Fire Fighter
Fire Simulation and Extinguishing

Course Project
CS 684 : Embedded Systems

Team 8
Jeet Patani [10305001] (Team Leader)
Kaustubh Keskar [10305909]
Anjali Singhal [10305919]

under the guidance of
Prof. Kavi Arya

Department of Computer Science and Engineering
Indian Institute of Technology, Bombay
Mumbai

1. Introduction

There exists a lot of places without continuous human presence. Such places are like data centers, inventories or any such big unmanned place. These places have probability of catching fire. Since those places are unmanned, it can take long by the time some action is taken to extinguish the fire. Even if we put the fire alarms, it takes time to fire brigade to reach to the location. By that time it can cause huge loss of the property.

In this project, we aim to build an automated fire extinguishing system. It does not require any human presence. It can start fire extinguishing immediately so that the fire does not spread a lot and can be controlled easily. As soon as the fire starts, human fire brigade is also informed to be on the safe side.

In the lab, we can use a candle or a lamp as a fire and extinguish it. But that does not model the real world fire. Real world fire has a characteristic of spreading which is difficult to deal with in the process of fire extinguishing. This cannot be achieved by modeling the fire using a candle or a lamp. We need to model the fire at the first place and then develop the system to extinguish it.

The major challenges in this work was how to simulate fire using electronic hardware like bots. The next challenge was to schedule the fire extinguishing task such that the spreading fire must come to an end at some point of time. The scheduling involves how the fire fighter should move on the arena in order to finish the fire. In this work we simulated the fire using Spark V bots and ZigBee radios. We also programmed Firebird V to act as a fire fighter to move on the arena. Here we assume that the fire fighter is aware of arena map.

2. Problem Statement

The goal of this project is to develop an embedded system which can automatically detect and extinguish the fire. In first place, we aim to simulate the real world fire using Spark V bots. By real world fire, we mean that the fire which can spread, not the one like candle. Once the fire starts, our goal is to extinguish the fire. At first place, the system must inform human authorities about the fire and it should start extinguishing the fire such that it should end at some point of time. We aim to schedule the fire fighter such that it can extinguish fire on all nodes.

3. Requirements

3.1 Functional Requirements :

Fire Simulation :

- The fire can initiate on any node in the arena.
- The fire should increase with the time and it should spread to its neighbors after some time.
- When fire fighter approaches the node, it should bring the fire to an end.
- It should be able to catch the fire again after extinguishing of fire.
- Neighbors of a node in the arena must be configurable, i.e. the user should be able to change the neighbors of the nodes without modifying the code that simulates the fire.

Fire Fighter :

- Fire fighter should send an SMS to the authorities as soon as the fire starts.
- Once fire starts, it should go to all the nodes one by one which are on fire and extinguish the fire.
- Once the fire is extinguished, it should stop.

3.2 Non-functional Requirements :

Hardware Requirement :

- Spark V bots
- Zigbee radios.
- Firebird 5 bot.
- GSM Modem.
- Zigbee adaptor.
- Male-to-male 9 pin RS-232 serial cable.
- Power supply / batteries.
- AVR Programmar.

Software Requirement :

- AVR Studio.
- X-CTU.

4. Implementation

Implementation of the project is mainly divided in fire simulation and fire extinguishing. Fire simulation is implemented on Spark V bots where as fire extinguishing is implemented on Firebird 5 bot. The overall scenario of the implementation is as shown in the figure 1.

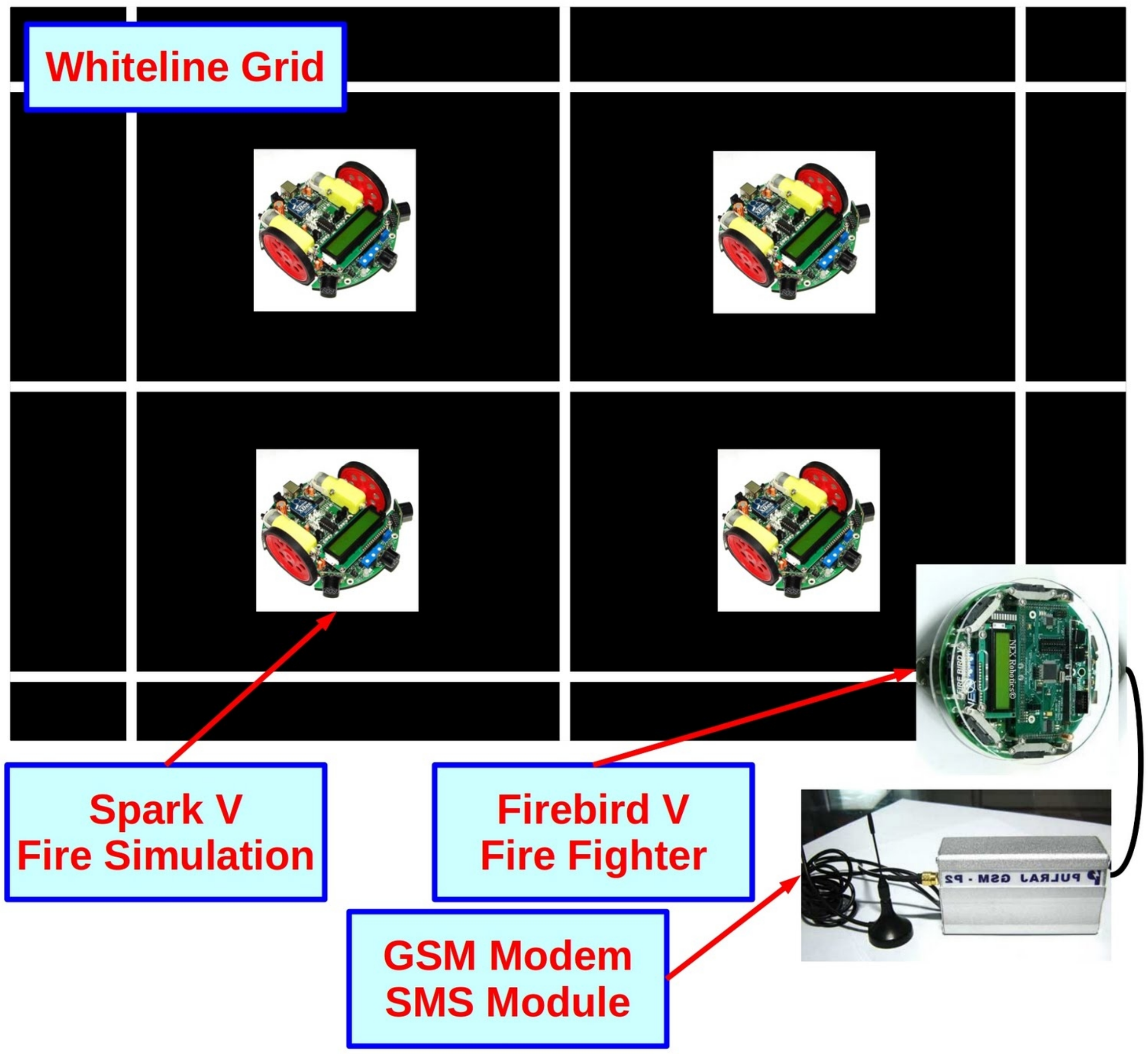


Figure 1 : Overall view of the system.

As shown in the figure 1. Sparks V bots simulating fire are kept on a white line grid. Firebird 5 is placed initially at fixed location. Firebird will go to all the nodes using white line and it is aware of physical location of all the nodes. GSM modem is connected with Firebird to send SMS.

Fire Simulation :

Fire simulation is done with the help of Zigbee communication. We implemented a Zigbee protocol in order to simulate the fire. The protocol is as follows. All the nodes are assigned unique software address. We call it node id. The packet shown in figure 2. First byte of the packet indicates type of the packet. Based on the type, length of the payload is decided. The possible types of the packet are FIRE_START, FIRE_SPREAD, FIRE_END, FIRE_INIT and TOPOLOGY.

TYPE	PAY_LOAD
1 Byte	Depends on TYPE

Figure 2 : General format of the fire simulation packet

For the packet types other than TOPOLOGY, we use 2 byte packet. The packet for the same is shown in figure 3.

TYPE	SOURCE
1 Byte	1 Byte

Figure 3 : 2 byte packet

Here the first byte indicates type of the packet and second byte is source of the packet. The recipient of the packet acts according to the type of the packet. FIRE_START indicates that fire has started on the node with specified node id. This packet in for fire fighter. When fire fighter receives this packet, it puts the source in the queue. The fire is indicated on the Spark V by displaying Fire on LCD. It increases with time as shown in figure 4 and 5.

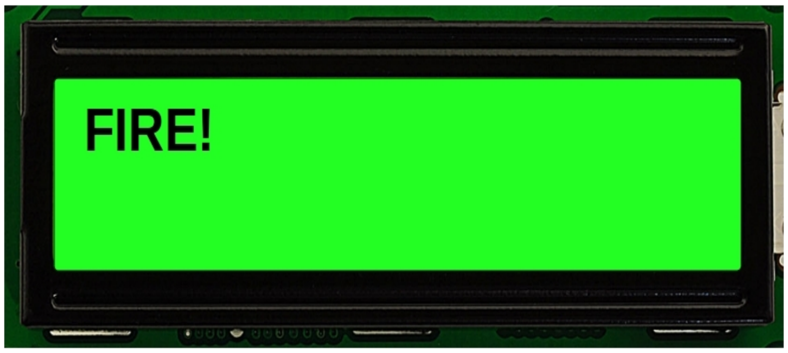


Figure 4 : Fire started on the bot.

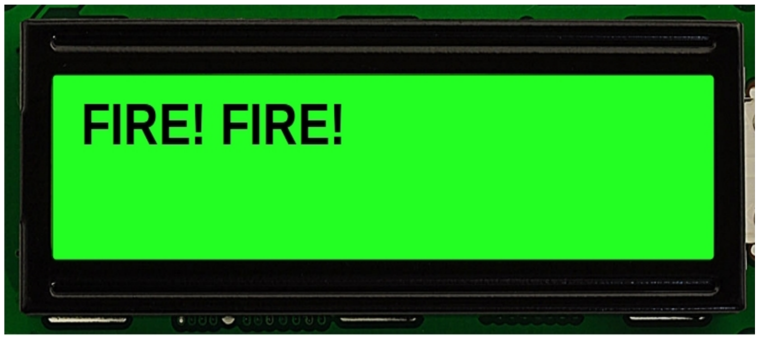


Figure 5 : Fire increased on the bot.

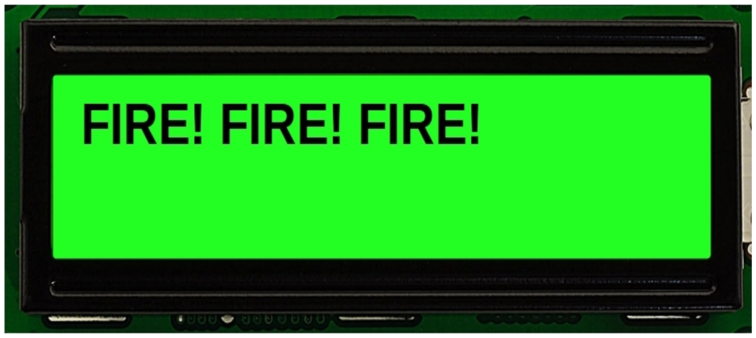


Figure 6 : Fire spreads from the bot.

FIRE_SPREAD indicates that the fire on the node with specified node id has become to much and now it is above to spread. This packet is intended for the nodes. Fire fighter simply ignores this type of packets. On receiving spread packet, Spark V checks whether this node is its neighbor of nor. If it is its neighbor, it also starts fire on it, else it does nothing. When fire spreads, FIRE! is displayed 3 times on the LCD as shown in figure 6.

When the fire fighter comes to a node on the fire, it detects the presence of fire fighter using front IR sensor and ends fire on it. It sends FIRE_END message to indicate that fire ended on it. This packet is also for the fire fighter. On receiving this packet, fire fighter will remove the node from its queue. When fire is extinguished, it displays NO FIRE! :-) on the LCD as shown in figure 7.

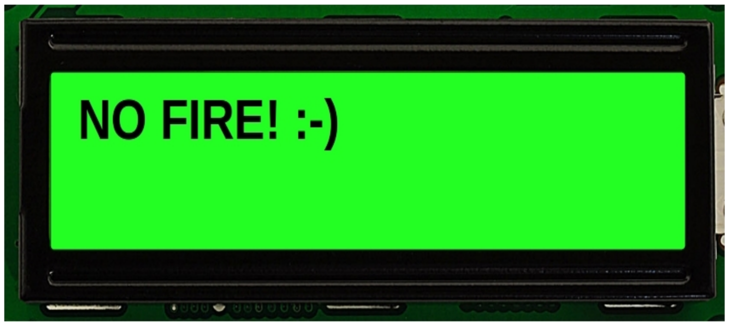


Figure 7 : Fire ended on the node.

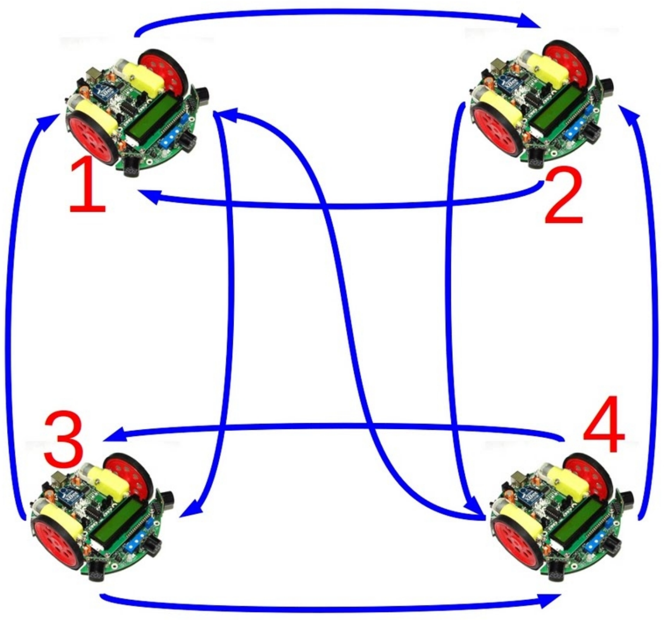
FIRE_INIT message is kept to externally start the fire on some bot. User can send this packet from PC connected with Zigbee adapter using X-CTU or any other software that communicate on serial port.

The topology of the Spark V can be dynamically configured irrespective of their physical location. The only required thing for two nodes to become neighbor is that they should be in communication range of each other. User can set the topology as required using ZIgbee adapter connected with PC. User need to send entire adjacency matrix of the topology. First user need to send TOPOLOGY byte to indicate that it is sending input for topology. The bots will enter in topology configuration mode and they will display TOPOLOGY on the LCD as shown in figure 8.

Figure 8 : Spark V in topology update mode.

Once the bot enters in topology mode, the user need to send the adjacency matrix row by row. Those messages are broadcast and all the nodes can be configured simultaneously. Here the edges of topology are directional to indicate the scenario where fire can spread from node a to b but it cannot spread from node b to a. This is possible on the wind direction. So this flexibility is kept to simulate such situation. figure 9 shows a sample topology and its adjacency matrix. This entire matrix needs to be sent.

In the Zigbee protocol, if the data collides, it backs up for some random time and then transmits next byte. On the Zigbee radio, we can transmit only one byte at a time, so two back to back bytes are considered as a packet. Now consider the following scenario. Node 1 sent one byte and it was sent successfully. Then node 2 and node 1 together sent byte which collided and garbled. After random back off, node 2 transmitted before node 1. In this case the second successful byte is from node 2. This entire packet becomes invalid but the recipients are not aware of it. To avoid this scenario, we intentionally back off before transmission. We give different back off to each node based on their node id so that it does not collide ever. Each node backs off 100*NODE_ID milliseconds before transmission. Two back to back bytes takes 50 ms to transmit with little delay between them. So the scenario mentioned above will never happen.



$$\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

Figure 9 : Sample topology and its adjacency matrix

Fire Extinguishing :

Firebird 5 acts as a fire fighter to extinguish the fire. It acts when it receives the FIRE_START packet. As soon as it receives first FIRE_START packet, it sends an SMS to the specified number reporting the fire. Fire fighter maintains a queue of the nodes on fire. It adds the node to the queue when it receives the FIRE_START packet. It then calculates the distance of each node which are on fire and goes to the node which is nearest from its current position. Once it starts moving to next destination, it goes there only. In midway if it receives the request, it will simply add it to the queue and process it when it is done extinguishing.

To extinguish the fire, it goes to the node and waits for one second. The node detects its presence using front IR sensor and ends the fire by sending FIRE_END. When the fire fighter receives this packet, it removes that node from the queue.

Fire righter follows white line to reach to the destination. Dimension of the grid are configured on the bot. The locations of the nodes on the topology are also configured on the fire fighter, i.e. it is aware of which node is placed where on the grid. It continuously maintains its position and direction. As soon as the fire comes to an end, it stops where it is. All the implementation is configurable. All the parameters are defined in the beginning of the file. Just changing them will change the system as required. The parameters are like number of nodes, grid dimensions etc.

5. Testing Strategy and Data

Testing of the system was done for the following test case and actual result is given below.

Fire simulation test : To test the fire simulation, we used 4 Spark V nodes with Zigbee radio. The fire was started on some node and then it spread according to the topology. We created all possible topologies using 4 nodes and it worked well for all the cases. We wanted to test fire simulation for more nodes but due to non availability of more Zigbee radios, we had to limit our test to 4 nodes only. We also started fire simultaneously on multiple nodes and it spread as expected.

Fire extinguishing test : To test the fire extinguishing, we brought Firebird in front of Spark V bot and it ended the fire. In this test case we also need to verify whether it catches fire again or not, and it caught fire again which is correct behavior.

Configuring the topology : We dynamically changed the topology and spread the fire. The fire was spread according to the topology. We changed many topologies without changing the code and every time fire spread as required.

SMS Module test : We connected GSM modem with Firebird 5 and which sent the SMS successfully when the GSM network was available. Due to non-availability of GSM network in the lab, it some time failed.

Movement on the grid : We moved the Firebird 5 on the 3x3 grid without putting any Spark V on the grid. We manually send all the FIRE_START and FIRE_END messages to the Firebird 5 and it moved accordingly. We sent some requests when it was moving. It successfully queued the requests and processed them. We sent it multiple requests to different distances and it went to the nearest node first. This testing proves that it moves on the arena as expected.

All system test : After all the test mentioned above, we tested the entire system. The system looked same as shown in figure 1. We placed 4 Spark V bots on the grid and placed 1 Firebird 5 bot. After that we configured the topology and initiated fire at some node. The fire spread as desired and fire fighter also went to the locations. We initiated fire at several places and tested with various topologies. It worked always.

6. Discussion of System

In this section we will discuss various aspects of the system development. We initially designed the communication protocol to simulate the fire. After that we implemented the fire simulation protocol on Spark V bots. We tried to work with esterel but basic send and receive of the packet did not work from esterel. We tried to make it work for couple of days but at the end, it did not work mostly because of programming. After that we moved to C.

After that we implemented the designed protocol on Spark V bots and tested it. It worked fine as per expectation. This protocol did not involved provision for topology configuration. We added support for that and allowed it to configure the topology as user wants to. Simultaneously we designed the movement of the fire fighter on the grid. We decided that we will use nearest node to extinguish the fire.

After successfully testing fire simulation we moved to fire fighter code. We first wrote a function to send SMS which sent SMSes successfully. We also wrote a function which takes source and destination on the grid points and moves from source to destination. We tested this function with many moves on the grid. In this case we faced some trouble with the Fierbird. The motors of the Firebird were not moving with equal speed even though we gave same speed. Because of that when we told it to turn 90 degrees, it used to turn more. We tried many angles but it never turned exactly 90 degree and because of that it was going off the white line. To over come this, we did not used degrees to turn 90 degree. Instead we started turning in one direction and stopped when we find white line. This function has advantage that the angle need not be 90 degrees or any fixed degrees. It can work on any angles. Using this function, we were able to move properly on the grid.

After that we integrated the code to find the nearest node on the grid. We fed the output of the nearest node function as an input to the grid movement function. This also worked as per the expectation. This brings to an end of development of the system. We then tested entire system together. In this case when Firebird goes near Spark V, IR sensors were not detect the presence of Firebird. The reason is that Firebird has some ground clearance where as IR sensor of the Spark V is very low in height. We places thermocol on the Firebird to remove the ground clearance and doing that, IR sensors detected presence of an object and extinguished the fire.

Overall everything worked as planned. We faced difficulties only with doing it in esterel and error in motors of Firebird 5.

7. Future Work

The work can be extended in the following ways. If the given arena is small enough, then single fire fighter can handle it but it will not be able to large areas. To over come this, the work can extended to use multiple fire fighters and they all synchronize the process of fire extinguishing with each other such that they all do not go to the same location at the same time.

At present, we assume all the nodes are equal and we go to the nearest node first to extinguish the fire. Instead of that, a priority map can be give to the fire fighter and it should extinguish the fire on the location having high priority first. E.g. in the case of inventory, we can divide the inventory according the cost of the material stored and we can assign high priority to the region with high cost. The fire fighter will go to that region first and extinguish the fire first. This method will minimize the loss.

Current implementation is done using C. Current implementation involves various functions which are called to achieve various tasks like send SMS, move on the grid, find nearest node on fire etc. Those functions can be easily converted to Esterel signals and the entire project can be implemented in esterel.

8. Conclusions

In this work, we simulated fire using Spark V bots. We designed Zigbee protocol which simulates the fire and implemented it. This protocol also involves communication with the fire fighter to inform it about the locations of the fire. We also implemented fire fighter using Firebird 5 bot. The fire fighter moves on the given arena and extinguishes the fire. We optimized task of fire extinguishing task by going to nearest possible node first. We also provided support for dynamic topology configuration among Spark V bots to simulate the fire. This allows to configure the topology irrespective of their physical locations. The code is written in such fashion that it should not be difficult to port it to esterel.

9. References

1. Spark V hardware manual, Nex robotics.
2. Spark V software manual, Nex robotics.
3. Firebird 5 hardware manual, Nex robotics.
4. Firebird 5 software manual, Nex robotics.
5. AT Command Guide, PCTEL.