# Flying Under the Radar

Angie Bu, Ayesha Gupta, Linsey Szabo

# Use Case

## Universal Drone Attachment

Maximize usability and cover large areas despite difficult terrain

## Fire SAR Missions

Reach areas where traditionally used infrared fails

## Break Cost Barrier

Our mmWave radar application is cheaper than drones currently used for SAR

## Human Detection

Detect and save locations of victims for efficient rescue

# Quantitative Use-Case Requirements

## Drone Compatible

- 30 minute flight duration
- Weighs <.5 kg
- Area of ~12 square inches

## Withstand Conditions

- Alert user around 100 degrees Celsius

## Safe Detection

- 10 m detection range
- 3 s end-to-end
- .7 F1 score for ML architecture

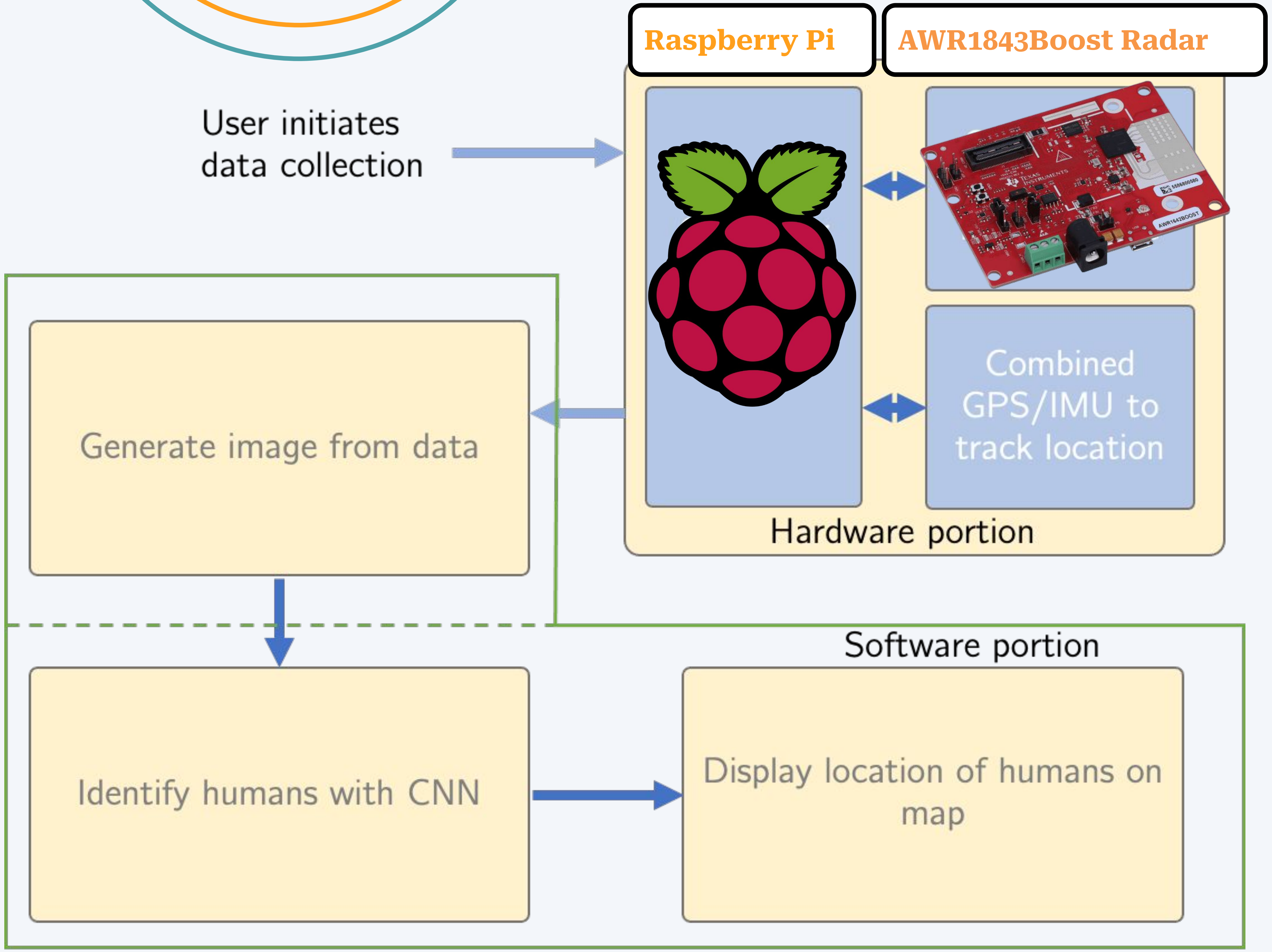
# Solution Approach

We propose a mmWave radar-based application robust to visible occlusions like fog or smoke that employs machine learning to accurately process the images and detect moving humans. Lastly, our web application frontend enables location saving to efficiently rescue victims.

**Since Last Time: Improving Usability**

- Alert user when device is reaching a temperature that will negatively affect functionality
- Loud beeping noise makes victims aware of device presence to make themselves more noticeable

# System Approach



# Hardware Approach

## Onboard sensors

Radar data streamed to Raspberry Pi at 10 Hz via SPI

GPS/IMU data streamed to Raspberry Pi at 10 Hz via I2C

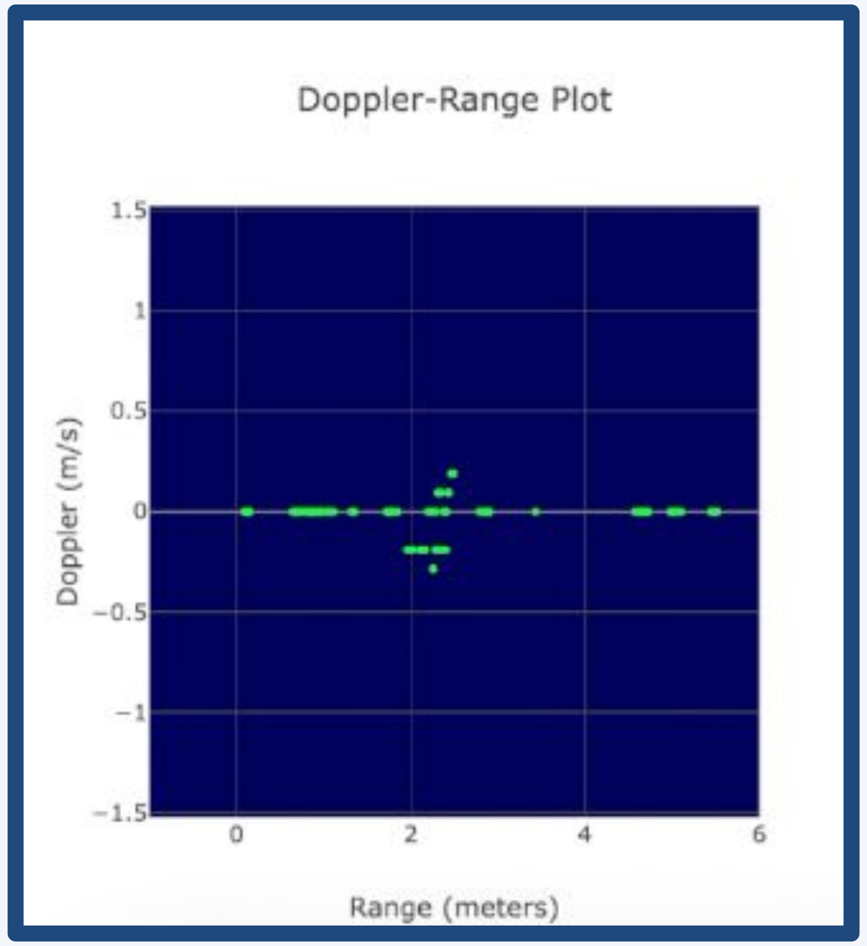
Temperature sensor data streamed at 1 Hz

## On the Raspberry Pi

Construct range-Doppler map from raw radar data

Determine if temperature is too high: 1 if too high, 0 otherwise

Speaker beeps and can play a pre-recorded message




Stream range-Doppler map, location, orientation, and temperature notification data to base station computer via WiFi

# Software Approach

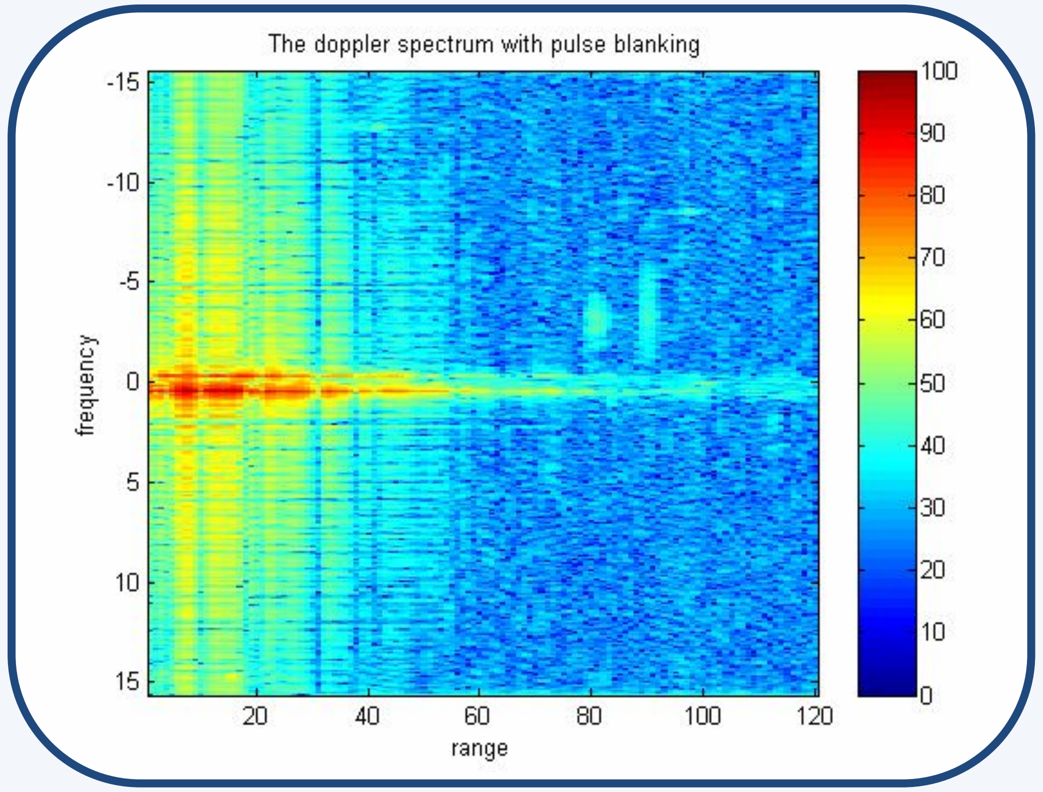
**Pre-Processing**

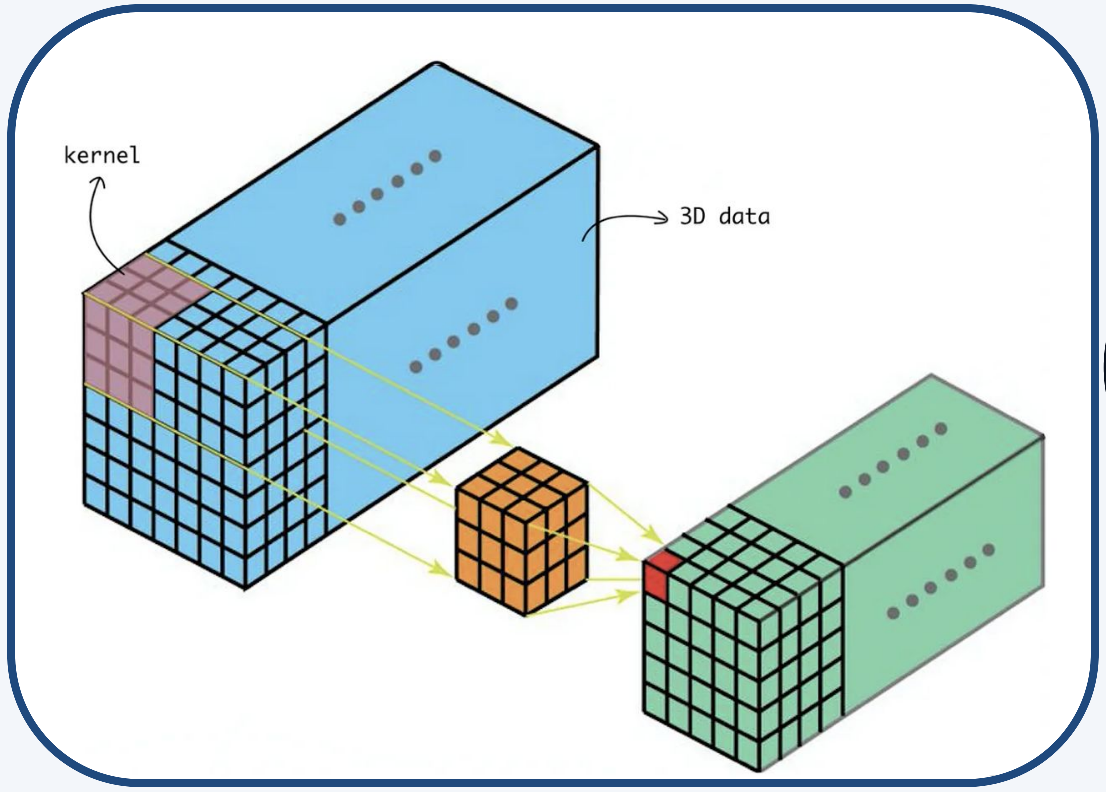
1. Constructing range-Doppler maps on Raspberry Pi

**Classification**

2. 3D CNN architecture embedded in web application

**The ML architecture will be encapsulated in a Django model to be run in the web application, deployed on Amazon EC2.**




**Human Localization**

3. Determine location of human based on range and radar pose, and then record location on map

**Network Output**

1 - human detected, 0 - otherwise

# Implementation Plan

| | |
|---|---|
| mmWave radar | AWR1843Boost courtesy of CyLab |
| Raspberry Pi | RPi 4 with 8GB RAM from course inventory |
| 3D printed chassis | PLA frame and radome printed at TechSpark |
| Temperature sensor | Adafruit TMP36 from TechSpark |
| GPS/IMU sensor | OzzMaker BerryGPS-IMUv4 purchased online |
| ML architecture | Keras API, drone radar dataset from Gent University |
| Google maps API | Will purchase online, easy integration into web app |
| Speaker | AS02008MR-LW152-R from previous coursework |
| 9V Battery | Assembly powered by single 9V battery |
| Buck-Boost Converter | From previous coursework |

# Testing, Verification, and Validation

## Quantitative Measures

- Timing (ms)
- Accuracy (%)
    - F1 Score
- Temperature (Celsius)

## Risks & Unknowns

- Hardware
    - Increase data rate for accuracy
    - Decrease data rate for speed
- ML
    - Tune hyperparameters
- Web app
    - Reduce HTTP requests
    - Simplify html/css files

# Testing, Verification, and Validation

## Test Inputs

- Temperature sensor: Heat gun
- Radar: Fog machine
- ML architecture: Drone Radar Dataset - Smart Robot Lab
- Web app: Manual geolocation inputs

## Test Outputs

- Radar angular resolution: 15º
- GPS/IMU localization accuracy: 0.5 m
- Temperature sensor: 100º C
- ML architecture: .7 F1 score
- Web app timing: 10 ms
- End-to-end timing: 3 s

# Drone Testing

We seek to test our sensor suite with the **DJI Matrice 100** drone after verification of our system:

- Integration with system on stationary, unpowered drone
- System functionality on hovering drone

# Project Management

| Week | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Acquiring radar | | | Angie | | | | | | | | | | | |
| Find a dataset | | | Linsey | | | | | | | | | | | |
| Set up web app | | | | Ayesha | Ayesha | | | | | | | | | |
| Train ML architecture on dataset | | | | | Linsey | | | | | | | | | |
| Validate ML architecture | | | | | | Linsey | | | | | | | | |
| Capture radar images | | | | Angie | Angie | Angie | Angie | Angie | Angie | | | | | |
| Radar integration with drone | | | | | | | | | | Angie | Angie | | | |
| Finish frontend functionality | | | | | | | | Ayesha | | | | | | |
| Test ML architectures on radar images | | | | | | | Angie | Linsey | | | | | | |
| Integrate ML architecture with web app | | | | | | | | | Ayesha | | | | | |
| Test whole pipeline (radar to web app) | | | | | | | | | | Angie | Linsey | Ayesha | | |
| Slack | | | | Ayesha | Angie | Ayesha | Linsey | Angie | Linsey | Ayesha | Angie | Linsey | Ayesha | Angie |

| | |
|---|---|
| Angie | |
| Linsey | |
| Ayesha | |