based on different probability density function. These include diffusion strategies such as simple diffusion, continuous, discontinuous and constant diffusion, as well as some more complex ones like R-random, which simulates a spread similar to a shell exploding, and exponential diffusion. The results in the study shows that a constant diffusion is the best solution for achieving high coverage and connectivity, but it quickly deteriorates when sensors expires/fails, showing a low tolerance against failure. This is also in accordance with the research done by Ishuzuka and Aida [12]. In their research they found R-random deployment to have the highest fault-tolerance. The difference in failure tolerance stems from the fact that the sink, which is a central node information can be sent to further distances from, becomes quickly isolated in a constant diffusion, because the concentration of sensors at the sink are low. On the other hand, the R-random and other simple diffusion distributions have the highest concentration of sensors at the sink. They addressed this issue by creating a hybrid diffusion by combining constant diffusion with simple diffusion. The resulting hybrid diffusion showed better overall performance compared to the other diffusion solutions.

When simulating these deployment strategies, it should be noted that they can be implemented with two different sensing models [13]. Sensing range can be deterministic or probabilistic. In a probabilistic sensing model, the sensitivity decreases as the distance from the sensor increases. In a deterministic sensing model, the sensor can only detect inside the sensing range and provides a binary answer. Either it has detected a fire or it hasn't. So, the difference is that there is a hard threshold for the deterministic sensing model, while there is a gradual sensitivity decrease in a probabilistic sensing.

## 3.4 Components

A fire detection system consists of multiple different components.

First, the primary component is the forest area in which we deploy our whole system. The size can vary. It can be a small or a wide area with trees and plants where the sensors have to be deployed in such a way that they create a network that can cover and protect the specified forest area.

Next, we have the sensor nodes that have two sub-components: The fire detection unit that can successfully detect fire in a specific detection range, and a communication unit, which can transmit and receive messages with other nodes it is

neighbors with, in a range that is usually larger than the detection range.

A sensor component is also associated with a cost. The cost can vary depending on if a sensor contains a GPS or not. Because of the way we examine our system in the case of randomly distributed sensors, the implementation requires each sensor to contain a GPS component, so we can track the position after deployment. The trade-off is that the price of a sensor is higher, but it will be cheaper to place. In the grid approach the cost of the component itself is lower, without a GPS, but the cost to deploy will be significantly higher as each sensor has to be placed manually, which increases labor cost.

The fire station is another important component in the system. It plays the role of the main receiver where all of the detection messages end up. Once it receives a message, its role is to immediately forward it to the fire department for evaluation and possibly deploying a response team.

### 3.5 P2P network

In our system, we implement a peer-to-peer network in which we distribute the sensors in various layouts. The main goal is, by passing the message regarding a detected fire from one sensor to the other in the network, this message should reach the central fire station to react how they see fit.

The way the current network works is each sensor has a transmission range. In order for a sensor to be able to forward the message, there should be at least one other sensor somewhere in it's own transmitting range, i.e it should have a neighbor. Then the next sensor should have at least one neighbor different from the one that passed it a message so the message can move forward like a chain, until it eventually reaches the fire station.

## 4. SIMULATION

### 4.1 Simulation Scope

A custom simulator was created and implemented by us, as a tool for testing and evaluating different scenarios. It gives us the ability to introduce and adjust different kinds of parameters and limitations in the system, so that we can analyze various scenarios and evaluate them.

The environment this tool simulates is a forest with a wireless sensor network. In this wireless network, we distribute sensors with different deployment strategies so that the number of sensors and the way they are connected with each other, varies.

We simulate 3 different deployment strategies: Sensors placed in a grid layout, sensors distributed randomly in a constant diffusion, and sensors placed semi randomly. Each of those layouts functions differently, allowing us to compare the different approaches and determine the best solution to implement in a real-world scenario. In each of these scenarios, we place the Fire Station in the bottom right corner of the forest. Additionally, in the randomly placed scenario and semi-randomly placed scenario, we have hardcoded that there are two sensors neighboring the fire station. This was done to eliminate the scenario where the fire station is not connected to any sensors.

A sensor is a small electronic system, and like every electronic system, it is vulnerable to failure at any given time for all kinds of reasons. We therefore implemented an option to randomly fail sensors, so that we could evaluate the fault tolerance of different deployment strategies.

### 4.2 Simulation model and implementation

The simulation takes into consideration many different concepts and components relevant to the problem we wish to solve. This section describes how our simulation models them. Every part of this simulation was written in Java 8.

The main class is called *EmbeddedSystem*, which loads and initializes every part of the simulation. In it, we have defined all the possible operations that the user is able to perform in the UI options panel.

The *MyCanvas* class takes care of the graphic environment in the simulation. When it is called, it displays a rectangle panel, which represents a forest, and a chosen arrangement of the sensors. The sensors are outline as black points with two circles with different radius and colors, blue for the communication range and black for the sensing range.

The fire is displayed as red concentric circles, which randomly appears in a point in the forest and starts expanding.

The *Sensor* class define how we structure the sensor object in our simulation. Each sensor has a position, expressed as an (X,Y) coordinate on a 2d plane, and the two types of range: the communication and the sensing range. For simplicity, the sensing range is implemented as a deterministic sensing model. In this simulation, we make the assumption that the sensors placed on position (x,y) won't move over time. For every sensor we defined two boolean variables: **state** and **forwardMsg**. Those describe the current state of the sensor. The first tells us if it is perfectly working or if something happened and it is not able to perform any activities, which we are calling a "broken" state. The other boolean lets us know if the sensor has received a message from one of its neighbours.

Since our sensors act both as fire detection devices and as communication nodes in a network, for each sensor in the network, we implemented a list of neighbours, which is calculated based on the distance between sensors and their communication range.

The *Fire* class describes the random fire that can be displayed in the simulation and creates a matrix of points equally distributed in the forest that are used to evaluate the coverage of the system. The evaluation, done by *MyCanvas*, is accomplished by counting how many of the fire's points lay inside of at least one sensor sensing range thus if the fire started there it would be detected.

Once we had the basic layout and a perfectly working network, we wanted to be able to simulate different parameters, so we implemented the *OptionClass* class with which we are able to alter some of the simulation's parameters: such as the detection and communication range of the sensors, the amount of sensor and the placing mechanism: grid, completely random and semi-random. Moreover, since our goals

are reliability and efficiency of the network, we implemented a function that randomly fails sensors, which allows us to study the connectivity of the network when some of the sensors don't work anymore.

The *CheckSubNetworks* class investigate if and which of the network nodes are able to transmit a message all the way from its location to the fire station through the its neighboring nodes. In order to do that, every sensor sends a message to every neighbour and we collect the ones that managed to get their message to the fire station, and the remaining ones are considered not connected to the main network.

This class will return a list of every existing sub-network, and which ones are connected to the fire station, so we are able to analyze the connectivity for each deployment strategy.

The *OrToolEmbedded* class uses Google's OR-Tools to optimize our grid deployment layout. It allows you to give it a data set, which in our case is a matrix with the dimensions of the forest, representing all of the possible places a sensor could be placed. A 0 represents no sensor in that place, and a 1 represents a sensor in that place. It then allows you to give it a set of constraints to apply to the data. Our overall goal was to create a sensor layout that maximized connectivity and minimized the total number of sensors. Our main constraint was that every sensor should have at least two neighbors, in order to eliminate the chance of having isolated sensors. Another constraint was that there should be at least three sensors for each row and three for each column so that the sensor are distributed evenly over the forest. The outcomes of this class are saved in a file that will be loaded in simulation to assess the performances.

# 5. ANALYSIS

## 5.1 Parameters

When creating our parameters, we chose values that are plausible, but not exactly identical to anything existing. For example, the sensing and communication ranges of our sensors, as well as their prices, are not infeasible values. However, the sensor ranges and their prices vary an extraordinary amount depending on the technology used, so we decided to choose values that act somewhat as averages to what we have found. But they do not exactly represent a product, and thus are arbitrary values that can be altered by a company that wishes to further investigate this problem with their own sensors and prices that they would like to compare using our simulation tool we have created. Our simulation allows for full control over things such as a sensor's detection and communication range, price, and size of forest, to allow for analysis using the products or data a company would like to compare. This analysis is a demonstration of the conclusion that gets drawn with the parameter values we have provided, but the real strength of this tool is in allowing others to compare their own products and parameter values they are considering. With that said, the found in section A.2 of the appendix are what we used to conduct our analysis and obtain a final conclusion to our question.

## 5.2 Grid layout

We initially begun this project thinking that arranging the sensors in a grid would be the best solution. Arranging each of our sensors in a grid would allow us to be able to place them in such a way that they are all connected to multiple neighbors, and cover the forest as precisely as we would like. Upon running this setup in the simulation and analyzing the results, we learned that this is indeed a very reliable arrangement in the right scenario.

Throughout our analysis, we will define connectivity as the percentage of total sensors that are able to send a message from their location, to the fire station using the sensors around them in the network. We will define coverage as the percentage of the forest that is covered by the fire detection range of a sensor that is also connected to the network. This means that sensors disconnected from the network, and thus unable to communicate with the Fire Station, are not counted in this coverage statistic. This value is obtained by checking how many points in a grid of 289 coordinates are in the detection range of a sensor.

When the sensors in this grid layout are placed such that their communication ranges are always in range of each of their neighbors, we were able to obtain 100% connectivity, and 94.8% coverage. This percentage of connectivity aligns with what we would assume, since we are able to arrange these sensors so that they are all within their neighbors communication range. The percentage for coverage also makes sense. The sensors were placed in a way such that none of the fire detection ranges for the sensors overlapped with each other, but still cover the vast majority of the forest's area. Doing it this way also provides our first attempt to not place more sensors than are needed in order to maximize this connectivity and coverage.

In our scenario, considering the stated parameters for the forest, the grid setup requires 120 sensors plus the fire station to achieve those numbers.

## 5.3 Random layout

When starting to consider cost, we thought that the grid system would probably be more expensive to implement, especially in larger forests, because of the cost of manual labor to place each of the sensors precisely in a grid pattern. With this, we decided to figure out how the data would look if the sensors were randomly placed in the forest instead of placed in a grid pattern. This would allow for an alternative, much quicker, mean of placing the sensors, such as dropping them out of a aerial vehicle over the forest and allowing them to scatter.

The data we collected included finding the connectivity and coverage of a bunch of randomly placed sensors, increasing the number of random sensors being thrown in each iteration. We started with 50 sensors, increasing that number by 10 each trial up to 400. Each of the trials for each 10 sensor increase was ran 30 times, so an accurate normal distribution could be calculated, storing the data in an Excel file written to by the simulation.

We then calculated the average for every interval on how many sensors were indeed connected to the fire station, and

turning that into a relative percentage based on the number of sensors thrown in that trial. The same approach was followed for the data associated with the coverage. The results are displayed in the following graph.

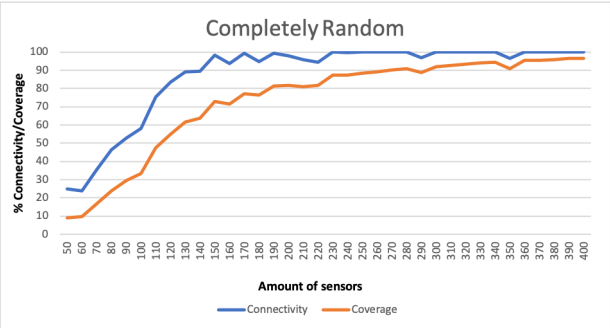


**Figure 1: Completely Random Layout**

As we can see in the graph, the connectivity and coverage of this completely random layout when using the same number of sensors as the grid, 120, is much lower. At this point, the connectivity is 83.5%, and the coverage is 54.9%. These are significantly worse values in comparison to the grid. Although the connectivity of a randomly placed system becomes close to the grid's 100% once 150 sensors are thrown, the coverage of this randomly placed system doesn't get close to the grid's 94.8% until 340 sensors are thrown. Both of these percentages obtained make logical sense. In the case of both connectivity and coverage, more sensors would be needed in the completely random layout in comparison to the grid layout because there is no longer any control over where these sensors end up being located. Some sensors may be right on top of each other, reducing coverage, and some may be far away from the rest, causing them to be disconnected from the network.

## 5.4 Semi-Random

Being unsatisfied with the results of the random deployment, we decided to see if there was a way we could improve it, especially its coverage. Noticing the high coverage of the grid system, we wondered if that could be utilized somehow to improve the situation. With this, we chose to implement a layout that is a combination of the grid and completely random approaches. It is based on dividing the forest into a 4x4 grid (16 sections), and randomly placing the sensors in each of the squares created by this invisible grid. In an attempt to improve consistency compared to the completely random approach, we say that the density distribution in each grid section must be equal, so each square will have the same number of sensors. While both the completely random and semi-random deployment strategy could be implemented with aerial vehicles, the latter would require more precision and deployment time.

We follow the same procedure to collect and analyze the data regarding the connectivity and the coverage as we did in the grid and completely random layouts. We begin with 50 sensors being thrown, and increase that number by 10 each trial after doing 30 iterations of that number. Doing this, we obtain the following results.

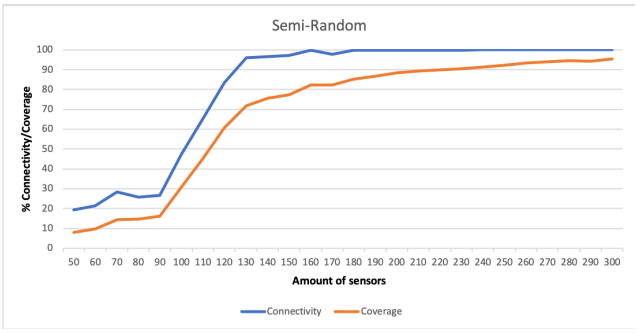


**Figure 2: Semi Random Layout**

The results for this concept were a bit more lackluster than we were initially hoping. Because there is less randomness overall here, the number of sensors needed to reach the grid's connectivity and coverage is less than what is required in the completely random layout. However, it is not better by much. We can see that around 160 sensors would be needed to match the grid's connectivity, and around 280 would be needed to match the grid's coverage. This means that the results here are an improvement to the completely random, but the number of sensors required is still more than double the amount needed in the grid. However, these numbers don't mean as much when we don't know how much each of these three systems would cost to get similar results. This is what we will explore next.

## 5.5 Cost comparison

Now that we have the results of the connectivity and coverage provided by each of the different modes, we can do some monetary comparisons. Using the prices found in the parameters tables, as well as the numbers of sensors that can be placed each hour, we can calculate the cost it would take to create each of these systems at the point where they would have the connection and coverage similar to the grid. We are considering this point to be where there is at least 99% connectivity, as well as at least 95% coverage, though in each of our scenarios, the connectivity requirement is met much before the coverage one. The following is the equation we are using to calculate the cost of implementation. Note that the parameter values for each variable can be replaced with different values depending on the prices of the sensors being compared, or differing labor costs around the world.

(Cost of Sensor) * (Nbr. of Sensors)) + [((Nbr. of Sensors) / (Nbr of Sensors Placed per Hour)) x (Labor Cost per Hour)]

Using this equation, we obtain the following results for the cost to implement each of the three sensor layouts we collected data on. Note that the price of a sensor in either of the random layouts is $10 more than the grid because of the required GPS chip.

The results here were quite surprising to us, even though we are using our own parameter values that may or may not be similar to those that a future company utilizing this tool

| | Grid | Random | SemiR |
|---|---|---|---|
| Number of Sensors | 121 | 340 | 280 |
| Sensor Price | 40 | 50 | 50 |
| Labour hours | 12.06 | 1.42 | 1.52 |
| Total price deployment | $6655 | $17255 | $14280 |

**Table 1: Price Comparison**

would use. We can see that the price of implementing either the completely random or semi-random layouts is drastically higher than the price to implement the grid system due to the huge amount of surplus sensors that we need to use in comparison to the grid. This led us to figuring out how these results are possible, since we had expected the grid to be more expensive due to the increased amount of time it would take for people to arrange the sensors carefully in the specific pattern. However, looking at the parameter values we are using, these results do make logical sense. We can easily see that it would take around 12 hours to place the grid sensors, since that pattern requires 120 sensors to be placed at a rate of 10 sensors per hour. This is between 10 and 11 hours more than the completely random or semi-random approaches would require. When faced with the fact that the two random approaches require hundreds of additional sensors to be thrown, that also cost slightly more than the grid sensors due to requiring a GPS chip, the higher overall cost of that number of sensors more than overshadows the amount of money saved by requiring less manual labor time. But, even though our parameter values give us this conclusion, a company that wishes to run this analysis with our tool could get a completely different result depending on the values they use as price and range parameters.

## 5.6 Reliability

Another metric we wanted to use to compare these three deployment options, was reliability. In our case, we are defining reliability as the percentage of the system's total sensors that can fail before the connectivity drops below 80%. Another way to think of this statistic is the "fault-tolerance" each system has. This value is important to us because we would like to know how often the broken sensors in each system would have to be replaced.

To conduct this experiment, we start each deployment type with the number of sensors required to have 99% connectivity and 94.5% coverage, the values we have been using as a baseline because it is what our grid pattern is able to produce by default. Then, we fail five sensors at random in each deployment type, increasing that number of failures by five each trial. Each increase in the number of sensors failed is ran a total of 30 times before increasing the number of failures, in order to obtain an accurate normal distribution of data. With each iteration, we calculate the connectivity and coverage of the system with that number of failed sensors. Using one of our changeable parameter values, we say that around 10% of the total sensors in a network will fail per month.

Using the results of connectivity and coverage per broken sensor in each layout, we created a graph to compare each system. This graph represents the connectivity of each layout based on the percentage of the total sensors that are broken in each system. Comparing percentage of sensors that are broken, instead of the actual number of sensors that are broken, allows us to get a more true comparison of reliability since each system has a vastly different total number of sensors. The results of this can be seen in Figure 3.

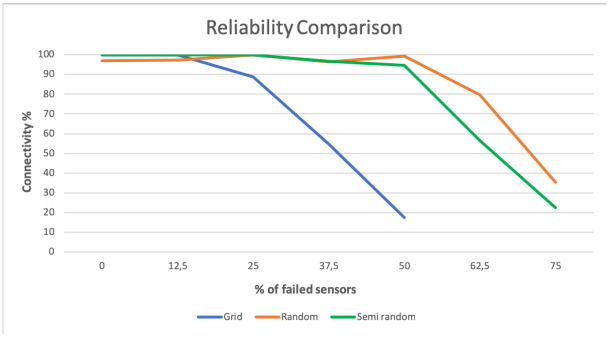


**Figure 3: Reliability Comparison**

It can be clearly seen that the random layouts are actually the most reliable here. Over 50% of the sensors have to fail before the network connectivity of the remaining sensors drops below 80%, whereas only about 30% of the sensors have to fail before this same reliability threshold is met. We can also see that coverage follows a similar trend as connectivity, as seen in Figure 5 in the appendix. The coverage of the grid layout begins to suffer much quicker than either of the random scenarios. Even though the random layouts appear to be more reliable in the sense that a higher percentage of the total sensors have to break for the connectivity and coverage to suffer, we do not yet know if it is cheaper to replace the broken grid sensors more often, or the broken randomly placed sensors less often. Using our parameter of show many of the total sensors will break per month, and the data collected about how many sensors must fail in each layout to reach the 80% connectivity threshold, we can calculate the number of times the broken sensors will have to be replaced per year, as well as how much money this would cost each year. The number of sensors that need to break to reach this 80% connectivity threshold, is what is represented by the 80% column in the following price calculation table.

| Mode | 80% | Replace/year | price/year |
|---|---|---|---|
| Grid | 30 | 4,8 | $7920 |
| Completely Random | 150 | 2,24 | $17136 |
| Semi-random | 215 | 1,897674419 | $20706 |

**Table 2: Maintenance Comparison**

Upon evaluation, it can be seen that it is actually still cheaper to maintain the grid system than it would be to maintain either of the random layouts. However, this is only based on our equation we use for calculating cost, which may not be perfectly reflected in the real world. This is because we calculate the number of hours it would take to replace these sensors, based on the number of sensors that need to be placed, divided by the number of sensors that can be placed per hour. This calculation does not take into account the

amount of extra time it would take to travel through the forest to place these sensors, regardless of the number that needs to be placed. So having to travel through the entire forest almost five times a year in the grid pattern, might take more time than we give credit for, when the forest would only need to be traveled through twice in either of the random layouts. However, using our parameters and calculations, we are able to conclude that the grid system is still the cheapest layout to maintain.

## 5.7 Optimal grid

After concluding that both the completely random and semi-random approaches were much more expensive to implement than the grid in order to obtain the grid's connectivity and coverage, we decided to try to optimize our grid solution. For us, this optimization meant removing as many sensors from the grid pattern as we could, while still keeping 100% connectivity and sacrificing as little coverage as possible. Doing this would allow us to minimize the cost of our grid system, while still reaping the benefits of the good connectivity and coverage it provides.

To do this, we first had to determine what an "optimized" grid pattern would look like, which we did using Google's OR-Tools.These constraints were made with thought of only using as many sensors as are needed to keep connectivity at 100%, while still keeping the coverage high as well. The tool then generates every possible result with those constraints and that data set, starting with the results following the constraints that use the fewest number of sensors. Doing this ultimately gave us a new layout for how to arrange the sensors in a grid pattern, while leaving some gaps where sensors would normally be in order to use fewer sensors in the system. Although this may not be truly the most "optimal" solution, this is our attempt at finding a more optimized solution than the grid we started with. The solution we found has a total of 90 sensors and is shown in Figure 4. Looking closely at it, one can see that it follows the normal grid pattern, but certain spots where a sensor would normally be, have been removed.

One of the first things we can compare between the normal grid solution and the optimized grid, is their connectivity and coverage. When the normal grid is using its full 120 sensors, and the optimized grid is using its full 90 sensors, we still get 100% connectivity on both layouts because one of the constraints we gave the OR-Tool was that the every sensor needed to have at least one neighbor, which would cause it to have full connectivity. But the optimized grid has 76.81% coverage compared to the normal grid's 94.8%. This makes logical sense since the optimized grid has 30 fewer sensors in it to provide fire detection coverage, even though every sensor is still connected to each other.

We can also compare the optimized grid's connectivity and coverage to what a normal 120 sensor grid solution with 30 random failed sensors looks like. This allows us to compare the optimal solution to a grid solution if it was filled randomly with 90 sensors instead of the optimal placement.

The results we obtain by doing this are the grid of 90 arbitrarily placed sensors has a 95.63% connectivity, and 65.09 % and coverage on average after this is ran across 30 trials of random grid placement. Seeing that these results are much lower than those calculated from our optimal grid, we can conclude that our found optimal solution is indeed optimized in comparison to if 30 random sensors were removed from a normal grid solution. This experiment was run by creating the 120 sensor grid, and then failing 30 sensors at random, to simulate 90 sensors being placed at random in a grid layout.

In table 6 we compare the cost of deploying these two solutions using the same equation we have been using to calculate cost.

| Mode | Price |
|---|---|
| Grid | $6655 |
| Optimal Grid | $4950 |

**Table 3: Deployment Cost**

The optimized grid being about $1700 cheaper to deploy than the normal grid makes logical sense, since 30 fewer sensors would need to be placed. To follow suit with the types of comparisons we had made in past trials, we could now compare the reliability between the normal grid and the optimized grid, using the same means as how we had done it before.


**Figure 4: Grid Reliability Comparison**

As we can see, the optimized grid has a much lower fault tolerance than the normal grid. Again, this is a statistic that makes sense since there are fewer sensors to start with in the optimal solution. Each sensor removed from the optimal solution has a higher chance of disconnecting neighboring nodes from the rest of the network, since there are fewer neighbors each node can have in the first place. The coverage follows a similar pattern as the number of sensors failing increases. The amount of area covered by the optimal grid decreases much quicker than the normal grid, because of the fact that it requires fewer disconnected sensors to cause mass disconnection within the system. The trend here can be seen in Figure 6 in the Appendix.

Finally, we can compare the cost efficiency in terms of maintaining the two systems.

Looking at the results, we can see that it is still cheaper to maintain the optimized grid system, even though it requires

| Mode | Price |
|---|---|
| Grid | $7920 |
| Optimal Grid | $5940 |

**Table 4: Maintenance cost per Year Comparison**

many more trips per year to replace the broken sensors once the connectivity drops below 80%.

Even though the optimized grid layout is cheaper to implement and maintain compared to the normal grid, it is at the cost of a bit of coverage. So someone that chooses to implement this layout must be willing to sacrifice a small amount of coverage to do this. Someone that would rather keep their coverage at its maximum may choose to stay with the normal grid pattern.

# 6. CONCLUSION

Designing a forest fire detection system is a very difficult task. We implemented and used a custom simulator to compare different deployment strategies for sensor networks in order to get the best results for reliability, coverage, connectivity and cost. After many different configurations, and using our own arbitrary parameter values, we ended up with some very interesting results. In our own personal scenario, we ended up coming to the conclusion that the grid layout was most cost effective to obtain the best connectivity and coverage, and that we could further optimize our grid by strategically removing some of the sensors in the layout. However, it should be emphasized that this is only the conclusion that is drawn using our own sensor prices, sensor ranges, and forest size. We want this to be a tool that other companies can utilize with their own values for these parameters, potentially obtaining a different conclusion. This report should function as an example for how this process can be done, and how we ended up coming to our own conclusion.

Our simulator was very useful because it gave us the possibility and the opportunity to test and evaluate different kind of scenarios and cases that, sometimes, lead us to different directions and follow different approaches that we didn't think about from the start. Throughout the whole process of creating and testing with the simulator we came across difficulties and obstacles that we had to evaluate and overcome in order to retrieve useful results. Each time we had to face a problem we modified or added more functionality to the simulator. The question that we ended up answering in this report, was very different from the original question we intended to answer when we began this project.

We ended up with a useful and powerful tool that can generate very interesting information and data for the current project. Of course it is still open for more adjustments and optimization.

# 7. FUTURE WORK

Our simulation is just an initial step into the field of research related to deployment of sensors, and there is room for more implementation to make a more comprehensive study of deployment strategies. The way we measure coverage, could be improved to actually resemble how coverage is measured, instead of checking how many points in a grid of fire points, fall inside of the detection radius of sensors. Our way is not completely scientific, but serves as a representation for what the coverage of our layouts would look like.

In our research, we also saw that there were multiple different deployment strategies with varying performances. In our simulation we only implemented a constant diffusion and constant diffusion mixed with a grid layout. For future works other strategies should be examined, such as simple diffusion and hybrid diffusion, as well as different deployment paths that can be used to scatter sensors more optimally.

The complexity of the simulation can also be increased. Since energy consumption are such a vital part of wireless sensor networks, it could be important to include it when analyzing different deployment strategies. They might change the perception of which strategies are the best solution or entirely different strategies might be needed. Additionally, the simulation does not take into account how different communication protocols such as Bluetooth 5.0 and ZigBee [14] affect the system, which could have an effect on how plausible some of the different layouts may be.

Something that we partially implemented, but did not have time to do further analysis on, is the way each node communicates with another. Currently, our nodes are able to communicate with each other by broadcasting a message to all nearby nodes. We have also partially implemented a way of sending messages from the sensor that detects the fire, to the fire station, using shortest-path routing. This needs further work though to be able to do any analysis pertaining to this concept.

Our implementation is also void of simulating the impact of fire on the sensors. This could be interesting to examine. An example could be to examine if using utilizing protective frames for sensors in deployment strategies with low fault-tolerance is an efficient way to improve robustness.

Another issue that we have partially implemented is the impact environmental variables can have on how the system functions. We can simulate how weather change can influence the system, by having it automatically reduce the communication range of the nodes when the weather is bad. However, this idea fell out of the scope of our final question we wished to answer. So this is an idea that could be elaborated on in the future, to see how bad weather can effect each of our sensor layouts, or of the network in general.

# 8. REFERENCES


[1] Forest Europe. *Assessment of Forest Fire Risk and Innovative Strategies for Fire Prevention*. 2010.

[2] Mark A Adams. Mega-fires, tipping points and ecosystem services: Managing forests and woodlands in an uncertain future. *Forest Ecology and Management*, 294:250–261, 2013.

[3] P Vivek, G Raju, and S Akarsh. Forest fire detection system. *International Journal of Innovation Research in Science, Engineering and Technology*, 3(6):714–718, 2014.

[4] Thierry Antoine-Santoni, Jean-François Santucci, Emmanuelle De Gentili, Xavier Silvani, and Frederic Morandini. Performance of a protected wireless sensor network in a fire. analysis of fire spread and data transmission. *Sensors*, 9(8):5878–5893, 2009.

[5] Wolfgang Krüll, Robert Tobera, Ingolf Willms, Helmut Essen, and Nora von Wahl. Early forest fire detection and verification using optical smoke, gas and microwave sensors. *Procedia Engineering*, 45:584–594, 2012.

[6] Vinay Chowdary, Mukul Kumar Gupta, and Rajesh Singh. A review on forest fire detection techniques: A decadal perspective. *International Journal of Engineering Technology*, 7(3.12):1312–1316, 2018.

[7] Javier Solobera. Detecting forest fires using wireless sensor networks with waspmote — libelium, 2010.

[8] Kechar Bouabdellah, Houache Noureddine, and Sekhri Larbi. Using wireless sensor networks for reliable forest fires detection. *Procedia Computer Science*, 19:794–801, 2013.

[9] Vikrant Sharma, RB Patel, HS Bhadauria, and D Prasad. Deployment schemes in wireless sensor network to achieve blanket coverage in large-scale open area: A review. *Egyptian Informatics Journal*, 17(1):45–56, 2016.

[10] Vikrant Sharma, RB Patel, HS Bhaduria, and D Prasad. Policy for random aerial deployment in large scale wireless sensor networks. In *Computing, Communication & Automation (ICCCA), 2015 International Conference on*, pages 367–373. IEEE, 2015.

[11] Mustapha Reda Senouci, Abdelhamid Mellouk, and Amar Aissani. Random deployment of wireless sensor networks: a survey and approach. *International Journal of Ad Hoc and Ubiquitous Computing*, 15(1-3):133–146, 2014.

[12] Mika Ishizuka and Masaki Aida. Performance study of node placement in sensor networks. In *null*, pages 598–603. IEEE, 2004.

[13] Avinash More and Vijay Raisinghani. A survey on energy efficient coverage protocols in wireless sensor networks. *Journal of King Saud University-Computer and Information Sciences*, 29(4):428–448, 2017.

[14] Jennifer Yick, Biswanath Mukherjee, and Dipak Ghosal. Wireless sensor network survey. *Computer networks*, 52(12):2292–2330, 2008.


# APPENDIX

# A. SCRUM

Throughout the development of our project we tried the Scrum approach in order to keep everything organized and helped us improve the code for the simulator. By breaking the project to many smaller tasks and assign them to the members, we had the ability to work independently and more efficiently in short period of time. We picked a Product Owner and a Scrum Master and every week we had meetings with all of us to evaluate and merge the code and discuss about the results. After that we would discuss what we should do next, creating and assigning more tasks for the next week. This process proved to be very effective useful and worked pretty well.
For the tasks and timetables we used the tool called Trello available online. The process we followed was creating lists of TODOs and Completed items so that we know at any time what had to be done and where we should focus on.

## A.1 How to Run the Simulation

When the program is run, 2 windows are opened, one showing the deployment of sensors and their communication (blue circle) and sensing ranges (black circle), and another control window showing different options/functions that can be changed or used.

(Resize the Control Window to show all the options. It hides some of the options when first opened)

In the top there is an **CheckNet** option that computes subnetworks (networks that are isolated from other subnetworks) in the deployment.

After CheckNet is run, the **Print Sub Network** option can be used. It prints all subnetworks and which sensors belong in each subnetwork in the console. It also displays a message with the number of sensors connected to the fire station and the number of subnetworks. If there is only one subnetwork all of the sensors are connected to the fire station (by multihopping).

The **Shortest Path** options starts a fire in the simulation and calculates shortest paths from the sensors when the fire reaches their sensing range. It draws the shortest paths from the sensor to the fire station. This shortest path can sometimes bug out and stop the simulation. To run the shortest path, CheckNet has to be pressed.

The **Weather** option reduces the communication range of all sensors. It was implemented to simulate the effect of weather on the wireless network.

The **Fail Sensors** options allows the user to input the number of sensors he/she wants to fail to test the fault-tolerance of the network.

The **Start/Stop Fire** option as the name indicates gives the user the option to control the fire, if he/she wants it to the stop spreading and examine the system.

**Random Fire** starts a fire at a random coordinate in the system. When it is inside a sensing range of a sensor, the fire

is halted, and the sensor broadcasts a fire detection message that is forwarded to the fire station.

The **Coverage** button displays a 17x17 fire grid that we used to calculate the coverage.

**Width** allows the user to input a custom width of the forest.

**Height** allows the user to input a custom height of the forest.

**Mode** Allows the user to change deployment strategies. In our analysis we examine the options grid, completely random, semi random section and optimal.

**Sensing Range** and **Communication Range** allows the user to change the functionality of sensors.

With **Sensors** the user can change the number of sensors, but only for random deployment strategies.

with **DrawNeighbour** the user can chose a specific sensor by inputting its label and see who its neighbors are.

The **Draw subnet** allows the user to input the subnetwork number after running CheckNet. It colors the specified subnetwork, so the user is able to distinguish it from the rest of the network.

**Show Sensor Label** options displays the sensor number over each sensor. Although, it can only be used once, because it will not be cleared if a new deployment strategy (mode) is used.

With the **Save** button the user is able to save a specific deployment network and Load it later.

**Print Excel** button is used to print out all information such as communication and sensing range as well as if a sensor has failed or has received a fire detection message. This information is stored in an excel file.

Both **Print To Excel** was made to print out data such as connectivity and coverage to excel files. We used this data to analyze the different networks. The first textfield furthest to the left allows the user to input the initial number of sensors or failed sensors based on which the you use. The textfield furthest to the right is the upper limit for the number of sensors/failed sensors. The textfield in the middle is used for how to input the intervals until you reach the upper limit. Each iteration is run 30 times to have more data to analyze. For example in the Print to Excel (Sensor Amount) if the input is 50, 10, 200. The simulation is run 30 times with 50 sensors with whichever deployment strategy is picked. After that, it increases by 10 sensors to 60 sensors and runs this 30 times. Next time 70, and so on until it reaches 200. For each iteration the connectivity, coverage, amount of sensors, failed amount of sensors and subnetworks are collected and outputted to an excel file.

## A.2 Parameter Tables

| Element | Size |
|---|---|
| Sensor Detection | 50m |
| Sensor Communication | 120m |
| Forest | 600m x 600m |

**Table 5: Size Parameters**

| Component | Price |
|---|---|
| Sensor Base | $40 |
| GPS Chip | $10 |
| Labor/hr. | $150 |
| Nbr. Sensors that Break per Year | 10% of total |

**Table 6: Price Parameters**

| Mode | Price |
|---|---|
| Grid | 10 |
| Completely Random | 200 |
| Semi-random | 150 |

**Table 7: Nbr. Sensors that can be placed per hour**

## A.3 Graphs

Graph comparing the coverage percentages over the failing sensors.

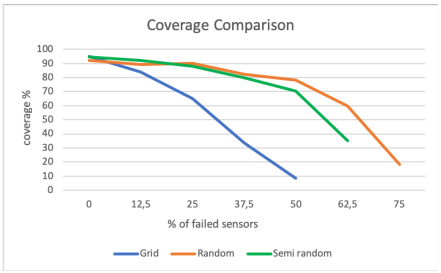


**Figure 5: Coverage Comparisons**

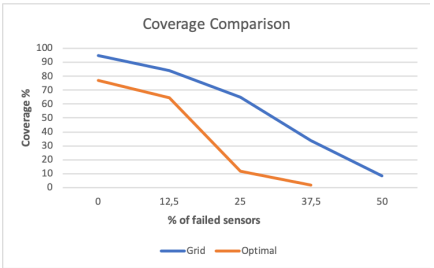


**Figure 6: Grid and Optimal Grid Coverage**

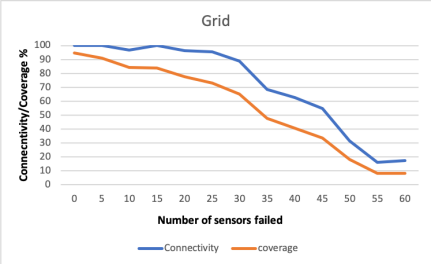


**Figure 7: Grid Reliability**

## A.4 Layouts

Here are the layouts that we mentioned but there wasn't space to include.

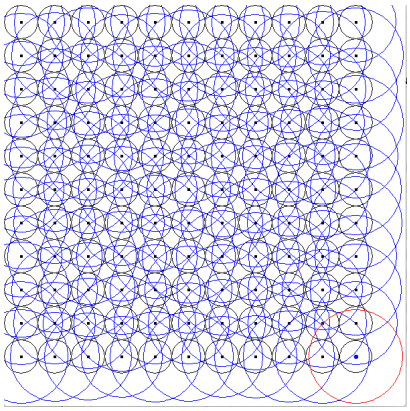

**Figure 8: Grid Layout**

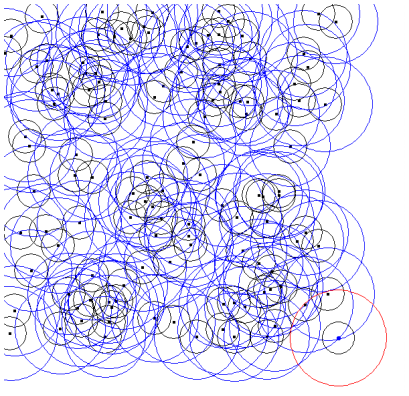

**Figure 9: Random Layout**

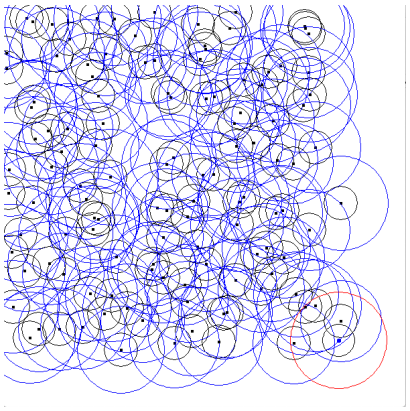

**Figure 10: Semi Random Layout**

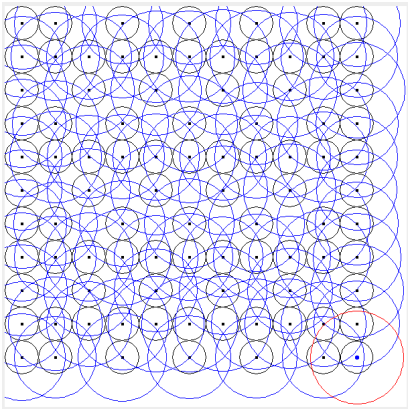

**Figure 11: Optimal Grid**

## A.5 SysML

A diagram representing the Blocks and Atomic Flow of the system. It represents how the system starts with a node detecting a fire, which then communicates with the nearby nodes saying that it has detected a fire, and the location it is at when it detected the fire. These neighboring nodes forward this message to their own neighbors, until eventually the message reaches the fire station. Here, the fire station is able to dispatch fire fighters to the location described in the initial fire detection message.

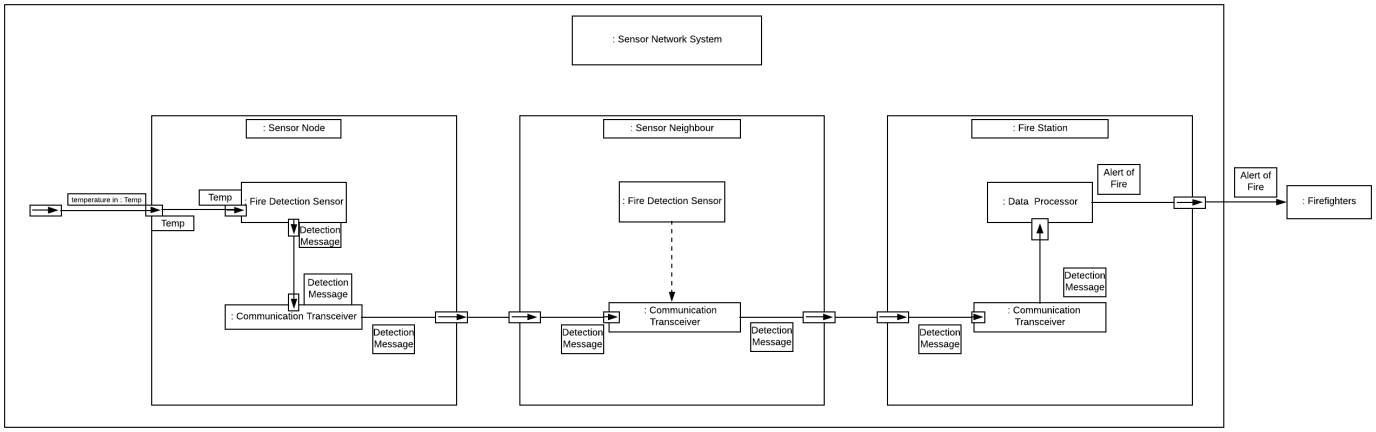


**Figure 12: SysML blocks and atomic flow**