Boston University
Electrical & Computer Engineering
EC464 Capstone Senior Design Project

User's Manual

Hazard Harbinger

by

Team 12
Team Hazard Harbinger

Team Members

Julia Hua jhua2@bu.edu
Noah Cherry ncherry@bu.edu
Timothy Borunov timobohr@bu.edu
Abdulaziz Almailam almailam@bu.edu

Submitted: April 19, 2024

User Manual

Table of Contents

Executive Summary

Wildfires are major environmental disasters with far-reaching consequences on the environment and human mortality. A common practice to prevent wildfires is prescribed burning, which requires thorough planning. To aid these burn plans, we propose a novel system that integrates an airborne LiDAR system and deployable sensor nodes. This system will result in 3D models and relevant environmental data.

1 Introduction

To address the increased frequency and intensity of wildfires across the world, fire managers have been practicing controlled burning to remove excess dead, flammable, and overgrown vegetation. In order to execute controlled burning safely and successfully, they require a comprehensive burn plan that accounts for the land's topography, vegetation, and environmental conditions, such as humidity, air quality, etc. Our project aims to provide fire managers with 3D models and environmental data so that they can plan for controlled burning operations more effectively.

Fire managers create the burn plan by analyzing data that can be useful for predicting the fire's speed, intensity, and direction. One method they use to analyze the gathered data is the 3D fuel map, which depicts the spatial arrangement and characteristics of the land and vegetation. For example, the topography of the land can predict the direction and rate of fire spread, and a denser forest and higher amounts of dead vegetation can indicate a more intense and uncontrollable fire. Fire managers must also consider the environmental conditions. This data can be collected by deploying several embedded sensor nodes. For example, temperature heavily influences how quickly a fire can develop; at higher temperatures, burning can be hazardous, but at lower temperatures, the vegetation may not catch fire as easily. Similarly, lower humidity may indicate drier fuel and, hence, a more intense fire. Together, the fuel map and environmental data can facilitate fire managers in safely planning prescribed burns.

Our final product is a land inspection system to collect and process both topographical and environmental data, resulting in a 3D fuel map featuring all gathered information. Our system has two components. The first uses an airborne LiDAR system (ALS), which scans a forest region to provide a 3D point cloud of the land and vegetation. Then, the 3D point cloud is processed to produce the final 3D fuel map. The second component is the embedded sensor nodes. These nodes are cheap and energy-efficient pieces of hardware that can be simply placed across the forest floor, and passively collect environmental data. Our system enables fire managers to efficiently prescribe controlled burns without onsite human investigation.

2 System Overview and Installation

2.1 Overview block diagram

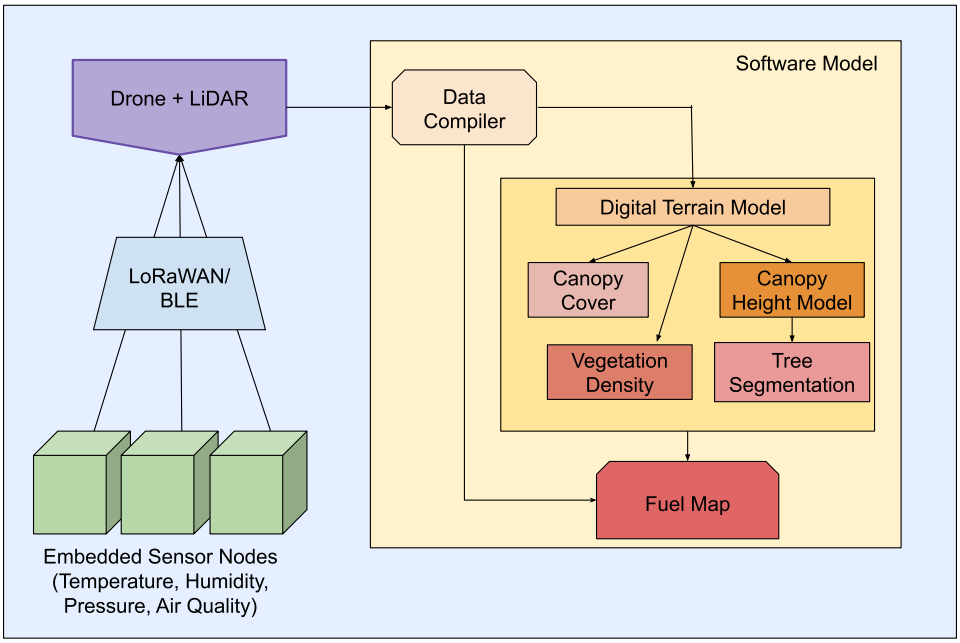

Figure 2.1: Block Diagram

Figure 2.1 illustrates the drone with a LiDAR mounted (i.e., ALS). The left side shows the ALS communicating with the embedded sensor nodes to capture environmental data, such as temperature and air quality. The right side demonstrates the drone scanning a forest region to acquire data for constructing the fuel map.

2.2 Physical description.

Figures 2.2 and 2.3 show the LiDAR and Jetson Nano mounted to the drone with a 3D-printed mount. The battery for the drone not only powers the drone but the LiDAR and Jetson Nano as well.

Figure 2.2: Aerial LiDAR System (Top View)

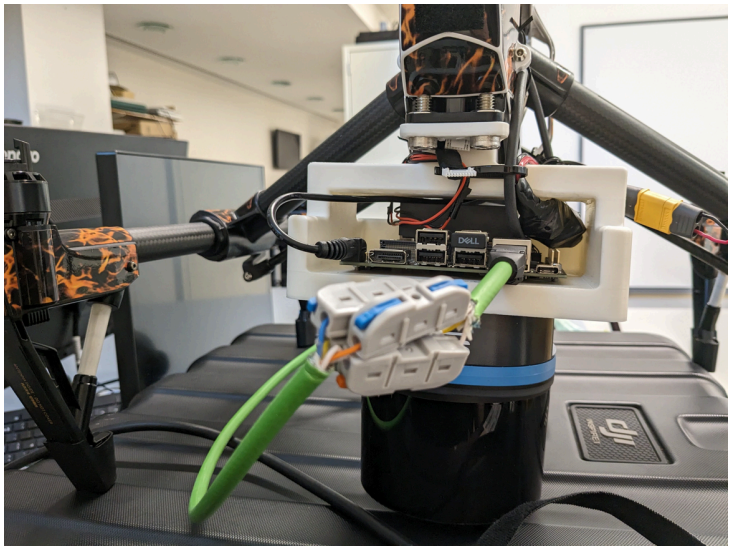

Figure 2.3: Aerial LiDAR System (Side View)

2.3 Installation, setup, and support

The device has an on board computer, the Jetson Nano which comes with all the dependencies and software pre-installed. As this is a prototype, it is highly discouraged to use hardware with our device which is different from the onboard computer as we cannot guarantee that the presets and dependencies of that device will allow for proper operation. However, if it is still desired to work on a separate device, please follow the following software installation procedure:

2.3.1 Software Installation

As this device is a prototype, multiple version support is not available for this device, so the installation and dependencies are very specific and must be followed exactly to guarantee proper functionality using the SICK and ROS libraries. The following steps describe the exact installation procedure which must be done in order to run the appropriate libraries:

1) First make sure that you have a computer running Ubuntu 20.04 and is using python version 2.7.10.
2) Install ROS Noetic by executing the following instructions:
    a) sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'
    b) sudo apt install curl # if you haven't already installed curl
    c) curl -s https://raw.githubusercontent.com/ros/rosdistro/master/ros.asc | sudo apt-key add -
    d) sudo apt update
    e) sudo apt install ros-noetic-desktop-full
    f) source /opt/ros/noetic/setup.bash
    g) sudo apt install python3-rosdep python3-rosinstall python3-rosinstall-generator python3-wstool build-essential
    h) sudo apt install python3-rosdep

i) sudo rosdep init
j) rosdep update

3) Next directly install SICK libraries using the following:
   a) sudo apt update
   b) sudo apt-get install ros-noetic-sick-scan-xd
4) Next install the libraries necessary for operating our specific device:
   a) git clone https://github.com/Hazard-Harbingers-EC463/LiDAR_Detector_Drone_HH.git
   b) From the cloned repository, add the laser_to_pcl folder with all of its contents to the ROS repository with the rest of the launch files, which if you followed all of the previous steps without changing directories, would be /opt/ros/noetic/share/
   c) Install the latest version of FUSION
   d) Install PDAL package
      i) conda create --yes –name myenv --channel conda-forge pdal
      ii) conda update pdal
      iii) conda activate myenv
      iv) conda install matplotlib

2.3.2 Hardware Installation

The product comes attached with the LiDAR, Jetson Nano, and drone already installed and working together. To view the data and utilize the scans, a way to view the contents of the computer must be accessed in order to start and stop the scans and perform model analysis on the locally attached Jetson Nano. To do this, there is an open DP port for an adapter/monitor/tablet to be attached, along with several USB ports optional for a mouse and keyboard. The Jetson Nano has all of the libraries to use the device preinstalled, running Ubuntu 20.04 and so every command can be executed with an understanding of linux terminal commands.

1. Sensor Node Placement
   a. Insert SD card into SD card slot
   b. Place sensor nodes in elevated position (1.5m) in environment to be observed
   c. Ensure all sensor nodes are separated by a minimum distance of 0.5km
   d. When sensor nodes is placed properly
      i. Remove snap-fit lid
      ii. Switch node into “on” state
      iii. Check all SoL LEDs are illuminated
      iv. Replace cover
2. Collector Node Setup
   a. Insert SD card
   b. Switch node into “on” state
3. Collection
   a. Fly within 0.5km of sensor nodes
   b. Return to operator
   c. remove SD card

d. Read/post process SD card data

3 Operation of the Project

3.1 Operating Mode 1: Normal Operation

Operation of our product starts with positioning and enabling the embedded sensor nodes in the desired observation area. Once fitted with an SD card and switched on, the sensor nodes are prepared to be placed in their observation location 0.5km apart.

In regular operation, a user would attach a mobile viewing platform such as a tablet or monitor to the display port on the Jetson Nano to gain access to the Jetson. From here, the user is able to start a scan using the LiDAR using the following commands:

1. Run this command in a terminal: roslaunch sick_scan_xd sick_multiscan.launch hostname:=192.168.0.2 udp_receiver_ip:=192.168.0.3
2. Run this command in a separate terminal: rosbag record /cloud_all_fields_fullframe

When the scan is started, the user can unplug the display, and fly the drone as if a regular drone around the area they wish to scan. The LiDAR scans in 360 degree range so the direction in which the drone is facing is irrelevant for the scan. In order to ensure accuracy of the scans and usability by models, the scan should feature below and above canopy flight in the same flight duration.

Prior to flight, the collector node should also be fitted with an SD and powered on to enable the collection broadcast message and receiver.

Once the flight is finished, the user can plug in the display again, stop the scan, and access the created rosbag file. The next step is to concatenate it and produce a las file. To do this, execute the following commands in separate terminals:

1. Run the command "roscore"
2. Run the command "rqt_bag" in the directory with the rosbag. This brings up a user UI where you can choose the rosbag file and right click on the created stream to publish the data to topic /cloud_all_fields_fullframe
3. Run the command "roslaunch laser_to_pcl start.launch
4. Run the command "rosrun laser_to_pcl laser2pcl.py" It will output a command line counter of all the pcd points gather. Run the command until the counter does not change.
5. Run the command "rosbag record /laser_pointcloud" until it gives buffer size reached warning
6. Run the command “rosrun pcl_ros bag_to_pcd <input.bag> /laser_pointcloud <pcd_output_directory>”, where input.bag is the resulting rosbag file from step 5.
7. In pcd_output_directory, run the command “pdal translate <input.pcd> <output.las>”. Let the current directory be DATA_DIR.

The final step is to produce the models. Complete the following steps:

1. Change the directory to where the Hazard-Harbingers-EC463 github repository was cloned to
2. Run: cd Models
3. Run: ./processData.sh
4. When a prompt for a file path appears, input the file path to the output LAS file in DATA_DIR.
5. The resulting models and 2D heat maps are stored in DATA_DIR/outputs folder. The user can also extract the SD card from the collector node and process the stored CSV data as needed.

3.2 Operating Mode 2: Abnormal Operations

Since the SICK multiscan itself is a prototype, there may be issues with using the SICK library commands. Since we are simply using those libraries to operate the LiDAR, abnormal operations could arise with the libraries used from SICK. In that case we highly recommend looking at the SICK github (https://github.com/SICKAG/sick_scan_xd) for support on library issues. Same applies to using ROS functions.

As for our device, abnormal operations could arise with either the software written to concatenate scans or with node information transmission.

For the scans, if the user manual has an issue with the counter for the cloud being zero without ever updating, then it is recommended to check the following common issues:

1. Check to make sure that clock is set to the correct time. The bag read operation is tied to the clock of the Jetson and if the time is set differently during the scan vs during the playback during rosbag record, the concatenator will not find any points in the correct time range to record.

2. Check to make sure that *roscore* is running and try rebooting it.

3. If the issues persist, reboot the entire Jetson and retry the network connection as sometimes it does not properly connect with the specified network connection and becomes incapable of performing the concatenation.

For the nodes:

The embedded nodes contain several hardware and software components that may face abnormal operating conditions. For ease of troubleshooting this manual differentiates hardware and software related issues.

3.2.1 Embedded Node Software Troubleshooting

1. The user may experience device panic (device freezing) during setup or operation. This results from the sensor nodes being placed too densely in the observed region. If the collector node receives two responses from a collection request packet, the LoRa interrupt will trigger twice before completion, triggering the collector to panic and cease functioning until it is reset. Ensure sensor nodes are distant enough to avoid interference (distances > 0.5km).

2. For issues with SD card storage and LoRa transmissions, set debug flag definition in script header to 1. All SD reads and LoRa packets are dumped to the serial console.

3.2.1 Embedded Node Hardware Troubleshooting

1. The sensor nodes require an exposed enclosure to accurately read air quality and humidity conditions. Long exposure may cause shorts to hardware connections. To check proper connectivity to each subsystem (BME688, SD card, LoRa), initialization messages are output via Serial port. Shorts or disconnected systems will output an init failed message to assist with hardware debugging.
2. BME688, SAMD21 Pro RF, and SD reader each have sign of life (SoL) LEDs that signal proper functionality.
3. The SAMD21 illuminates its onboard LED when sending and receiving LoRa packets. If no lights are illuminated when transmitting or receiving is expected, inspect verbose debug messages (set debug flag to 1).

3.3 Safety Issues

Particularly pertaining to the airborne aspect of this product, significant regulations are in place to protect users and civilians from airborne hazards. The FAA specifically requires drones greater than approximately 250 grams to be registered and piloted by a trained operator. As the fully-equipped drone (incl. LiDAR & Jetson Nano) weighs approximately 3.34 kilograms, it falls under very strict FAA guidelines. *As the FAA and FCC guidelines frequently change, please refer to the FAA and the FCC's website for the latest regulatory information.*

The embedded sensor nodes pose no significant risk to the environment or the operator, and are classified as both non-toxic and non-hazardous. The embedded sensor nodes were designed to be weatherproof and are housed in a structurally-stable housing that should be mounted at a height above 3 meters from the ground level.

Furthermore, the embedded sensor does operate on the unlicensed and open 915 MHz frequency. Note that regulations vary by region and that the 915 MHz frequency is license-free specifically in North America. Use of this product in the European market would require slight modification to the operating frequency as only the 433 MHz frequency is open; the 915 MHz frequency is strictly regulated. *In any case, always refer to the appropriate communications regulatory authority in your area before any installation or operation of these devices. If you are in the United States, please refer to the FCC.*

4 Technical Background

4.1 Aerial LiDAR System

The first component in our system is the aerial LiDAR system (ALS), which includes a 360° LiDAR. The drone, mounted with the LiDAR and a Jetson Nano, scans above and below the canopy to provide data on the vegetation and topography of a forest region. This captured data, which is stored as a bag file created by a tool called rosbag[1], is then converted into a LAS file using Laser Assembler[2] and PDAL.[3] This LAS file can be viewed as a 3D point cloud using LDV.[4]

After the preprocessing step, we derive several models from the 3D point cloud: digital terrain model (DTM), canopy height model (CHM), tree segmentation, canopy cover model, and vegetation density models. These models are created using a shell script that uses tools for processing LiDAR data called FUSION[5] and LAStools.[6] First, the DTM provides valuable information about the land's topography, such as the terrain and the direction of the slope. This model is created by first filtering out the ground points (GroundFilter from FUSION) and then interpolating those points to create a smooth surface (GridSurfaceCreate from FUSION). Second, the CHM provides information about the height and structure of the vegetation, which is produced by taking the highest elevation in each grid cell and subtracting the ground elevations.

Next, to model the density and distribution of trees, we identify tree segments using TreeSeg from FUSION, which applies a watershed segmentation algorithm to the CHM[7] and results in a CSV file. Subsequently, the percentage of canopy cover and vegetation density, which ranges between 0% and 100%, at different height stratas in each region is calculated to create a canopy cover model and vegetation density models, both of which use Cover from FUSION. These models result in DTM files. Finally, to better visualize the tree clusters, canopy cover percentages, and vegetation density, a python script reads from the CSV and DTM files to plot them as a heat map.

4.2 Embedded Nodes

The embedded nodes are an expandable and modifiable system used to collect long term (> 1 Month) data in regions greater than 0.5km^2. The nodes are designed around a SAMD21 Pro RF microcontroller with integrated long range RFM95W LoRa transceiver. The SAMD21 chip natively supports a low power deep sleep state as well as watchdog and GPIO hardware interrupts. Placing the SAMD21 Pro RF into deep sleep reduces nominal power consumption to 3μA which allows for greatly extended lifetime when compared to its standard operating mode. Power consumption can be modeled by

[1] https://wiki.ros.org/rosbag
[2] https://wiki.ros.org/laser_assembler
[3] https://pdal.io/en/2.7-maintenance/
[4] http://forsys.cfr.washington.edu/fusion/fusion_overview.html
[5] http://forsys.cfr.washington.edu/fusion/fusion_overview.html
[6] https://lastools.github.io/
[7] http://forsys.cfr.washington.edu/software/fusion/FUSION_manual.pdf

$Power\ Consumption\ =\ 3.3V\frac{\left(T_{active}\times 23mA+T_{sleep}\times 3\mu A\right)}{T_{active}+T_{sleep}}$. T_{active} and T_{sleep} are adjusted by modifying the duration of data collection after wake or the wakeup policy. Extending the duration of data collection provides more samples for each collection cycle which may reduce the impact of outliers at the cost of power consumption. A high frequency wakeup policy increases granularity of data, but incurs more wakeup overhead. It is recommended for $\%T_{active}$ to not exceed 99% of total deployed time. This restriction ensures maximum battery lifetime.

The RFM95W 915MHz LoRa transceiver has an estimated expected range of 0.5km in forest environments with a dipole wire antenna operating at 10dBm. This figure is adjustable by increasing transmission power or substituting a higher gain antenna in exchange for more power consumption. Higher transmission range increases the minimum distance from the sensor node the drone must reach, but decreases the density of sensor nodes.

The BME688 sensor and SD card storage peripherals are connected to the SAMD21 Pro RF via the SPI bus. The SAMD21 watchdog timer wakes the board at an interval specified by the wakeup policy, collects ambient temperature, pressure, humidity, node altitude, volatile organic compounds, and sulfur organic compounds. Raw data from each read is stored in CSV format on the sensor node SD card before returning the node to deep sleep. Our wireless transmission and collection protocol is agnostic to the size of each entry in the CSV file, so additional sensors and measurement devices can be added to the node by connecting to the SPI bus and adding the collected values to the CSV write format.

The drone collector node has the same SAMD21-SD storage framework as the sensor nodes without the sensor peripherals. When active, every 10 seconds it broadcasts a LoRa packet containing a request message. When the sensor node receives the request message it responds by reading the SD card packet by packet and transmitting it to the collector node. Upon sending a packet with the EOF character, the sensor node sets a flag preventing it from sending data for 2 hours to prevent multiple data collection.
Sensor and collector nodes are powered by two AA batteries connected to the battery terminals on the SAMD21.

The enclosure is printed from PLA and has a perforated lid and base to allow sufficient airflow for cooling and air quality and humidity measurement. Future iterations will increase the mounting ergonomics of the enclosure and environmental shielding for the board.

5 Relevant Engineering Standards

Our project conforms to various standard regulatory guidelines. Our devices are compliant with the European Restriction of Hazardous Substances in Electrical and Electronic Equipment (RoHS) as well as suitable for UL compliance. As per the RoHS, all solder is lead-free and contains minimal flux (<3%) content to limit corrosion.

The project further abides by relevant National Electrical Code (NEC) rules and regulations, particularly pertaining to safety functions. Our hardware features single-way connectors as well as reverse polarity, overcurrent, overtemperature, overvoltage, and undervoltage protection.

Furthermore, the multiScan100 is connected to the Jetson Orin Nano via a PROFINET[8] connection, which is described as the industry technical standard for data communication over Industrial Ethernet.This four-wire connection provides uninterrupted real-time data transmission with added failsafe features.

[8] https://us.profinet.com/technology/profinet/

6 Cost Breakdown

The major project costs are expressed in the table below. Line items 1 through 3 are dedicated to the airborne LiDAR aspect of the project and represent the majority of the project's cost. The remaining line items represent the embedded node aspect of the project.

Project Costs for Production of Beta Version (Next Unit after Prototype)				
Item	Quantity	Description	Unit Cost	Extended Cost
1	1	SICK multiScan100	$4,827.00	$4,827.00
2	1	DJI Inspire 1 Pro V2.0	$690.00	$690.00
3	1	NVIDIA Jetson Orin Nano	$499.00	$499.00
4	3	Adafruit MicroSD SPI Breakout Board	$3.50	$10.50
5	2	Bosch Sensortech BME688 Module	$19.95	$39.00
6	3	Sparkfun SAMD21G Pro RF Board	$33.95	$101.85
7	3	SanDisk Ultra 16GB MicroSD Card Class 10	$7.75	$23.25
			Beta Version-Total Cost	$6,190.60

The Bill of Materials (BOM) and general project cost estimate above omits minor, negligible costs such as breadboarding wires and jumpers. When purchased in bulk quantities (>100), significant price breaks are offered for line items 4 through 7 of approximately (20%) of the cost, resulting in notable savings.

These cost savings are particularly relevant when considering that a typical embedded node installation would require large quantities (i.e., 100 ~ 200) of embedded node hardware (ref. line items 4 through 7), extending significant discounts to the end user.

7 Appendices

7.1 Appendix A - Specifications

Mounted LiDAR drone system specifications:

1. Minimum Flight Time: >20 minutes
2. Data Storage: 128GB microSD to hold all data
3. SICK LiDAR multiScan100 mounted using custom 3D printed mount
4. Jetson Orin Nano 8GB with RAM for live data processing and collection
5. DJI Inspire 1 Pro V2.0 with TB48 Battery (~25.6V nominal voltage)

Embedded node system specifications:

1. Battery Life: >1 year (typ. usage)
2. BME688 Sensor Measuring:
    a. Temperature (-40 – 85 °C, ± 3°C)
    b. Humidity (0 – 100 %, ± 3%)
    c. Pressure (300 – 1100 hPa, ± 0.25%)
    d. Gas (Raw Resistance of Heater kΩ) – Additional Estimated Outputs:
        i. Index for Air Quality (IAQ)
        ii. Biogenic Volatile Organic Compounds (bVOC) (ppm)
        iii. CO2 equivalents (ppm)
3. Operating Temperature Range: -20C to 50C
4. Wireless Data Transmission via Hope RFM95W LoRa modem

Software system specifications:

1. LiDAR scan in ROSBAG format can be converted to a LAS file.
2. 3D models of DTM, CHM, tree clusters, canopy cover, and vegetation density can be produced.
3. 2D heat maps of the tree clusters, canopy cover, and vegetation density can be produced.

7.2 Appendix B – Team Information

Team Members:

- Abdulaziz Almailam[1], almailam@bu.edu
- Timothy Borunov[1], timobohr@bu.edu
- Noah Cherry[1], ncherry@bu.edu
- Julia Hua[1], jhua2@bu.edu

Faculty Advisors:

- Prof. Osama AlShaykh[1], osama@bu.edu
- Prof. Alan Pisano[1], apisano@bu.edu
- Prof. Michael Hirsch[1], mhirsch@bu.edu

[1]Department of Electrical and Computer Engineering, Boston University
Boston, MA 02215