https://www.youtube.com/watch?v=eFTLKWw542g

Fire Early Warning Alarm (FEWA)

Low-range, low-power consumption

Abstract

Fires are a major cause of forest degradation and have wide ranging adverse ecological, economic and social impacts, including: loss of valuable timber resources, degradation of catchment areas, loss of biodiversity and extinction of plants and animals, vast territories are being destroyed and a contributor to that might be the not-so-fast response that firefighters have. Not necessarily their fault, but the delayed alert.

Introduction

This report will try to present an alternative solution to firewatch lookouts, one that is easily scalable and adaptable to forests of various sizes. Furthermore, the approach taken and techniques used and how they are applied in the design and development of FEWA (Fire Early Warning Alarm) will be presented.

Purpose of the project

The FEWA units are a system of heat sensors mounted on trees in the forest. They'll send signals in case of a fire. Whenever the environmental conditions are met(high temperature and flame is detected) it sends a signal to the nearest FEWA unit. This way the signal is able to travel long distances through a combination of low-range & low-power increments, ensuring the signal is strong and received by the HQ and people, who are able to take the appropriate actions.

Specifications of product:

- Two sensors mounted
- Low range (around 80-100 meters radius for project purposes, more for real-life solution)
- Low power consumption
- Mesh network topology
- System prevents loose of message
- System prevents looping of the signal
- End-point for dispatching message (one for project purposes, more for real-life solution)
- Battery powered (remote)
- Does not require maintenance for a long period
- Units have GPS coordinates to point to fire location
- Units can be placed randomly in the forest

Methods and tools

Architecture of the product

We designed our product, using a method from Claire Rowland's book(Claire Rowland - "Designing connected products"), to get an overview of our components that creates the FEWA product. (Explained further under block diagram).

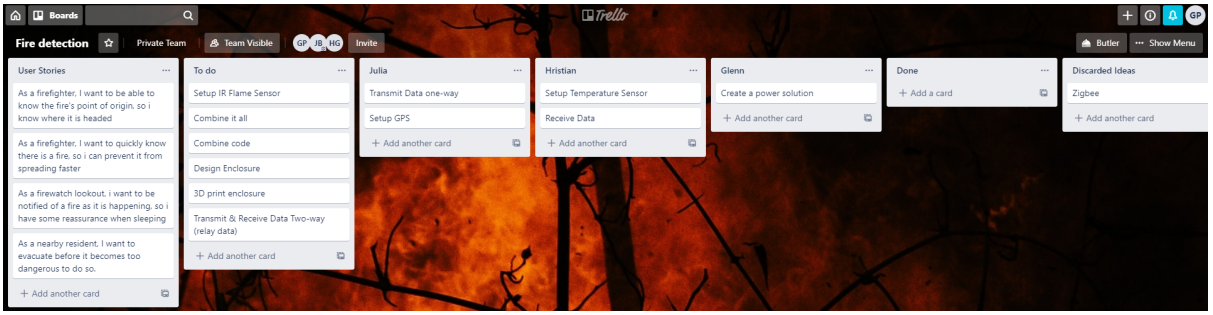
As we got further into the process of building the product, we encountered a problem regarding the availability of the parts we had originally planned to have, which is where the architecture drawing is helpful.

The communication technique will be using a mesh system, which consists of multiple points all being able to forward the data to the endpoint. This ensures we can cover as large an area as possible, because the units can be set down in a chain, so the unit furthest away from the endpoint still is able to send data to the endpoint, because the data is going through every unit and being forwarded.

C/C++ will be the main programming language, since the libraries for the different components are more readily available for the Ardunio.

Trello

To assist in the production and development of the FEWA product, a Trello board was created with user stories based on what a “customer” in the final product would like. Tasks were assigned to the members, in order to achieve a viable product and meet the technical specifications.



Google Drive/Docs

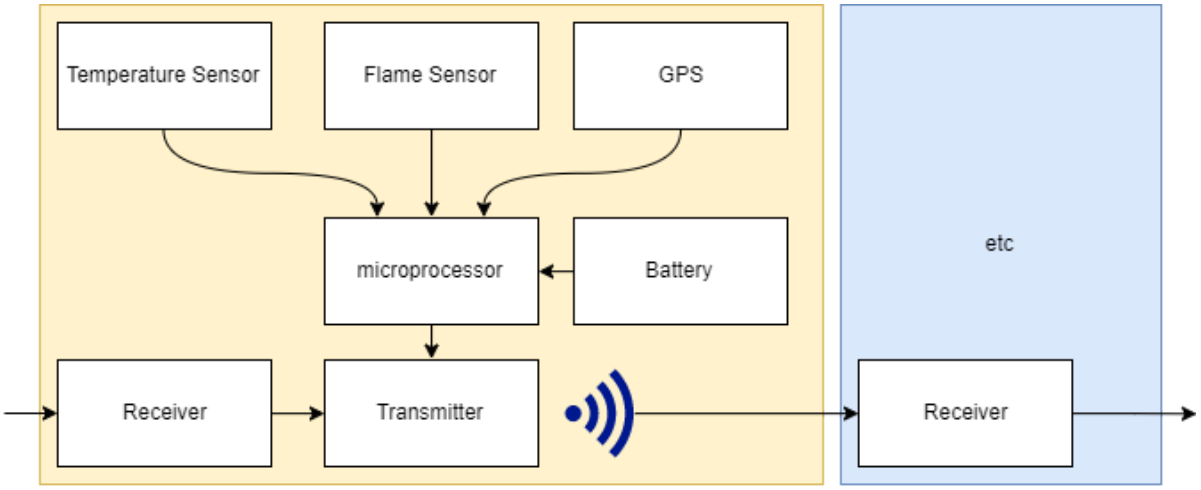
We agreed upon using Google Drive to share files and important documents. After creating a directory to which each group member had access, we began to store all of our documents, diagrams and other hand-ins that were necessary during the project creation process in that folder.
The report was written using Google Docs and other documentation, so that the file could be easily accessible to all and so that each member could add and edit the text simultaneously, to avoid transferring a locally stored document.

LPD433

When designing product with radio frequency transmitter the regulation must be taken into consideration.
In most countries, including Denmark, specific devices can work freely on frequency of 443 MHz as LPD433.
FEWA falls under those specifications, where it’s operated on low power, doesn’t transmit continuously and doesn’t transmit voice.

The product

Block Diagram

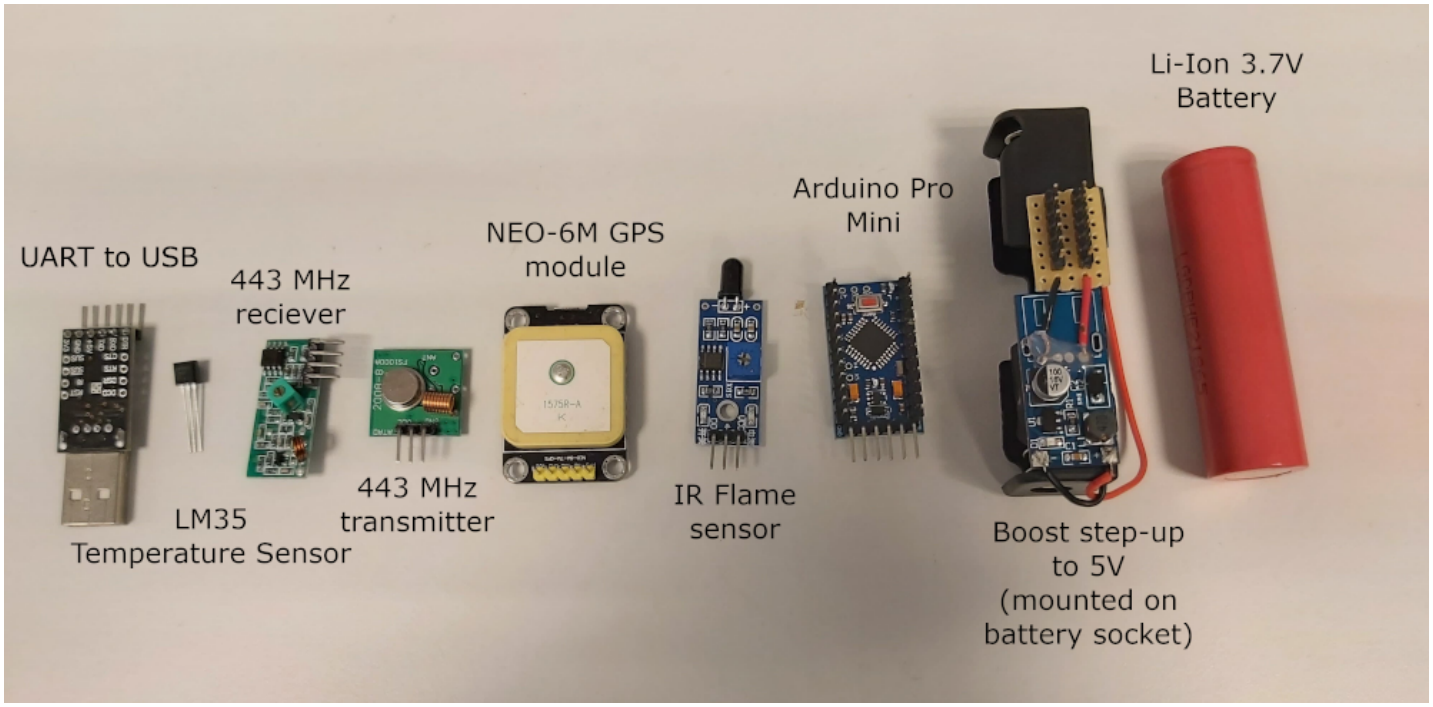


The single unit consists of a microprocessor with 2 sensors and a location tracker, all of which is receiving power from a long-lasting and rechargeable power source. A unit is then completed with some Tx(Transmitter), RX(Receiver) attached so the data can be transferred to another unit.

This setup is then repeated for every unit, and all of them should be able to communicate with each other.

Every unit can receive data, and transmit it again as a repeated signal.
Any of the units can work as the endpoint, but instead of transmitting the signal further, it is instead displayed on a computer.

Hardware



Arduino Pro Mini

For our project we decided to base the setup on Arduino Pro Mini, to handle the logistics of the code and sensors. The main advantage was the size of the board - it's much smaller than the widely used Arduino UNO and other popular microcontrollers. This model is also cheaper than the Arduino UNO, making it easier to acquire multiple units to scatter around vast areas. The purpose of the product is to stay up in the forest, with a minimalistic impact on the scenic view and not disturb the local wildlife. Arduino Pro Mini was designed with the idea of being permanently embedded in a project. As the last advantage it would be the possibility of powering it with 3.3 volts, that was abandoned in the end, due to change of other components.

Neo-6M GPS module

U-blox NEO-6M is a high-performance GPS module with a ceramic patch antenna and an on-board memory chip (https://www.u-blox.com/sites/default/files/products/documents/NEO-6_DataSheet_(GPS.G6-HW-09005).pdf). It gets GPS coordinates of single unit to establish where the fire started, as the signal travels through many units. It also allows us to place units randomly, without worrying about dispatching signal without knowledge of where it came from.

IR Flame sensor

To recognize forest fire we're using IR Flame sensor that can detect infrared light with a wavelength ranging from 700nm to 1000nm (https://arduinomodules.info/ky-026-flame-sensor-module/).The far-infrared flame probe converts the light detected in the form of infrared light into current changes. LM393 comparator is conditioning signal that can be easily read as digital output.

Temperature sensor

To support IR Flame sensor (as it may not see the flame if it's too far away, or the sensor is pointed in another direction) we also added the LM35 temperature sensor. (http://www.ti.com/lit/ds/symlink/lm35.pdf)

Power supply

To provide the necessary power to the system a LG LGDBHE21865 Lithium-Ion battery is used. It give out 3.7 volts (https://www.powerstream.com/p/LG-ICR18650HE2-REV0.pdf) that are later step-up with a converter. To charge the batteries in between tests we're using the battery chargers, we made ourselves as part of electronics course.

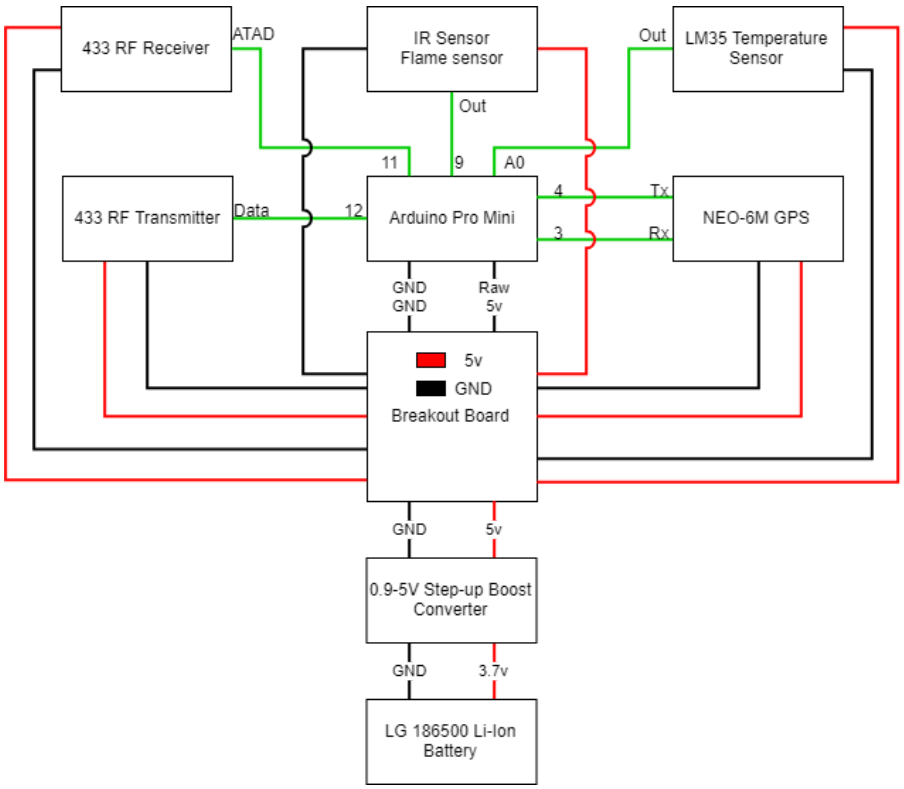
433 Mhz RF module

To communicate between units we established radio-frequency protocol, where both transmitter and receiver are included in the setup of one unit. They're working on frequency of 443 MHz and have a low range. These modules work perfectly with Arduino RadioHead library, that made it easy for us to work with. Transmitting signal that way can ensure as, that the message will not get lost, as the signal is sent a few times, to multiple units. It also allows us to manipulate our mesh network, making sure we're able to avoid looping the signal. (http://www.mantech.co.za/Datasheets/Products/433Mhz_RF-TX&RX.pdf)

0.9-5v Boost step-up

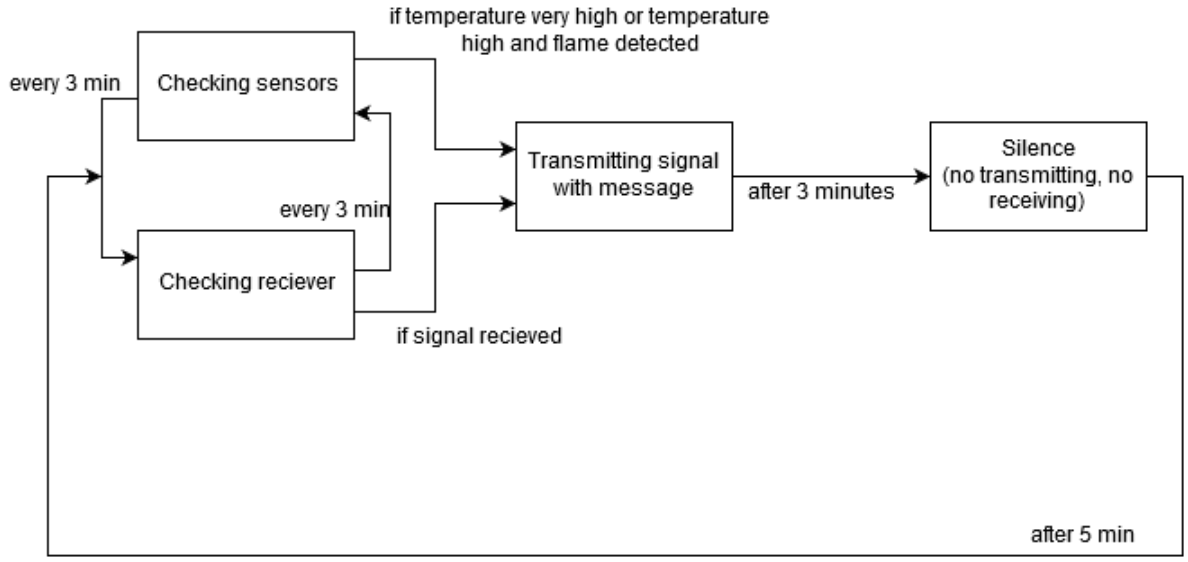
In the beginning our setup was supposed to be powered by 3.7V battery, but as we changed the transmitter and receiver we had to also establish a Vcc of 5V, as they required higher input voltage. That's why we implemented a DC-DC 0.9-5V step-up converter, mounted on the side of battery socket. We ordered it as a DC-DC USB converter, but simply modified it to fit our needs by removing unnecessary USB pins. (http://www.haoyuelectronics.com/Attachment/CE8301/CE8301%20Module.zip)

Wiring diagram



At the center of our design stands an arduino pro mini where all the sensors will be attached. A single 5v power supply with a breakout board will provide all the components with power.

Code state diagram

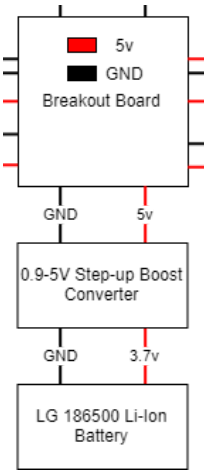


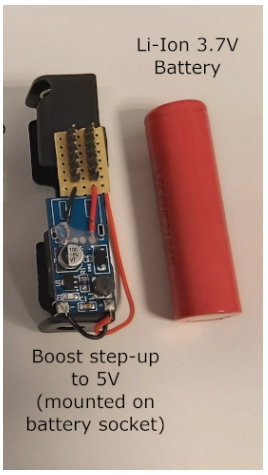
The code state diagram gives an abstract description of how the FEWA system behaves.
Every FEWA unit would function like this, with the exception being the end-point.

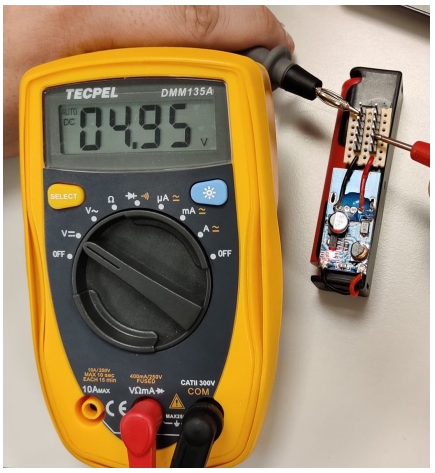
(Explained in detail in the code section)

Testing

A crucial point of the tests was to see if every component functions as expected, and if the accuracy was within the specifications for the project. The first component to be tested was the power delivery to all the other parts. A simple 0.9v-5v usb step up module was retrofitted with a small breakout board instead of the USB connection, and then glued onto a battery socket.

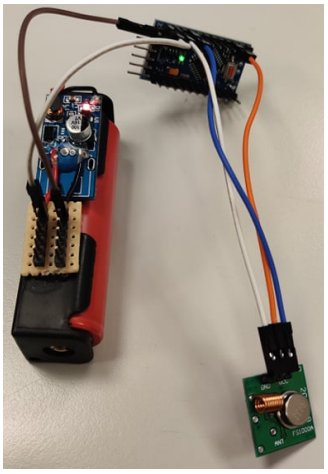


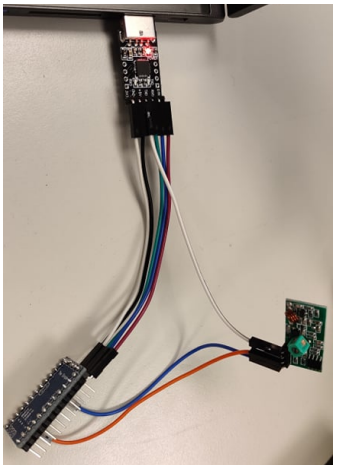




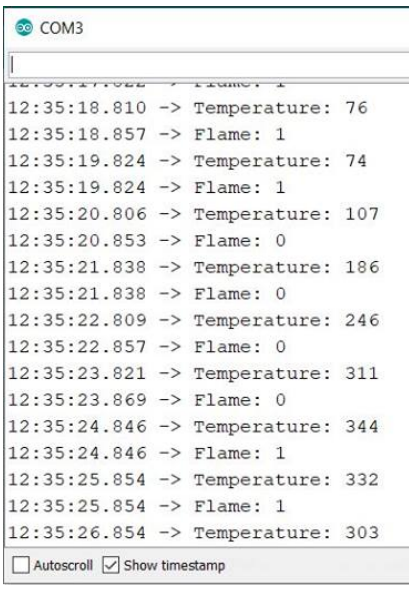
As observed, the output of the breakout board is at 4.95 volts which is just below the recommended for the receiver, but enough for all other components. The receiver needs around 5 volts whereas rest of the components are content with only 3.3 volts.

Next up was the testing of the transmissions with simple messages on the 433 mhz RF band. The transmitter was connected to the Arduino Pro Mini via pin 12, and the Arduino was getting power from the battery.
The receiving end was connected to the Arduino via pin 11, and receiving power from the UART module that we used to establish a serial connection from the arduino to the computer.

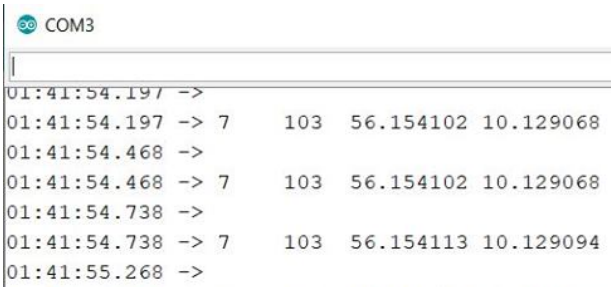




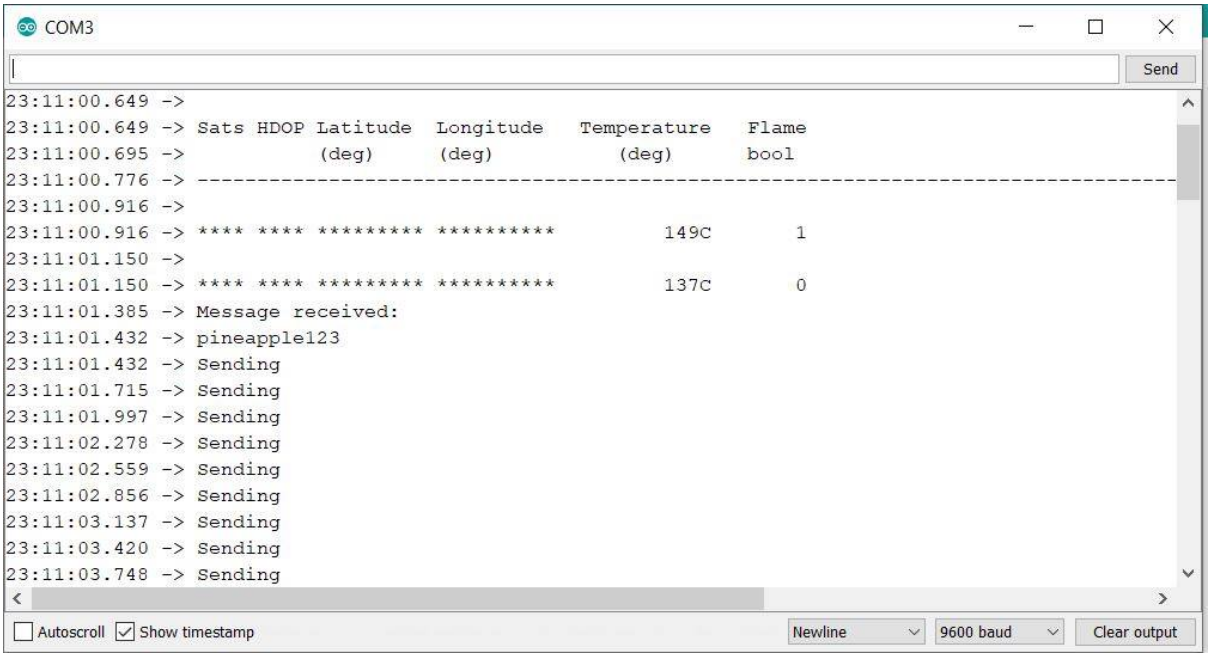
This leaves the GPS and 2 sensors to be tested, the IR flame sensor has a digital output, so when there is a flame value of 1 means that there is no fire, and a 0 means that there is a fire. The temperature is also getting measured by the LM35, during testing the value wasn't remapped to anything, so during testing the sensor is outputting values from 0 to 1023. Turning on a lighter made the Flame switch to 0, and then the temperature started rising, until the flame value switched back to a 1 and the temperature started dropping back.



And the last component to be tested is the GPS device, to get this working the component had to be outside with a clear view of the sky for a prolonged period until it could connect to enough satellites to trilaterate a position.The image below indicates that during this test, the unit connected to 7 satellites.



An aspect of the final product was to see if FEWA was able to relay unaltered data to another FEWA unit, and if the data being sent was still intact. A single FEWA unit was connected to a computer with a serial connection using a UART module. To test this, the units were put in a line, few meters apart and a fire was “started” at the point furthest away from the computer.



Here you can observe a FEWA unit change mode after having received a seemingly random message.

Solutions in product

Powering

Measuring the current of every component in a FEWA unit, results in around 200 mA of current, this is conveniently is the max output of our 0.9-5v boost module, this also means that the units might not be running in tiptop condition, as those 200 mA was the average, peak current could be considerably higher. Further development of FEWA, would change the components out for some less power hungry components, to lengthen the battery life and an integrated sleep mode for the units to conserve even more power, as the current solution with our LG batteries, only have around 12.5 hours of runtime, as they have a capacity of 2500mAh.

$$2500\ mAh\ /\ 200\ mA\ = 12.5\ h$$

12.5 hours is not within the specifications we've put out for the product, as 200 mA is also unexpectedly high for those components.

The life expectancy of FEWA could be greatly increased with the introduction of small solar panels, a 6v 2w solar panel has at peak conditions an output of 330 mA, which is enough to power FEWA, and charge the battery at the same time. This is not enough for FEWA to run infinitely, as it would take 19 hours to charge the battery. The idea will still stand if everything else was optimised and the current was reduced

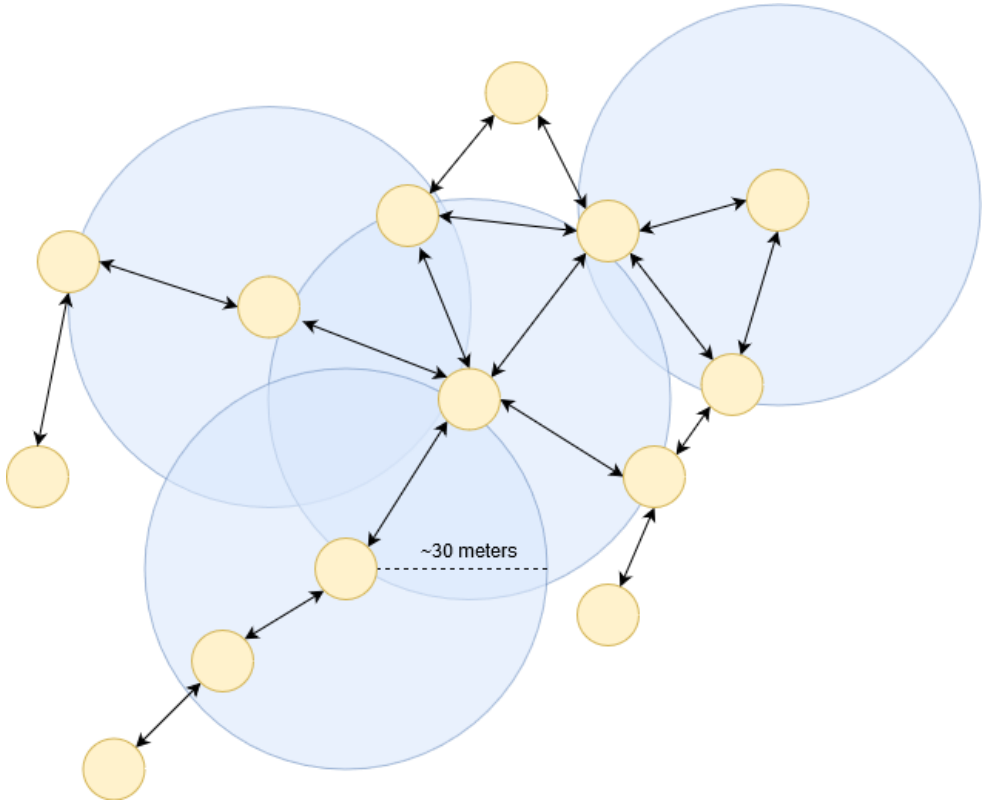
to respectable levels.
(https://www.verical.com/datasheet/adafruit-solar-panels-200-3430007.pdf)

Sensors

To recognize when a fire starts in the forest we decided to mount two sensors on our unit. One of them is a IR flame sensor that detects infrared light with a wavelength ranging from 700nm to 1000nm. Whenever flame is detected the sensor sets pin 9 on Arduino Pro Mini high.
To prevent any false readings we also implemented LM35 temperature sensor that monitors changes in ambient temperature.
Average temperature in forest ranges from 10 to 25 degrees. As a team we agreed that if a flame is detected and temperature needs to be around 50 degrees Celsius and flame sensor has to recognize fire in order for transmitter to start sending signal out.
In case flame detector is pointed in another direction it can't recognize any flame, we added another trigger to alarm when the temperature is very high (around 90 degrees Celsius). Under no other circumstances other than forest fires, the temperature would rise to this degree, but in this case it's most likely that FEWA unit would be impossible to retrieve and use again.

Mesh Network

One of the results of establishing radio communication between the units was the creation of a mesh network where a single transmitter is able to send signals to all units within its range.



The transmitter we implemented in our project has a range of around 30 meters, however with antenna mounted it can extend to 200 meters according to the datasheet. For proof of concept we used the minimal range.

Prevention of message loss or looping

As the signal is containing the GPS coordinates of the first unit that detected fire, it was necessary to come up with a system that will assure the relaying of the message to an end-point where it's dispatched. Whenever the alarm is triggered the microprocessor get the GPS coordinates and start the transmitter for around 1 minute. It sends the signal over 100 times, what means every unit in range has a big chance of receiving at least once. It's enough to trigger the relay and the signal is once again sent over 100 times.

As every unit is using a standalone transmitter and receiver, forwarding the signal only forward is complex and unnecessary, instead of circular radius, there's a chance that a unit which already relayed the message will receive it again. That looping can cause the system to get stuck on one signal forever. In perfect solution to this could have been resolved with a transceiver that has built-in functionality that prevents looping of messages internally. With standalone receiver we had to solve this by setting the devices to sleep mode after it's done transmitting. Since unit is not "listening" for around 2 minutes, the signal already passed from its range and won't be received again.

Amplitude Shift Keying

In order to send GPS coordinates through we had to implement a way of putting data into a wave signal transmitter is generating. Amplitude Shift Keying (ASK) was a perfect fit for this project and we also found an Arduino library *RadioHead written by Mike McCauley for Airspayce* that already had it programmed.
ASK refers to a type of amplitude modulation that assigns bit values to discrete amplitude levels. In our case the values were only binary, that means it's digital "1" - the carrier wave is at full strength, or digital "0" - the carrier wave is completely turned off.
A code handling the receiver is also using the same library, meaning it can easily decode the message and display it as a readable string.

GPS coordinates

A discussion took place to determine whether the setup needs a GPS module or not. On one side it was an expensive and highly power consuming component, on the other hand the setup would be much easier to work with, especially as a real life product, where there have to be over 100 FEWA units scattered around.

End-Point

For dispatching the message we decided to make a dedicated device with just a receiver. It would be placed in a firewatch lookout and have an alarm system to notify people working there, that they have to call in for help from the fire department.
The alarm can be many different solutions, like sound or phone notification. For our proof of concept we have set of five LEDs that light up whenever there's a signal.
As Arduino at the end-point is constantly connected to laptop through UART we can open a serial monitor and observe the message that we receive. There are the coordinates of the unit that triggered.

Problems encountered

Working with XBee:

Following the specifications of the product and after researching, a consensus was reached on what components are necessary. Following suggestions from our teacher we decided to work with Zigbee RF protocol - an IEEE 802.15.4-based

specification for a suite of high-level communication protocols used to create personal area networks with small, low-power digital radios. It would make it easier for us to create a simple mesh network and manage the message going with the signal.
XBee 3 Pro module seemed like the perfect fit for our project, meeting most of our specifications - the range of the signal is up to 3,2 kilometers, it's a transmitter and a receiver, small in size and can be programmed with MicroPython, what would eliminate the need for additional microprocessor.
Datasheet:
https://www.digi.com/resources/documentation/digidocs/pdfs/90001543.pdf
The disadvantages we've encountered were that we've never worked with similar technology. In our idea it was expensive to develop and we felt really unsure while purchasing that modules, as we didn't know how it works. Additionally, our order was mixed up and we received only one module, while we needed at least three to work with.

While researching the parts we came across the Xbee Explorer Regulated, a board that would make it possible for us to regulate the power flowing into Xbee module and give access to programmable pins. However, not only have we received only one of these, while needing at least three, but also encountered some troubles while trying to access the XBee module.

Following the guides we found online we tried programming the chip connecting it through UART to the laptop. We used XCTU software (special software for programming XBee modules) to access MicroPython, but all we discovered was that the boards comes with no firmware.

To avoid generating more cost and waiting for more parts ordered from the internet, we decided to change to components that were cheap and available for us to get the next day.

Power method

At first the initial component that was laid out to provide power for the circuit was a small flat 3.7 Li-Ion pack with a built-in battery management system, but due to unforeseen circumstances those were discovered to be unavailable a bit too late, this meant a plan B had to be put in motion.

As explained in the hardware section the backup plan was a LG 18650 battery.

Providing power to every component with the 0.9v-5v boost module did not work as expected. It was done to satisfy the power needs of the 433 mhz receiver, but as the

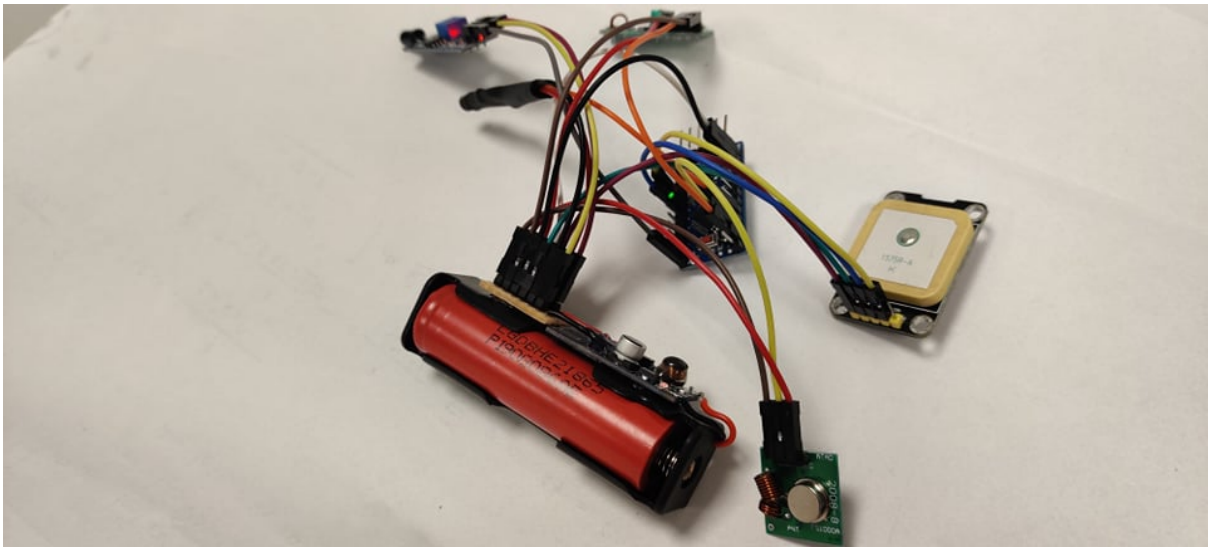
testing showed, the module never reached 5 volts, it boosted up to a voltage of 4.95. Initially we discarded this as being ok because datasheets claimed it could take between 3.3 volts to 5 volts.

Further testing made it clear that the receiver slowed down to the point of sometimes not being able to receive anything, just by providing it 4.95 volts instead of 5 volts.

This could have been resolved by building a static boost converter like with a MAX732, which has specs of providing 5 volts with an input of +1.8 volts, which is perfect for the current batteries, as their cutoff is at 2.0 volts.

Another method could have been to go with a single chip transceiver like a NRf24l01 which needs a power supply range of 1.9 to 3.6 volts. Ultimately a better component for transmitting would have been the Xbee pro 3 module.

Finished product



Code

To program the Arduino Pro Mini, the Arduino IDE 1.8.10 was used, a standard software. The coding part began with a state diagram, which can be seen in the Design section of this document. In order to achieve the necessary functionality, a few libraries were required, such as SoftwareSerial and TinyGPS for the GPS and

RadioHead for the radio signals. Using useful code from the examples that the libraries provide, the setup was relatively simple. Unfortunately, some of the draft codes had unexpected problems, which even after troubleshooting, the origins of, were unknown. The first thing that comes to mind, when designing a device like this, is functionality. The devices ought to execute different code, depending on time and inputs, so a switch was chosen for a framework of the main loop.
Which was used to determine the 3 modes that the device would operate in:

- Sending - case 1
- Listening/Checking sensors - case 2 (default)
- Silence/Sleep - case 3

One of the initial problems that occured was the halt of the program upon receiving signal and switching to case 1. Surprisingly, this problem was resolved by creating a manual switch with if statements. This is a simple solution to a problem which doesn't have an obvious cause.

```
void loop() {
if (x==1){ //SENDING
  // mode 1
  // x = 3
}

if (x==2){ //LISTENING
  // mode 2
  // x = 1
}

if (x==3){ //SLEEPING
  // mode 3
  // x = 2
}
}
```

The next problem that resulted in removing one of the key components for a well structured program, was a failure of the timer/interrupt. The chosen time period for sending signals is long, compared to the speed of the microprocessor's timers. In order to make efficient use of the timers, the Atmega 328's Timer 1 was used to control the sending mode. It is the only 16bit timer the MCU has and it allows the counter to reach a value of 65536. Using the largest prescaler of 1024, one tick equals 256 us, it was easy to reach a period of 1 minute, by making use of the overflow interrupts. But, for uncovered reasons, the timer increased speed after one 1 minute of running. Also, the use of Timer 1 interfered with receiving the signal, so the assumption is that the RadioHead library is also using Timer 1. In order to combat this setback, for loops were introduced to the sending and listening mode.

```
void loop(){
  if(x==1){
    for(i=0; i<170; i++){

    x = 3; // after approximately 1 minute, change to mode 3(SILENCE)
  }
  if(x==2){ // WAITING FOR SIGNAL AND CHECKING SENSORS
    for(int i = 0; i < 50; i++)
    {

    } //end of for loop
    delay(10000); // Sleep, to conserve power
  }
```

IMPORTANT: In some parts of the code there are delays of 10 - 15 seconds, ways of conserving power will be looked into in further development.

```
if(x==1){
  for(i=0; i<170; i++){
    Serial.println("Sending");

    const char *msg = "example coordinates"; // supposed to be the coordinates of the received signal
                                             // OR self.coordinates if sensors triggered
    driver.send((uint8_t *)msg, strlen(msg)); // transmit the message
    driver.waitPacketSent();
    delay(200);
  }
  x = 3; // after approximately 1 minute, change to mode 3(SILENCE)
}
```

Mode 1 - Sending

```
if(x==2){ // WAITING FOR SIGNAL AND CHECKING SENSORS
  for(int i = 0; i < 50; i++) // check signal and sensors periodically
  {
 // read the input pin:
  int temperature;
  int flame = digitalRead(senl); // check infrared sensor
  int y = analogRead(temp);      // check temperature sensor
  temperature = map(y, 0, 1024, 0, 150); // map to get degrees

  delay(200);
  Serial.println();

  gps.f_get_position(&flat, &flon, &age);

  // Receive signal
  if (driver.recv(buf, &buflen)){ // Non-blocking
    Serial.println("Message received: ");
    Serial.println((char*)buf);

    x = 1; // upon receiving a signal start sending the coordinates received
  }

//   Check sensors
  if(y >= 500){ // 75 degrees
    if(y >= 850 | flame == 0){ // triger on either VERY HIGH temperature or  high temperature and flame
      delay(500);
      x = 1; // upon triggering send own coordinates
    }
  }
  } //end of for loop
  delay(10000); // Sleep, to conserve power
```

Mode 2 - Listening/Checking sensors

To prevent looping of the signal from other nearby devices, the third mode puts the MCU to sleep for 60 seconds. Again, an alternative to the big delay will be introduced in the future.

```
if(x==3){ // SILENCE to avoid looping
  digitalWrite(13, LOW);
  delay(60000);// 1 min
  x = 2;
}
```

The above images depict one of the final drafts on how the program looks, without structured printing on the serial port.
The second mode includes two conditional statements, which determine the switch to the sending mode. The first condition is upon receiving a message, start transmitting it, but more on that later. The other condition is upon triggering the sensors placed on the unit. Unfortunately, due to unforeseen circumstances, the team couldn't use the multiple infrared sensor they desired and there is only one flame sensor positioned in one direction. In order to combat this problem, the switch is initiated on detecting flame and having a high temperature input from the temperature sensor OR having very high temperature, even without detecting flame.

```
  // Receive signal
  if (driver.recv(buf, &buflen)){ // Non-blocking
    Serial.println("Message received: ");
    Serial.println((char*)buf);

    x = 1; // upon receiving a signal start sending the coordinates received
  }

//   Check sensors
  if(y >= 500){ // 75 degrees
    if(y >= 850 | flame == 0){ // triger on either VERY HIGH temperature or  high temperature and flame
      delay(500);
      x = 1; // upon triggering send own coordinates
    }
  }
```

A few things that will be addressed in the next few paragraphs are missing and solutions are being developed.

There was a problem of significant importance with the receiver component. After troubleshooting, it was discovered that not enough power reached the receiver and that prevented it from receiving properly. While testing and outputting to the PCs, the receiver was powered by the UART 5V, but the output of the boost converter (4.95V) wasn't enough.

One of the main functions of the unit is to sense the environment around it and upon meeting the conditions, send the units own coordinates.The RadioHead library can

only send strings, but the coordinates values are of data type float. There isn't a function in the Arduino arsenal that can convert a float, straight to a string. One of the solutions to this problem is using the <string.h> library, but working with temporaries and String objects might take too much of the already not too big heap of the arduino. The essence of this idea is transforming the GPS coordinates from floats to integers, by multiplying them with 10000 and storing them in separate integer containers. Having four significant digits after the decimal point, gives a range of approximately 11 meters.

decimal places	degrees	distance
0	1	111 km
1	0.1	11.1 km
2	0.01	1.11 km
3	0.001	111 m
4	0.0001	11.1 m
5	0.00001	1.11 m
6	0.000001	11.1 cm
7	0.0000001	1.11 cm
8	0.00000001	1.11 mm

(https://gis.stackexchange.com/questions/8650/measuring-accuracy-of-latitude-and-longitude/8674#8674)

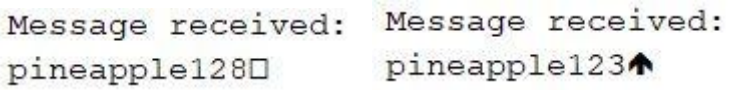
The <stdlib.h> contains a function that converts the integers into strings. Instead of creating objects, the <string.h> library is simply used to concatenate the final strings into a buffer and send that buffer over the radio signals. Here is an example code:

```
char *la;
char *lo;
void loop() {

char buf[15] = ""; //coordinates message
char int_buff[7];

float latitude;
float longitude; // float coordinates
long ilat;
long ilon;   // integer coordinates

latitude = 56.123456;
longitude = 10.123456;

ilat = (latitude*10000); // multiply by 10000 and store in the integer
ilon = (longitude*10000); // multuply by 10000 and store in the integer

la = ltoa(ilat, int_buff, 10); // convert latitude from long to string
strcat(buf, la); // concatenate with message

strcat(buf, "!");// separate the lat from lon in the message

lo = ltoa(ilon, int_buff, 10);// convert longitude from long to string
strcat(buf, lo); // concatenate with message

Serial.println(buf);


}
```

```
18:50:28.258 -> 561234!101234
18:50:28.258 -> 561234!101234
18:50:28.292 -> 561234!101234
18:50:28.292 -> 561234!101234
18:50:28.325 -> 561234!101234
18:50:28.325 -> 561234!101234
18:50:28.358 -> 561234!101234
18:50:28.358 -> 561234!101234
18:50:28.393 -> 561234!101234
18:50:28.393 -> 561234!101234
18:50:28.428 -> 561234!101234
18:50:28.428 -> 561234!101234
18:50:28.428 -> 561234!101234
18:50:28.463 -> 561234!101234
18:50:28.463 -> 561234!101234
18:50:28.496 -> 561234!101234
18:50:28.496 -> 561234!101234
18:50:28.529 -> 561234!101234
18:50:28.529 -> 561234!101234
18:50:28.563 -> 561234!101234
18:50:28.563 -> 561234!101234
18:50:28.598 -> 561234!101234
18:50:28.598 -> 561234!101234
18:50:28.632 -> 561234!101234
18:50:28.632 -> 561234!101234
18:50:28.632 -> 561234!101234
18:50:28.665 -> 561234!101234
18:50:28.665 -> 561234!101234
18:50:28.699 -> 561234!101234
18:50:28.699 -> 561234
```

Another problem when working with radio signals is identification, because transmitters send signals in no particular direction, as well as the receiver's

reception. In order to deal with a problem such as this, identification for each unit is necessary, additionally a passcode to confirm that the incoming signal is sent by one of the units in the FEWA network. The problem with comparing the received message with the prearranged password is that, often appended to the received message are different characters that in our estimate are unpredictable. Eg.



An important detail to mention is that the device only has to send its own coordinates when the sensors detect fire. In order to keep the code short and simple, instead of creating another mode to go into when sending signals. A solution is to check the sensors and send the units own coordinates while in the for loop for sending.
Here is how that will look:

```
if(x==1){
  for(i=0; i<170; i++){ // check for signals and sensors periodically
    Serial.println("Sending");
    const char *msg = "example coordinates";
    if(y >= 500){ // 75 degrees
    if(y >= 850 | flame == 0){ // triger on either VERY HIGH temperature or  high temperature and flame
      msg = ConvertOwnGPSCoordinatesToString();
    }
  }
  else{
    msg = ((char*)buf);
  }
    driver.send((uint8_t *)msg, strlen(msg));// transmit the message
    driver.waitPacketSent();
    delay(200);
  }
  x = 3; // after approximately 1 minute, change to mode 3(SILENCE)
```

End-point receiver code
On the end-point the code is very simple. We again use a RadioHead library as it only requires to receive signal and whenever it does, the arduino sets pins 3-7 HIGH, lighting up the LEDs and indicating the fire. It also prints out the GPS coordinates on the serial monitor.

```
if (driver.recv(buf, &buflen)) // if signal received
{
  int i;
  Serial.print("GPS: ");
  Serial.println((char*)buf);
  digitalWrite(3,HIGH);
  digitalWrite(4,HIGH);
  digitalWrite(5,HIGH);
  digitalWrite(6,HIGH);
  digitalWrite(7,HIGH);
```

If there's no signal all LEDs are set to LOW.

Perspective/reflection

We could have taken more time to brainstorm and research the parts we wanted to implement in our project. That lead us to some serious dead-ends and last minute changes, which resulted in loss of time while looking and ordering new components. In the end, we were able to implement all the features we had in plans.

As described in various places there is a lot of room for improvement in many different aspects of the product.

Power save mode

The current processor has a built-in power save mode, but it requires an outside power source to reactivate, and with the current setup this would be impossible. Doing so would greatly improve battery life of the product, as there is no need to constantly be searching for fires.

Introducing solar panels in combination with the abovementioned power save mode would greatly increase the battery-life of the product, and reduce the need for constant maintenance of FEWA

Maintenance

To help with maintenance of FEWA and make it a long-lasting solution, having the units transmit information of its battery level and sensor condition would be a great help for servicing the units.

A single FEWA unit might, because of some unforeseen circumstances, be the only link between two parts of the forest, and it would be unfortunate to have the link broken because of lack of maintenance.

Broadcasting with an included unique identification code would be another great assist in maintenance, as it could assist in locating the problematic FEWA unit.

Passcode

Further development of FEWA would solve a problem that we've encountered testing our signal is not encrypted or keyed in any fashion, so if anyone with a person with malicious intent were to spoof a signal on the same wavelength, then all nearby FEWA units would trigger and start relaying that information, which wouldn't necessarily trigger an alarm as the data being sent didn't include GPS data, but if someone with malicious intent wanted to, it would be possible for them to spoof the fire signal.

To resolve this a password could be introduced into every device, so that the FEWA units won't relay information unless the specified password is included.

This isn't a foolproof plan, as an individual with malicious intent could sniff out the password but that would require them to start a fire below one of the units.

End-point

The main purpose of end-point is to give firewatch lookouts a signal about the occuring fire. However, as an additional function it might also notify the fire department or any one who could be affected by the fire (for example people living nearby the forest).
This could be done by sending out an SMS notification to any person interested who signed up for receiving this kind of warning. It may make any evacuation scenario much easier.

Casing

The current state of the FEWA prototype does not have a protective case and everything is dangling from wires which makes it rather fragile. An immediate improvement upon the product would be designing and building a waterproof plastic case.

The case would have to be made of a plastic that is a bad conductor of heat so that the transmitter will have time to send before dying in a fiery death. A material like

nylon 66 would be appropriate as it has a high mechanical strength, rigidity and stability under extreme heat.

Conclusion

Our project was developed as a proof of concept and because of the implemented technology FEWA is not suitable to apply in real life solutions. Due to certain limitations, either because of the capacities of the hardware, materials or the software used we were not able to finish the entire setup.

The receiver and transmitter is one of the major setbacks in the FEWA units. The components were not designed for our scenario, as the transmission range is nowhere near the capabilities we would have wanted and the input voltage turned out to not be fully compatible with our initial idea. The other major problem was researching the components, as most of them don’t have proper datasheet. Some of the values were approximate or noticed by some other users and posted on internet forums.

Appendices

NOT DONE

Code is attached

- IR flame sensor (octopus one) - detection range around 120°, digital or analog output, “digital outputs adjustable detection range, the analog output sensitivity adjustable”, 3.3V-9V operating voltage
- LM35 temperature sensor - linear +10-mV/°C, 0.5°C accuracy, −55°C to 150°C range, 4-30 V operating voltage
- NEO-6M gps module - 2,7-3,6V operating voltage, UART/USB/SPI/DDC interface (https://www.rlocman.ru/i/File/2011/04/22/1.pdf) https://arduinotech.dk/shop/arduino-gps-modul/ Module 3, low power consumption
- Batteries (included protection circuit) https://let-elektronik.dk/shop/1570-batteri/400075-lithium-ion-polymer-rechargeable-battery---37v-2000mah-cables-70mm/
- Arduino pro mini https://let-elektronik.dk/shop/300-boards/11114-arduino-pro-mini-328---33v8mhz/
- 433 mhz RF module https://elektronik-lavpris.dk/p128724/modu0007-433mhz-wireless-modules-mx-fs-03v mx-05/
- 0.9V-5V boost step-up for RF Receiver https://elektronik-lavpris.dk/p129671/modu0018-dc-dc-usb-09v-5v-to-5vdc-boost-step-up-mini-pfm-control/