Detection and Prevention of Forest Fires

Devina Sachin Dhuri  
Department of Computer Science  
ddhuri1@binghamton.edu

Sanved Tapkeer  
Department of Computer Science  
stapkee1@binghamton.edu

Tejas Kapadia  
Department of Computer Science  
tkapadi1@binghamton.edu

Abstract

Forest fires constitute a serious problem throughout the globe. Forest fires not only increase the carbon dioxide levels in the atmosphere, but also contribute to greenhouse effects. Each year, between 60,000 and 80,000 forest fires occur in US, destroying between 3 and 10 million hectares. It proposes a great risk to vegetation, wildlife, livestock and humans. We aim to provide a solution to these issues with our proposed system. We aim to wirelessly detect the presence of fires in forest areas and predict the next possible areas for the spread using learning algorithms. Further on, we also aim in the containment of the spread using viable solutions.

Keywords

Simulate, Prediction, Instantiation of fire, Unity3D, Console

1. Introduction

Forest fires are among the disasters that have multidimensional negative effects in social, economic and ecological matters [1]. The improvement of the level of modernization of forest fires monitoring using information and communication technologies has strategic significance for many countries where forest fires occur frequently. Compared with the traditional techniques of forest fires detection, wireless sensor networks (WSNs) technology is a very promising green technology for the future in detecting efficiently the forest fires. In our proposed method, we are aiming to detect the presence of fires using temperature and humidity sensors. The sensors selected were three DHT11 sensors and one TMP36 sensor. The ESP8266 was used as the MCU node via which we established communication between the sensors and the Personal Computers (PC). We designed a star/star-mesh network where the hub acted as the master and the sensors as the slave. The temperature readings were collected from the sensors using the node MCU which were then given to the PC’s. To simulate the fire spread Unity3D gaming engine and console-based simulation was used. After simulation, using two algorithms we predicted the spread of fire in the next areas. After the prediction we also attempted to show how the fire could be contained using Unity3D tool, particle systems. Further on, we also showed the accuracy of the algorithms used for prediction, cost involved and research on the feasibility of this project.

2. Design

2.1 Hardware Specifications

We used the DHT11 sensor to get the temperature readings even at a distance of 20m with minimal loss of information. The use of TMP36 sensors was to get maximum information even above 100°C. The main reason for using two different sensors, digital and analog, was to get readings even above a certain threshold. The DHT11 sensors have a limitation of the temperature recordings which can be solved using the TMP36 sensor.

2.1.1 DHT11 Sensor: DHT11 output marks a digital signal. It utilizes exclusive digital-signal-collecting-technique and HDTY sensing technology, reasurring its responsibility, accuracy and stability. Its sensing part is connected to a 8-bit single-chip microcontroller. Every detector of this model is temperature remunerated and mark in correct activity chamber also the calibration-coefficient is saved in OTP memory [6]. Small size & low consumption & long transmission distance(20m) change DHT11 to be suited all told styles of small to day to day life application occasions. Single row prepackaged with four pins, creating the association terribly convenient.

Figure 1: DHT 11 Sensor to record temperature and humidity

The features of DHT 11 are as follows:

1. Operating Voltage: 3.5V to 5.5V
2. Operating current: 0.3mA (measuring) 60μA (standby)
3. Output: Serial data
4. Temperature Range: 0°C to 50°C
5. Humidity Range: 20% to 90%
6. Resolution: Temperature and Humidity both are 16-bit
7. Accuracy: ±1°C and ±1%

2.1.2 TMP36 Sensor:

The TMP36 is a low voltage analog temperature sensor. The sensor provides readings that are precise to centigrade level which means that the graph of voltage to Celsius is linear. The sensor does not require any special or extra component to provide a typical accuracy and efficiency at higher temperatures over 100°C [5]. The temperature also provides extra readings capabilities for cold temperature over greater distance without loss of much data.

Figure 2: TMP 36 Sensor to record temperature and humidity. This is an analog sensor

The features of TMP 36 are as follows:

1. Low voltage operation (2.7 V to 5.5 V)
2. Calibrated directly in °C
3. 10 mV/°C scale factor (20 mV/°C on TMP37)
4. ±2°C accuracy over temperature (typ)
5. ±0.5°C linearity (typ) 
6. Stable with large capacitive loads 
7. Specified –40°C to +125°C, operation to +150°C 
8. Less than 50 μA quiescent current 
9. Shutdown current 0.5 μA max 
10. Low self-heating 

2.1.3 ESP8266 Node MCU: The ESP8266 is a development board built with Tensilica Xtensa® 32-bit LX106 RISC microprocessor. The Node MCU operates at 80 to 160 MHz with an adjustable clock frequency and supports RTOS [4]. It has a flash memory of 4Mb and RAM of 128Kb. It is integrated with 802.11b/g/n HT40 Wi-Fi transceiver, which allows it to create its own network rather than just interacting with another network and connect to Wi-Fi.

Figure 3: ESP8266 Node MCU to control the temperature and humidity of the sensors.

The features of Node MCU are as follows:

1. Microcontroller: Tensilica 32-bit RISC CPU Xtensa LX106 
2. Operating Voltage: 3.3V 
3. Input Voltage: 7-12V 
4. Digital I/O Pins (DIO): 16 
5. Analog Input Pins (ADC): 1 
6. UARTs: 1 
7. SPIs: 1 
8. I2Cs: 1 
9. Flash Memory: 4 MB 
10. SRAM: 64 KB 
11. Clock Speed: 80 MHz 

2.2 Hardware Setup 

The ESP8266 has 9 Digital input pins and 1 analog input pins, it also consists of 3 voltage output pins and 4 GND terminals. It also has a digital output pin and several other SD pins which we don’t require for this connection [3]. We have used 4 Temperature and humidity sensors mainly classified as: 3 DHT11 sensors [2], 1 TMP36 analog temperature sensor. As shown in Figure 4a, the connection are as follows:

1) The Data port of 1-DHT sensor is connected to the digital pin D1 of Node MCU. 
2) The Data port of 2-DHT sensor is connected to the digital pin D2 of Node MCU. 
3) The Data port of 3-DHT sensor is connected to the digital pin D3 of Node MCU. 
4) The VCC port of 1-DHT, 2-DHT, 3-DHT is connected to the 3V3 pin of the ESP8266. 
5) The GND port of the 1-DHT, 2-DHT, 3-DHT is connected to the GND pin of the ESP8266. 
6) The Data port of the TMP36 analog sensor is connected to analog pin A0 of Node MCU. 
7) The VCC port of the TMP36 analog sensor is connected to 3V3 of Node MCU. 

8) The GND port of the TMP36 analog sensor is connected to GND of Node MCU.

Figure 4a: Hardware connections for sensing with 3 DHT 11 sensors and 1 analog sensor.

2.3 System Design 

As shown in the Figure 4b, the sensors start sensing the temperature and recording it through the ESP. This data is then processed. Every temperature reading (T) obtained is then compared to the threshold temperature (Th), if \( T > Th \), then we check if there is a fire present. The reason for these two conditions is to take into consideration the hot climate. The Th is set to a value of 90 degrees. This is the base condition, once this is crossed another check of 120 degrees is performed to ensure accuracy. If neither of these two checks work, then the system keeps looping to record other readings, but when this limit is crossed, a signal is sent to the PC with the node/ sensor number that detected the fire. Based on this number, a fire is initiated in Unity and console, and we then predict the next possible area for the fire. If there is a fire already present at the predicted areas, the next are is predicted and sprinklers are activated accordingly to contain the fire. This system keeps looping this way till manually stopped.

Figure 4b: System design for the sensing, simulation and prediction of fire spread.

2.4 Software Setup 

In order to represent the fire nodes, the water sprinklers, we have used Unity3D’s particle system. To show that a fire has been initiated, we use an array. Every position that a fire gets initiated gets added in that array. Given to choose from 8 options, an index is randomly chosen each time a fire must be initiated. To indicate if or not the fire is initiated at a position we use another array to store a string variable. To detect fire nodes colliding, and the temperature increase, we use colliders around Unity objects.

3. Implementation 

The implementation of this project is done in three parts: the sensing algorithm to record the varying temperatures, the simulation algorithm to simulate the fire spread and the prediction algorithm to predict the next areas possible to be hit with the spread and measures to contain the fire.
3.1 Sensing Algorithm
The ESP8266 uses star topology for taking readings from all the DHT sensors and ADC sensors so that each sensor communicates with the Node MCU individually and there is no interference or data loss happening during transmission and reception. The utilization of DHT sensors requires importing the DHT libraries, once done we will initialize the nodes of ESP8266 Node MCU then assign node values to each DHT sensor. Once done with the digital sensors we’ll start with the analog sensor i.e. TMP36 sensor. Initializing the TMP36 sensor we’ll assign an ADC port to the analog sensor. To start taking the readings we will start the serial baud rate to 9600. This allows the Node MCU to read the values from digital and analog sensors. On the other end we will initialize a Boolean value and a variable with the name ‘f(node_no.)’ that checks whether the temperature has gone above the threshold or not. We’ll initialize a loop that checks whether the temperature from the sensors have gone above the threshold 95F and 120F if so, the Boolean value is updated as 2. Now, if it has just crossed 95F the Boolean value is updated with a value ‘True’ and if not, the Boolean value is updated as ‘False’. Depending on the state of value inside the Boolean the temperature is printed. This process could be monitored wirelessly a connection port is established between device and computer using HTTP. The readings taken by the sensor is saved in the host machine as a ‘.txt’ file as shown below.

![Figure 5: Sensing algorithm to record the temperature in an environment.](image)

3.2 Simulation Algorithm

As shown in Figure 6, as the ESP writes the sensor number to a text file, the Unity and console-based code reads the sensor number written. This sensor number written is the area where the sensor detected a fire. Once this number is acquired, based on that number the position is read in terms on x, y, z. A fire is initiated at this position and is saved in an array to be considered as already occurred fire positions. A delay is added based on how fast or slow we want to simulate the spread of fire. We then choose a random value between (0-8). This value indicates the next position to simulate the fire. With the random value we also choose a random index from the list of fires. Before we choose this position to initiate a fire, we check if there is a fire already present. If yes, we choose another value, if no, we initiate a fire, we keep looping to this till it collides with sensors to trigger it. Once the sensor is triggered, we choose a prediction algorithm (1-Boundary Detection, 2-Lane Detection). Based on the input for the algorithm, we simulate the algorithm working and initiate sprinklers to contain the fire. This simulation keeps looping till the user ends it.

3.3 Prediction Algorithm

There are two algorithms chosen to predict the fire spread once we get a sensor triggered. They are, Boundary Detection and Lane Detection.

3.3.1 Boundary Detection Algorithm: In this algorithm, given a position where the fire is initiated, the next possible position is selected by choosing one of the 8 value that surround the current position. As shown in the figure 7, if the center cell x, y, is taken to be the position of the fire initiated, any one of the 8 surrounding cells is considered next as the possible target.

![Figure 7: Boundary Detection Algorithm with the center as seed position.](image)

3.3.1 Lane Detection Algorithm: In this algorithm, unlike the previous one, we use lines to separate each cell and consider them as lanes. Using transforms, we detect lines and the output is a parametric description of the lines. In this, when the fire crosses a lane, the sensor adjacent to that is triggered, as shown below. Say the fire is initiated at the center cell x, y, it then crosses the left side on the cell, thus triggering the sensors in that area (highlighted by yellow).

![Figure 8: Lane Detection Algorithm with the center as seed position.](image)

4. Results

4.1. Initial Setup

As shown in Figure 9, the box is the forest that is considered, and the four corners are marked with 4 sensors.

![Figure 9: Initial setup for the experiment](image)

4.2. Simulation and Execution
4.2.1 Step 1: Figure 10 shows the initial stage when the forest bed is lit. Since there is no considerable fire yet, the values to the bottom right of the image all indicate 0. Note that the output is in the format; temperature, sensor number, value. The upper left and right simulation show the empty fields, and nothing is caught on fire yet.

Figure 10: Start of fire and simulation.

4.2.2 Step 2: Figure 11 shows that when the sensor is triggered, the simulation also reads the number and starts the fire at that location. We can see in the temperature recordings the value changing to 2, indicating a change in temperature.

Figure 11: Sensor 15 (1) getting triggered.

4.2.3 Step 3: Figure 12 shows that when the sensor is triggered, the prediction algorithm starts. Here sensor 21 is predicted to be the next area to be affected by both unity and the console.

Figure 12: Fire spread, and prediction of next possible spread.

4.2.4 Step 4: Figure 13 shows that when the sensor is triggered, the prediction algorithm starts. Here sensor 21 and 22 is predicted to be the next area to be affected by both unity and the console. The temperature reading following shows that sensor 21 started sensing presence of heat.

Figure 13: Correct prediction and temperature recording.

4.2.5 Step 5: Figure 14 shows the final stage all the sensors are triggered and the fire spread. The prediction worked correctly. We see that as the fire spreads in the model, due to the containment algorithm implemented, the fire doesn’t spread in Unity.

Figure 14: Spread detection and containment.

4.3. The Aftermath

Figure 15 shows the final stage when the forest bed is completely burnt. This image depicts the real-world forest scenario.

Figure 15: Result of the forest bed after the fire.

5. Evaluation

5.1 Accuracy

We plotted a graph for boundary detection algorithm and lane detection algorithm based on the number of sensors that were predicted to be the next plausible places for the fire spread and on the number of sensors that caught fire. As we see in Figure 16, boundary detection algorithm gives us an accuracy of 30 – 40%, whereas, lane detection algorithm gave an average accuracy of 80%.

Figure 16: Accuracy comparison between the prediction algorithms.

5.2 Temperature Readings
5.3 Analysis

According to the recent Amazon fires, 3500 sq mi of area was burnt creating a lot of loss. If this software was deployed, each mile would have 4 TMP36 sensors. 240 sensors are deployed within a range of 60 miles, thus total sensors in the area would be 57600. Each sensor costs about $1.5, thus total of $86400. For this same area, 11520 Node MCU's are needed, total cost is $23040. Thus, the total cost of sensors, MCU’s and other extra costing would sum up to $120000. Using the containment algorithm, if the fire was a slow spread, off the total 3500 sq mi, 2100 sq mi (9/15) area could be saved from burning. Whereas, if the spread was at a faster rate, 1633.333 sq mi (7/15) area would be saved. According to the latest reports, the government issued $22.2 million to combat this situation. Using this technique, for the fast fire $10.3 million and for slow spreads, $13.3 million would be saved.

6. Conclusion

To conclude, this shows that having enough information it is possible to sense the fire and predict the next areas that could be affected by the fire, thus contain it. Based on the evaluation, lane detection algorithm gives us better results when it comes to prediction of fire spread. As we saw in the results, if the fire was to spread, it would result in a large-scale destruction. Bus using this approach, we could easily contain the fire and not let it spread to a large extend. Thus saving 7/15th to 9/15th part of the forest and up to $10 - $13 million.

Acknowledgment

We are immensely grateful to our Professor Dr. Mo Sha and our TA’s Di Mu, and Junyang Shi for their comments on an earlier version of the project. This couldn’t have been achieved without their persistent support and feedbacks.

References