



DroneSense

BY: JEREMY GUY, JAYDEN GUY, ANGELINE DOYA,
ABBA ANARYU



Problem Statement

- ▶ Controlling a drone requires quick reflexes, especially if pilots are unfamiliar with their surroundings.
- ▶ Improper control causes accidents, injury, and property damage.
- ▶ Implementing a LiDAR sensor to detect nearby obstacles and send real-time data to the drone, allowing it to stop or reroute to prevent collisions.

Constraints



Must comply with **FAA** (flight regulations) and **FCC** (frequency & power) standards



Operation in various outdoor weather conditions



High altitude reduces lift, requiring efficient power management



Privacy concerns with camera use



Noise regulations in certain areas



Must identify obstacles and maintain stable flight control in real-time

Design Specifications



Drone Frame: F450 Frame



Flight Controller: Pixhawk 2.4.8



Operating System: Raspberry Pi 5 (Ubuntu 24.04)



LiDAR Sensor: Tfmini-S LiDAR



Power System: 3S LiPo 2200mAh 12.6V 50C



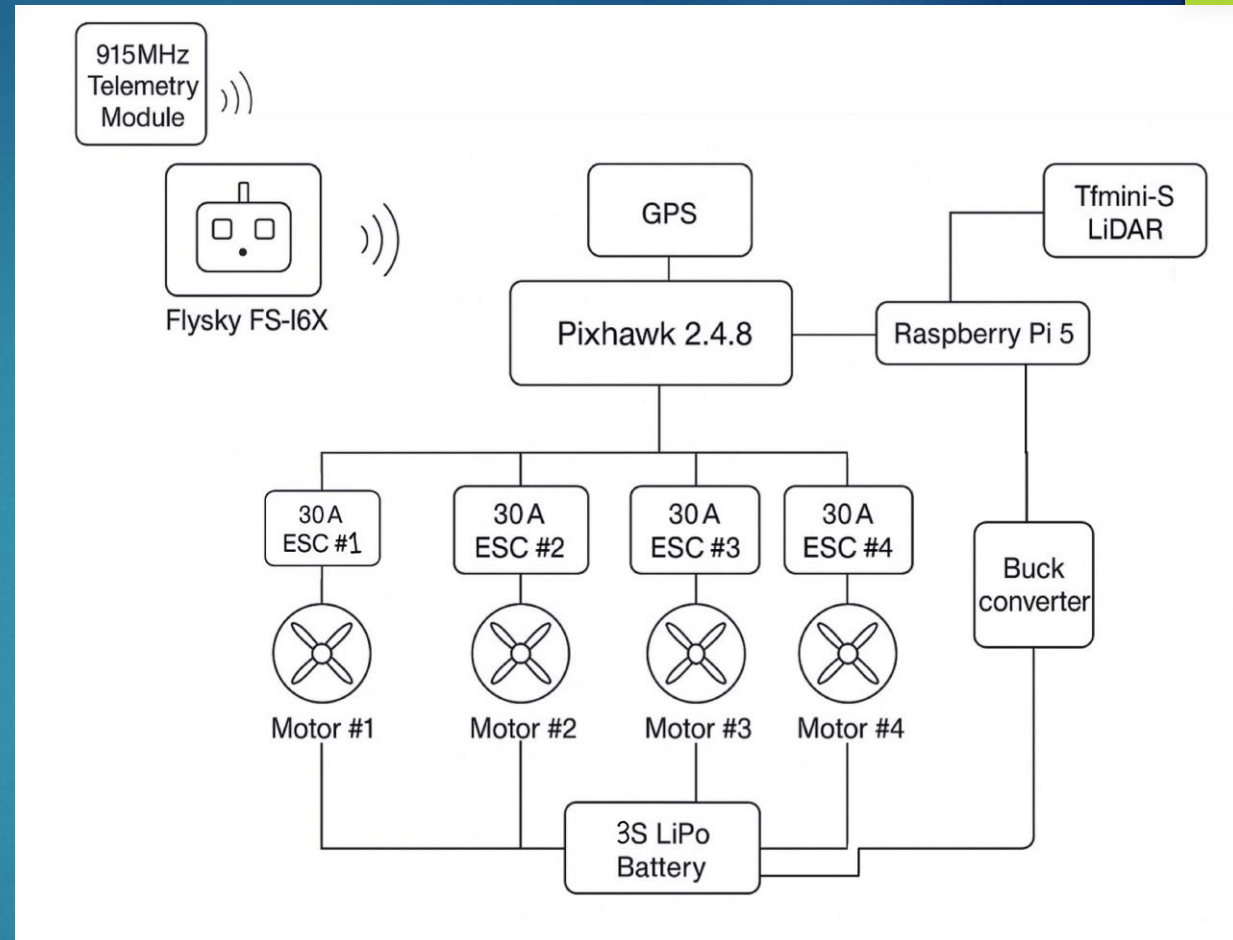
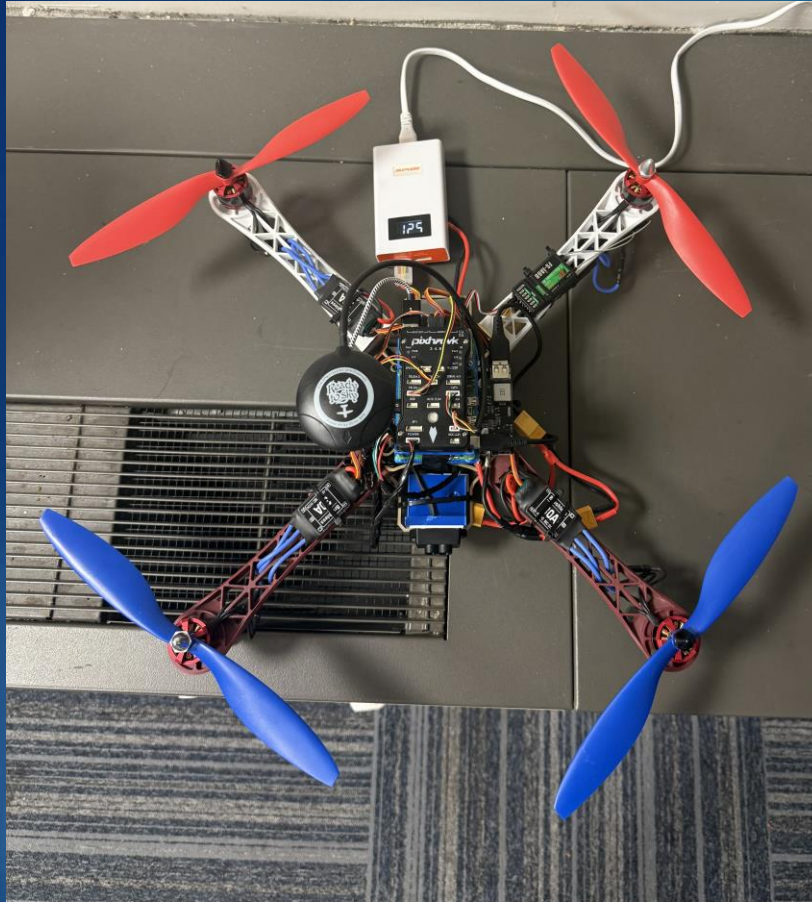
30A ESCs and brushless motors



Control System: Flysky FS-I6x transmitter and receiver



Telemetry: 915MHz 100mW RC Telemetry Kit



Component level design

Sprint #1



Basic Obstacle Detection algorithm in a simulated environment



Planned Weekly Tasks:



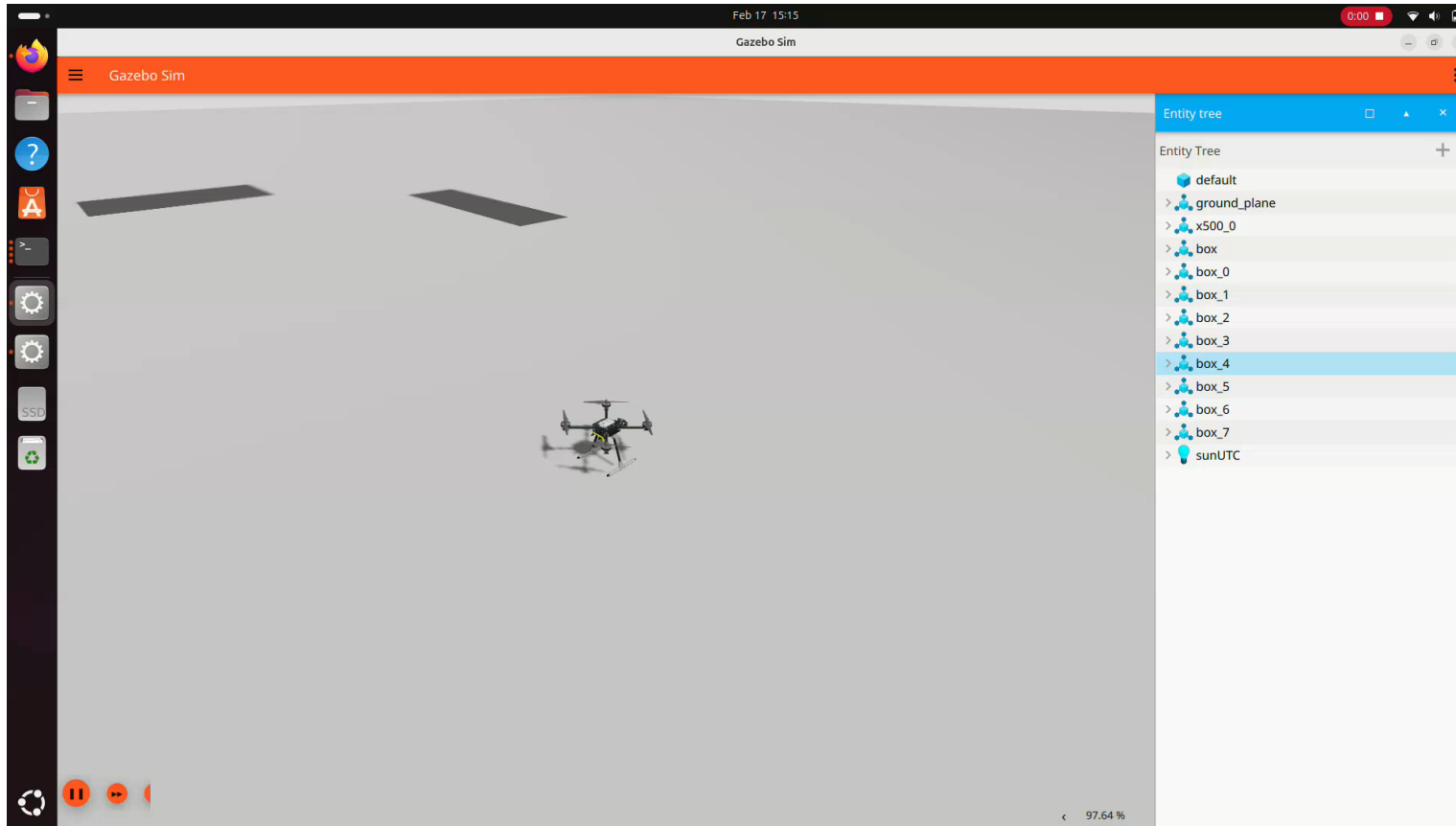
Week 1: Downloaded the necessary software that we need to test the drone in a simulated environment.



Week 2: Set up the drone in the simulated environment and create multiple obstacles in the environment



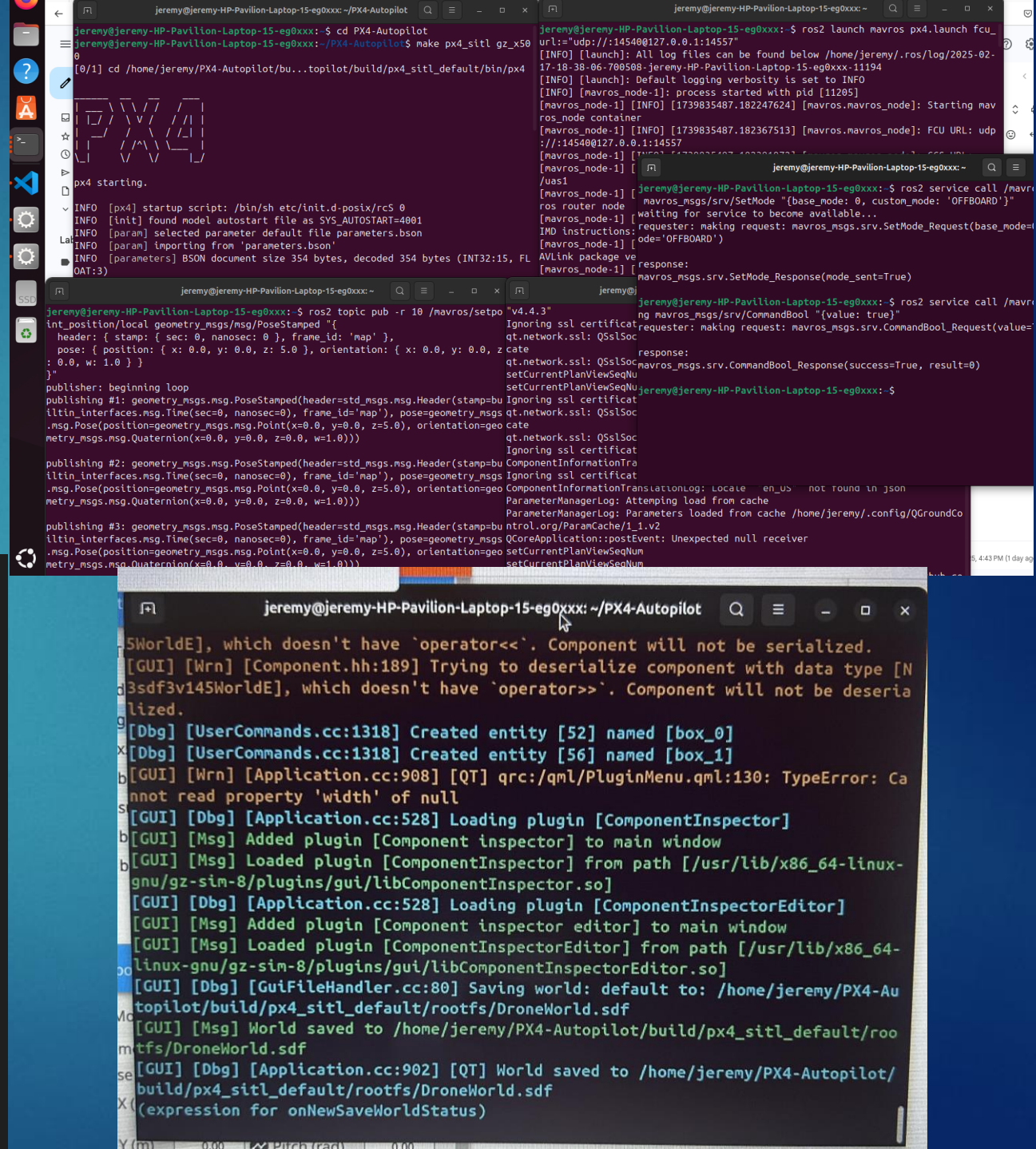
Week 3: Make the drone fly, and integrate a machine learning algorithm for obstacle avoidance



Demonstration

Code

```
5 # Safe Distance Threshold (meters)
6 SAFE_DISTANCE = 2.0
7
8 async def get_lidar_distance(drone):
9     """ Fetches LIDAR sensor data from PX4. """
10     async for distance in drone.telemetry.distance_sensor():
11         return distance.current_distance
12
13 async def avoid_obstacle(drone):
14     """ Main function to check for obstacles and adjust movement. """
15     print("Connecting to drone...")
16     await drone.connect(system_address="udp://:14540") # Connect to PX4
17
18     print("Waiting for drone to be ready...")
19     async for state in drone.core.connection_state():
20         if state.is_connected:
21             print("Drone is connected!")
22             break
23
24     async for health in drone.telemetry.health():
25         if health.is_global_position_ok and health.is_home_position_ok:
26             print("Drone is ready for navigation!")
```



```
jeremy@jeremy-HP-Pavilion-Laptop-15-eg0xxx: ~/PX4-Autopilot
jeremy@jeremy-HP-Pavilion-Laptop-15-eg0xxx: $ cd PX4-Autopilot
jeremy@jeremy-HP-Pavilion-Laptop-15-eg0xxx: ~/PX4-Autopilot$ make px4_sitl_gz_x500
[0/1] cd /home/jeremy/PX4-Autopilot/build/px4_sitl_default/bin/px4
px4 starting.
INFO [px4] startup script: /bin/sh etc/init.d-posix/rcS 0
INFO [init] found model autostart file as SYS_AUTOSTART=4001
INFO [param] selected parameter default file parameters.bson
INFO [param] importing from 'parameters.bson'
INFO [parameters] BSON document size 354 bytes, decoded 354 bytes (INT32:15, FLOAT:3)

jeremy@jeremy-HP-Pavilion-Laptop-15-eg0xxx: $ ros2 launch mavros px4.launch
[INFO] [launch]: All log files can be found below /home/jeremy/.log/2025-02-17-18-38-06-700508-jeremy-HP-Pavilion-Laptop-15-eg0xxx-11194
[INFO] [launch]: Default logging verbosity is set to INFO
[INFO] [mavros_node-1]: process started with pid [11205]
[mavros_node-1] [INFO] [1739835487.182247624] [mavros.mavros_node]: Starting mavros_node container
[mavros_node-1] [INFO] [1739835487.182367513] [mavros.mavros_node]: FCU URL: udp://:14540@127.0.0.1:14557
[mavros_node-1] [INFO] [1739835487.182367513] [mavros.mavros_node]: MAVLink package version: 2.0.0
[mavros_node-1] [INFO] [1739835487.182367513] [mavros.mavros_node]: mavros_msgs.srv.SetMode_Request(mode='OFFBOARD')
[mavros_node-1] [INFO] [1739835487.182367513] [mavros.mavros_node]: mavros_msgs.srv.SetMode_Response(mode_sent=True)

jeremy@jeremy-HP-Pavilion-Laptop-15-eg0xxx: $ ros2 service call /mavros/mavros_msgs/srv/SetMode "{base_mode: 0, custom_mode: 'OFFBOARD'}"
requester: making request: mavros_msgs.srv.SetMode_Request(base_mode=0, custom_mode='OFFBOARD')
response: mavros_msgs.srv.SetMode_Response(mode_sent=True)

jeremy@jeremy-HP-Pavilion-Laptop-15-eg0xxx: $ ros2 service call /mavros/mavros_msgs/srv/CommandBool "{value: true}"
requester: making request: mavros_msgs.srv.CommandBool_Request(value=true)
response: mavros_msgs.srv.CommandBool_Response(success=True, result=0)

jeremy@jeremy-HP-Pavilion-Laptop-15-eg0xxx: $ ros2 topic pub -r 10 /mavros/setpoint_velocity geometry_msgs/msg/PoseStamped "{header: {stamp: {sec: 0, nanosec: 0}, frame_id: 'map'}, pose: {position: {x: 0.0, y: 0.0, z: 5.0}, orientation: {x: 0.0, y: 0.0, z: 0.0, w: 1.0}}}"
publisher: beginning loop
publishing #1: geometry_msgs.msg.PoseStamped(header=std_msgs.msg.Header(stamp=builtin_interfaces.msg.Time(sec=0, nanosec=0), frame_id='map'), pose=geometry_msgs.msg.Pose(position=geometry_msgs.msg.Point(x=0.0, y=0.0, z=5.0), orientation=geometry_msgs.msg.Quaternion(x=0.0, y=0.0, z=0.0, w=1.0)))
publishing #2: geometry_msgs.msg.PoseStamped(header=std_msgs.msg.Header(stamp=builtin_interfaces.msg.Time(sec=0, nanosec=0), frame_id='map'), pose=geometry_msgs.msg.Pose(position=geometry_msgs.msg.Point(x=0.0, y=0.0, z=5.0), orientation=geometry_msgs.msg.Quaternion(x=0.0, y=0.0, z=0.0, w=1.0)))
publishing #3: geometry_msgs.msg.PoseStamped(header=std_msgs.msg.Header(stamp=builtin_interfaces.msg.Time(sec=0, nanosec=0), frame_id='map'), pose=geometry_msgs.msg.Pose(position=geometry_msgs.msg.Point(x=0.0, y=0.0, z=5.0), orientation=geometry_msgs.msg.Quaternion(x=0.0, y=0.0, z=0.0, w=1.0)))

jeremy@jeremy-HP-Pavilion-Laptop-15-eg0xxx: ~/PX4-Autopilot
[WorldE], which doesn't have 'operator<<'. Component will not be serialized.
[GUI] [Wrn] [Component.hh:189] Trying to deserialize component with data type [N3sdf3v145WorldE], which doesn't have 'operator>>'. Component will not be serialized.
[Dbg] [UserCommands.cc:1318] Created entity [52] named [box_0]
[Dbg] [UserCommands.cc:1318] Created entity [56] named [box_1]
[GUI] [Wrn] [Application.cc:908] [QT] qrc:/qml/PluginMenu.qml:130: TypeError: Cannot read property 'width' of null
[GUI] [Dbg] [Application.cc:528] Loading plugin [ComponentInspector]
[GUI] [Msg] Added plugin [Component inspector] to main window
[GUI] [Msg] Loaded plugin [ComponentInspector] from path [/usr/lib/x86_64-linux-gnu/gz-sim-8/plugins/gui/libComponentInspector.so]
[GUI] [Dbg] [Application.cc:528] Loading plugin [ComponentInspectorEditor]
[GUI] [Msg] Added plugin [Component inspector editor] to main window
[GUI] [Msg] Loaded plugin [ComponentInspectorEditor] from path [/usr/lib/x86_64-linux-gnu/gz-sim-8/plugins/gui/libComponentInspectorEditor.so]
[GUI] [Dbg] [GuiFileHandler.cc:80] Saving world: default to /home/jeremy/PX4-Autopilot/build/px4_sitl_default/rootfs/DroneWorld.sdf
[GUI] [Msg] World saved to /home/jeremy/PX4-Autopilot/build/px4_sitl_default/rootfs/DroneWorld.sdf
[GUI] [Dbg] [Application.cc:902] [QT] World saved to /home/jeremy/PX4-Autopilot/build/px4_sitl_default/rootfs/DroneWorld.sdf
(expression for onNewSaveWorldStatus)
```


Sprint #2



Software integrated with hardware components



Planned Weekly Tasks:



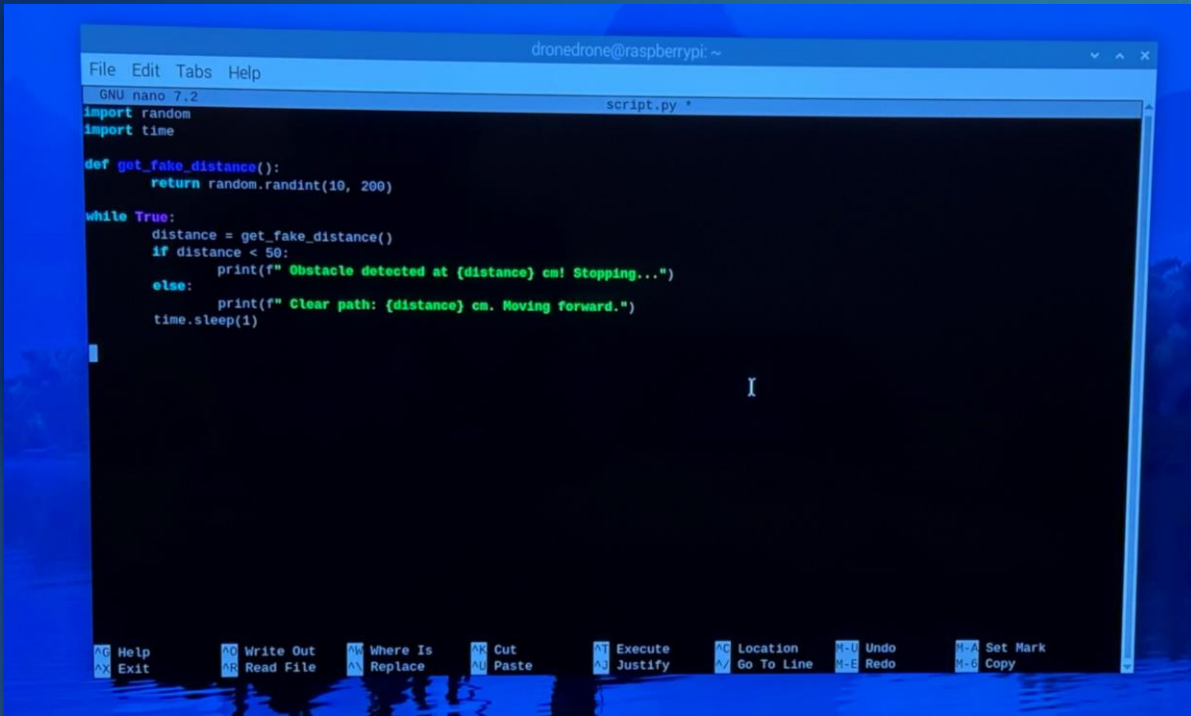
Week 1: Implemented LIDAR and camera-based object detection



Week 2: Refined machine learning algorithm to improve detection accuracy and avoidance maneuvers



Week 3: Changed the drone to make it fly autonomously and keep working on gazebo environment.



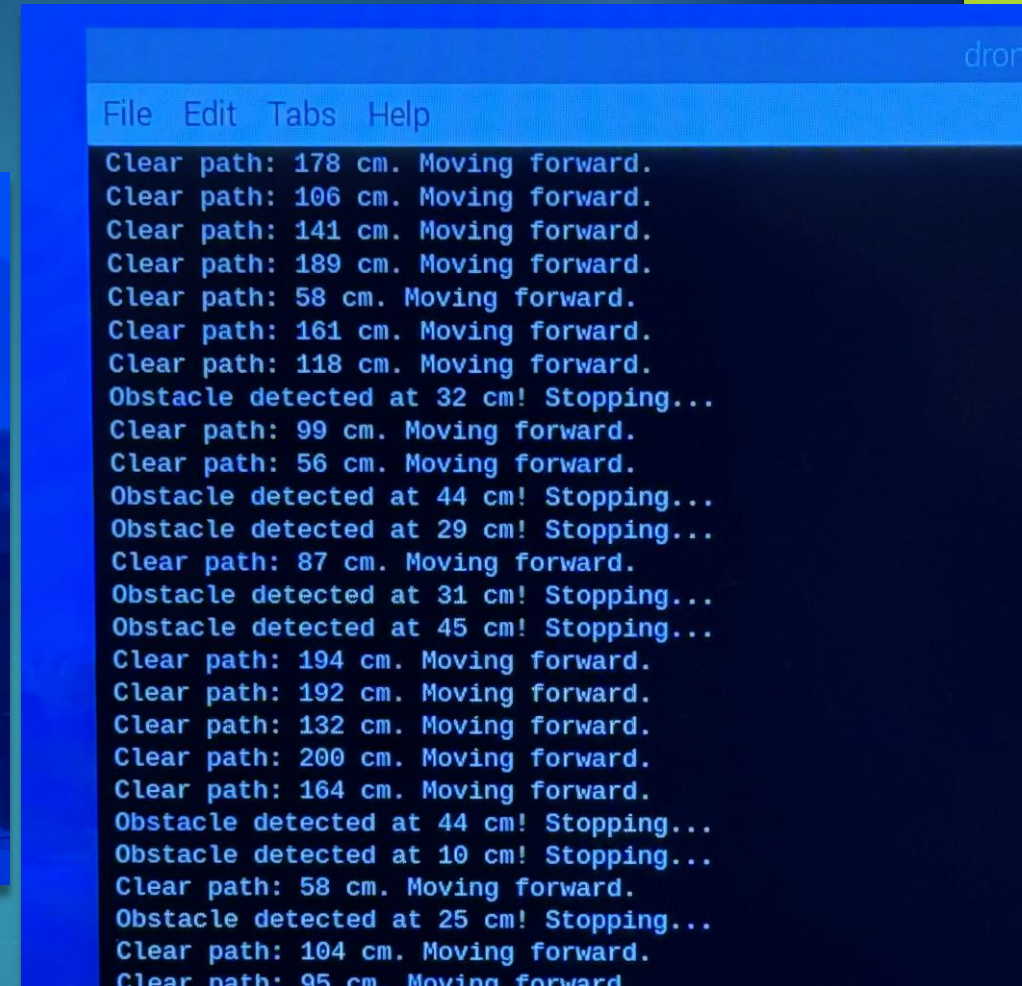
The screenshot shows a terminal window titled 'dronedrone@raspberrypi: ~'. Inside, the GNU nano 7.2 editor is open with a file named 'script.py'. The script contains the following Python code:

```
import random
import time

def get_fake_distance():
    return random.randint(10, 200)

while True:
    distance = get_fake_distance()
    if distance < 50:
        print(f"Obstacle detected at {distance} cm! Stopping...")
    else:
        print(f"Clear path: {distance} cm. Moving forward.")
        time.sleep(1)
```

The terminal has a status bar at the bottom with various icons and labels: Help, Exit, Write Out, Read File, Where Is, Replace, Cut, Paste, Execute, Justify, Location, Go To Line, Undo, Redo, Set Mark, and Copy.



The screenshot shows the output of the Python script running in a terminal window. The output consists of a series of messages indicating the distance to an obstacle and the action taken (moving forward or stopping). The messages are as follows:

```
Clear path: 178 cm. Moving forward.
Clear path: 106 cm. Moving forward.
Clear path: 141 cm. Moving forward.
Clear path: 189 cm. Moving forward.
Clear path: 58 cm. Moving forward.
Clear path: 161 cm. Moving forward.
Clear path: 118 cm. Moving forward.
Obstacle detected at 32 cm! Stopping...
Clear path: 99 cm. Moving forward.
Clear path: 56 cm. Moving forward.
Obstacle detected at 44 cm! Stopping...
Obstacle detected at 29 cm! Stopping...
Clear path: 87 cm. Moving forward.
Obstacle detected at 31 cm! Stopping...
Obstacle detected at 45 cm! Stopping...
Clear path: 194 cm. Moving forward.
Clear path: 192 cm. Moving forward.
Clear path: 132 cm. Moving forward.
Clear path: 200 cm. Moving forward.
Clear path: 164 cm. Moving forward.
Obstacle detected at 44 cm! Stopping...
Obstacle detected at 10 cm! Stopping...
Clear path: 58 cm. Moving forward.
Obstacle detected at 25 cm! Stopping...
Clear path: 104 cm. Moving forward.
Clear path: 95 cm. Moving forward.
```

Demonstration

Sprint #3

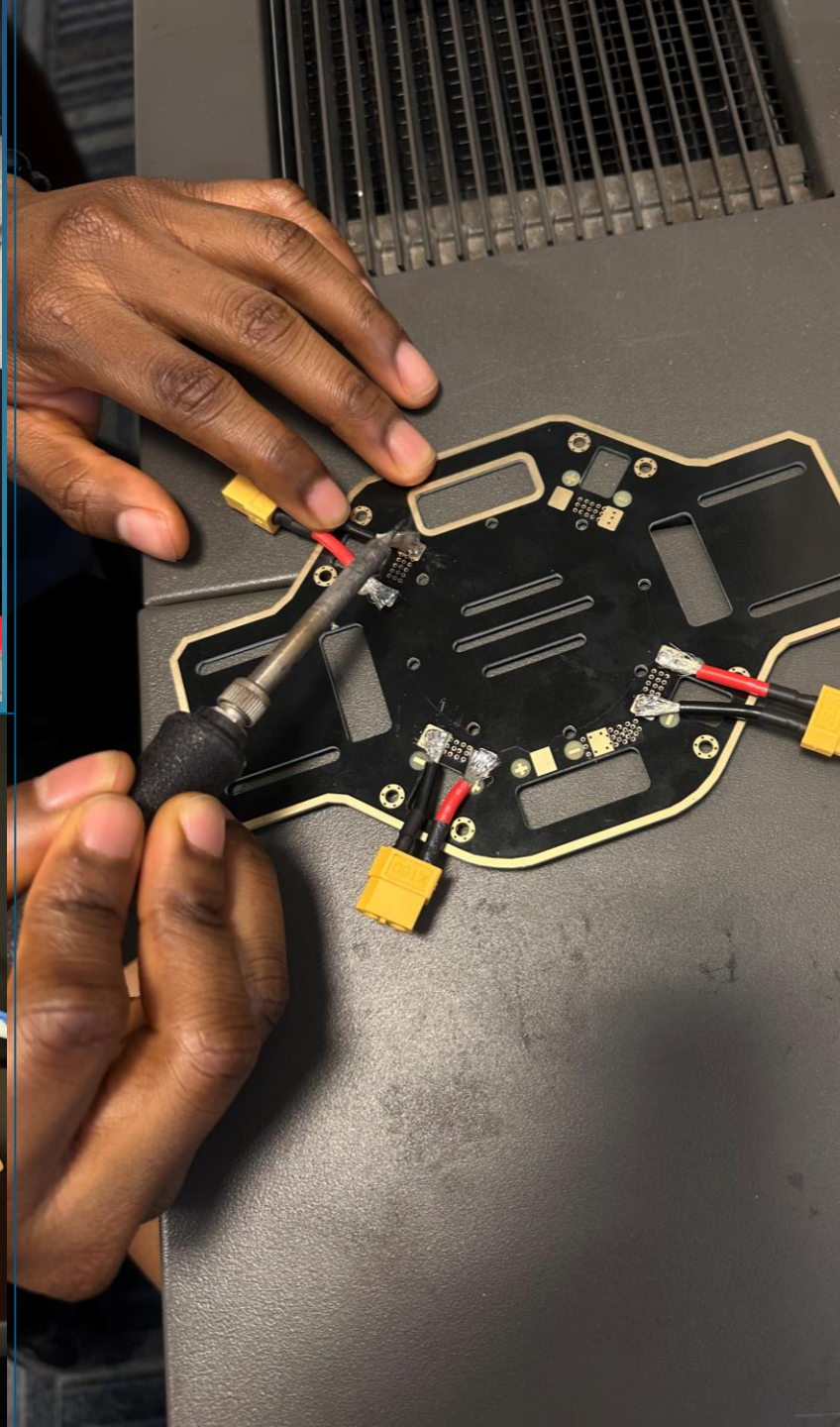
A fully integrated drone capable of navigating objects and real world testing

Planned Weekly Tasks:

Week 1: Work on Raspberry Pi and finalize drone simulation

Week 2: Assemble the drone frame and attach hardware (flight controller, battery, transmitter, receiver)

Week 3: Integrate the microcontroller GPS Raspberry Pi and power module to the drone



Demonstration



Demonstration

Final Integrated System

Drone will:

- ▶ Successfully navigate through complex environments
- ▶ Stable, autonomous operation in real world conditions
- ▶ Seamless integration of sensors, machine learning algorithms, and control system

