eYSIP2018

Flying Sensor Node

Intern : Abheet Verma
Intern : Chirag Shah

Mentor : Simranjeet Singh
Mentor : Saurav Shandilya

Duration of Internship: 21/05/2018 – 06/07/2018

2018, e-Yantra Publication

Flying Sensor Node

Abstract

Drone and IoT are two emerging technologies having a plethora of applications. Through this project we aim to bridge these technologies to develop an indoor and outdoor environment monitoring system. A drone acts as a sensor node and flies across a room using way points predetermined by the user, collecting the temperature and humidity data using on-board sensors. The sensor data along with the location and time stamp are sent to central IoT server for data visualization and analytics.

Different drones were used for indoor and outdoor environments to tackle the unique challenges faced in terms of mobility and localization in each environment. The concept of charging the drone through a landing deck was also explored in order to realize the potential of Flying Sensor Node in real world applications. The necessity of this system can be found in places such as greenhouses, where the environment needs to be monitored to schedule tasks such as irrigation, fertilizing etc.

A wireless sensor network if installed in such an area would require a huge number of nodes. Hence, resulting in increased maintenance effort and cost. Sensor nodes are static and hence can only give feedback at fixed points in the monitoring area, leading to loss in critical information. If considering a greenhouse a leakage in the irrigation system in a place not monitored by a sensor node will go undetected but a flying sensor node will be able to detect parameters at every point in its path.

The goal of this project was not only to get the flying sensor node's data but to also make it accessible to the user in a manner which is simple and intuitive. In order to achieve this goal an Internet of Things dashboard was built which allows the user to visualize sensor data and also control some basic functionalities of the drone. The platforms were different for both in-

door and outdoor flying sensor nodes, according to the capabilities of each drone. For example the dashboard for the outdoor flying sensor node has a live video stream and live tracking on maps whereas the indoor flying sensor node has control commands on the server.

Completion status

Indoor Flying Sensor Node Tasks

- Pluto X on-board sensor and peripheral interfacing using existing APIs.
- STM32f303CB peripheral interfacing using standard peripheral libraries.
- AR Drone navigation in Gazebo simulation for both Whycon and GPS.
- AR Drone communication in Gazebo simulation for IMU values.
- IoT Server for sensor data visualisation and live tracking was created on the control station.
- Establish communication between Drone and IoT Platform in simulation and reality.
- Decawave UWB tag and anchor hardware setup and configuration.
- UWB tag interfacing on Pluto X.
- Pluto X location co-ordinates transmission and reception using Multi Wii Serial Protocol.
- Complementary Filter design for localization data (Needs Improvement).
- Payload Testing of Pluto X.
- Waypoint planning and navigation with data logging using UWB and Whycon.
- Landing of Pluto X on designated charging area.

Outdoor Flying Sensor Node Tasks

- Quadcopter assembly and RC remote testing.
- Pixhawk setup and configuration using Mission Planner and Q Ground Control was attempted.
- Pixhawk control using mavros, dronekit and companion computer was attempted.
- Navio 2 setup for quadcopter was done on RPi.
- Quadcopter configuration and calibration was done using Mission Planner.
- IoT Server for sensor data visualisation, live tracking and video streaming was created on the RPi.
- Location logging using GPS was implemented.
- Temperature and humidity value from DHT-22 is communicated to the flight controller and is logged.
- Mounting structure for Sensor node was designed.
- Battery consumption was analyzed and appropriate power source was selected.
- Fail safes like Geofence,3D Fix and other Prearm Checks like Barometer,Gyro etc were removed to enable indoor testing.
- Stabilize mode of operation was tested indoors using RC transmitter.

1.1 Hardware parts

Pluto X

- Purpose : Indoor Drone based on STM32f303CB
- Vendor : Drona Aviation

RPI

- Purpose : RPI 3B is the processing unit for the outdoor quadcopter when using Navio 2.
- Product Link : RPI 3B
- Datasheet : RPI 3B Datasheet

Arduino Nano

- Purpose : Arduino Nano is used as sensor node.
- Product Link : Arduino Nano
- Datasheet : Arduino Nano Datasheet

Navio 2

- Purpose : Navio 2 is an autopilot hat for RPi that has all the sensors and controllers onboard like Dual IMU,Barometer,GNSS receiver,RC I/O co-processor etc.
- Product Link : Navio 2
- Product Brief : Navio 2 Documentation

Decawave DWM1001

- Purpose :Indoor Localization
- Product Link : DWM 1001
- Product Brief : DWM 1001 Resources

LiPo Battery 6200 mAh, 40C & 11.1 V

- Purpose : Powering the quadcopter.
- Product Link : Orange Battery
- Product Brief : Orange Battery Specifications

Pixhawk

- Purpose : Flight controller
- Product Link : Pixhawk
- Product Brief : Pixhawk Resources

Telemetry Radio 433MHz for Navio 2

- Purpose : Communication between ground control station and quadcopter.
- Product Link : Mrobotics Telemetry Module

Telemetry Radio 433MHz for Pixhawk

- Purpose : Communication between ground control station and quadcopter.
- Product Link : Pixhawk Compatible Telemetry Module

DHT-22

- Purpose : Sensor for obtaining temperature and humidity data.
- Product Link : DHT-22
- Product Brief : DHT-22 Datasheet

Brushless Motor Speed Controller ESC 30A

- Purpose : Electronic speed controller for BLDCs.
- Product Link : ESC Specifications

RC Brushless Motor 2212 1000KV With Soldered Banana Connector

- Purpose : BLDCs with propellers to provide motion to the quadcopter. Various movements are possible by varying the direction of rotation of propellers & by altering the speed of the motors.
- Product Link : BLDC Specifications

1.2 Software used

1.2.1 Indoor

- Java

  For Windows:Link
  For Ubuntu:Link

  Use Java 8 only as it is required for Cygnus.

- Cygnus IDE

  This integrated developement environment is necessary for flashing code to the pluto drone.Install cygnus from here.Follow the instructions to setup Cygnus on you system.Enjoy coding!!!!

- ROS(Indigo)

  Robot Operating System (ROS) is robotics middleware (i.e. collection of software frameworks for robot software development).Ros is required for interfacing Pluto and PlutoX.To know more about ROS visit here.

  The main ROS client libraries (C++ and Python) are geared toward a Unix-like system, primarily because of their dependence on large collections of open-source software. Hence these client libraries require Linux operating system.
  You **must** install the **ROS-Indigo in Ubuntu 14.04** on your PC/Laptop.

  Follow the installation instructions from here.

- pluto_drone

  Metapackage to control the plutodrone via services and topics. This package establishes a connection between laptop and the ESP module on the drone using a socket. Thereby using MSP protocol we can send instructions to the drone and receive data from onboard and external sensors. wiki

- Roslibjs

  Roslibjs is the core JavaScript library for interacting with ROS from the browser. It uses WebSockets to connect with rosbridge and provides publishing, subscribing, service calls.
  The topics and services are used in the form of JSON. A connection string is used to establish a connection and then get access to the topics.

Download link: min :: full Source:Github Wiki:Roslibjs

- Rosbridge_suite

  Rosbridge provides a JSON API to ROS functionality for non-ROS programs. There are a variety of front ends that interface with rosbridge, including a WebSocket server for web browsers to interact with.

  Rosbridge suite allows access to the topics and services on the server.Hence topics can be subscribed and published upon to give commands to the robot.
  For installation visit this link.

Figure 1.1: Server for indoor drone

1.2.2 Outdoor

- Ground Control Station

  A ground control station (GCS) is a land-based control centre that provides the facilities for human control of Unmanned Aerial Vehicles (UAVs or "drones").
  A ground control station helps in sending mission details,setting essential parameters,configuring the sensors,getting real time values of sensors etc.

  You may install one the below GCS:

  Mission Planner::Link(only for Windows)
  QGroundControl::Link

- Python

  Python examples were available for Raspberry Pi in the documentation for navio 2. The GPS example was used to obtain latitude, longitude and fix status. Python was also used for obtaining sensor data through USB port from the arduino nano (Serial library was used in this case). The server was written using flask which again is based on python. Socket programming was also implemented in the flask server to update position of drone in the background. Also python was used to fetch and update data in the Sqlite3 database

  We used Python 2.7 for this project. **Always use same python version for every library as it may cause errrors**.
  For installation visit here.

- Emlid Image for Raspberry Pi

  Raspberry Pi was flashed with a custom image of raspbian that comes with pre installed Ardupilot, Ros and other necessary packages required for automated vehicles(Plane,Rover,Copter). The image does not have a GUI and should be used by connecting through SSH. Reconfiguring the image to enable GUI is not recommended as certain issues in the working were observed such as git-hub commands not working and autopilot parameters not updating.
  Download the image and flash it on a memory card using Etcher
  Run and install Etcher using admin rights.
  Select the archive with Image and drive location.
  Click Flash!!.This may take a few minutes.

- Motion

  Motion is a highly configurable program that monitors video signals from many types of cameras. It was used for displaying live video feed on the web server coming from RPi Camera mounted on the Raspberry Pi. Setting it up on raspberry pi can be understood from this link. If the video doesn't start as soon as the server starts and a gray screen is visible visit this discussion

- Flask

  Flask is a micro web framework written in Python. It is classified as a micro-framework because it does not require particular tools or libraries. It is also very intuitive to use. Also as it is based in python, it is compatible with examples provided by emlid for navio 2. Read more about flask from here.

- Sqlite 3

  Sqlite 3 is the database which was selected to log data locally. Follow this tutorial to understand setting up an IoT server using flask and displaying data logged in an Sqlite 3 database.

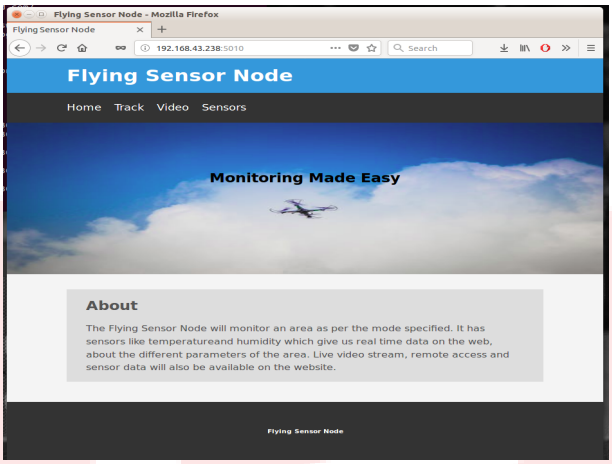


Figure 1.2: Home page

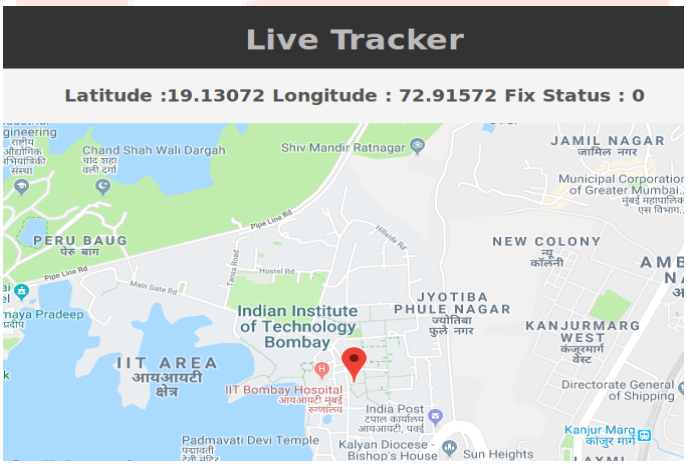


Figure 1.3: Live tracking

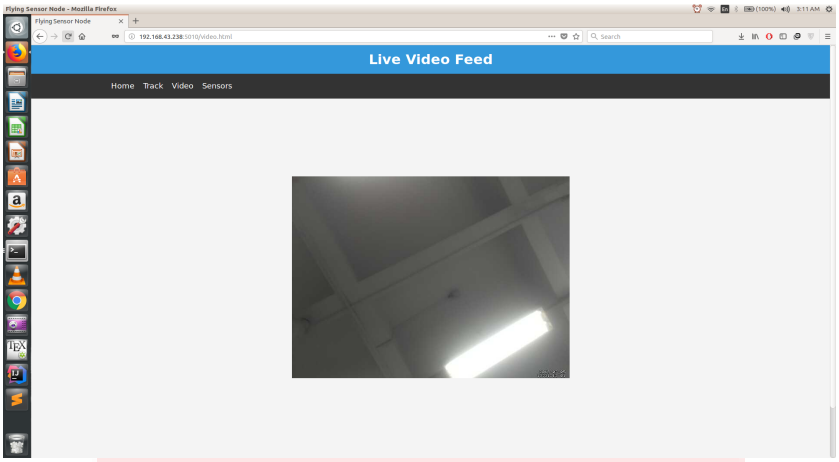


Figure 1.4: Live Video Feed

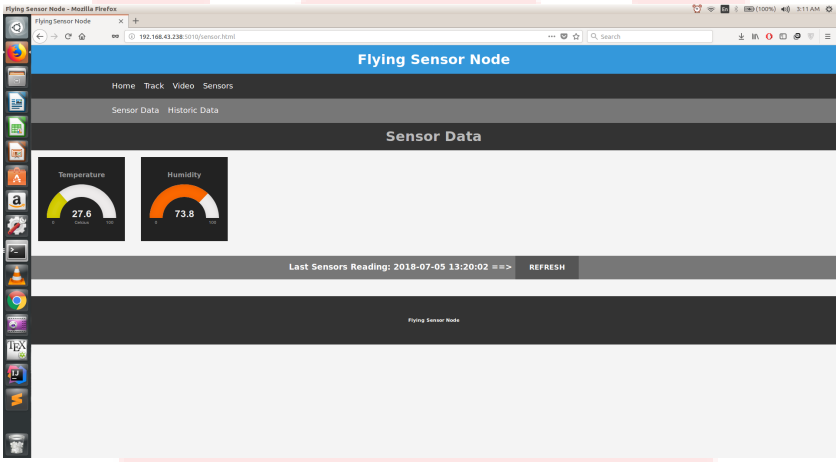


Figure 1.5: Sensor data

1.3 Assembly of hardware

The assembly for both indoor and outdoor drones is explained in this section.

Indoor Flying Sensor Node

DWM1001 Module Setup

- One module is used as a tag on the drone.Connection between tag and PlutoX is shown as shown in the Figure-1.6 below.

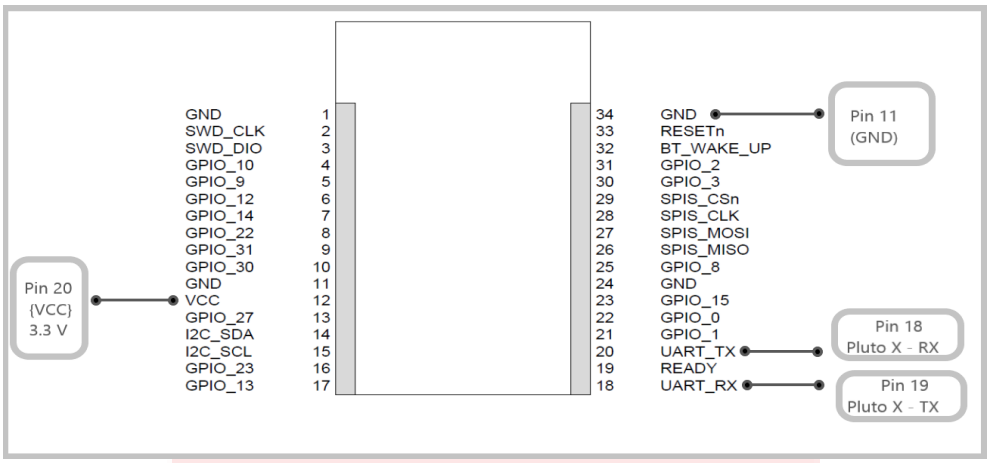


Figure 1.6: Pinout of DWM Module

- Four modules are used as anchors on the wall.Each one of them requires a power source of 3.3V.
- The modules are configured using the official Decawave Android App
- The instructions for the same are explained in the android application in great detail.

Pluto X

- The pin mapping of the shield with respect to the STM32f303CB microcontroller on Pluto X is as shown in the table in the Figure 1.7.

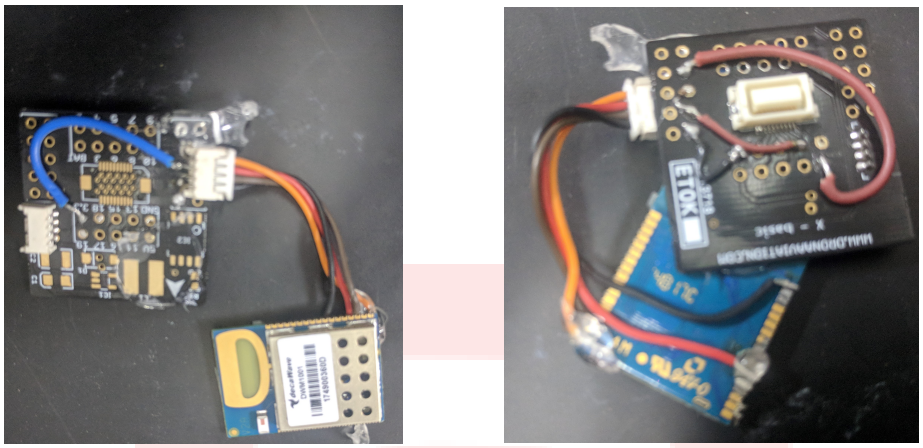
	S. No	Pin	FT		STM pir	STM r	TIM I	TIM II	ADC	DAC	IR	I2S2	USART3	UART2	I2C1	OPAMP4	debug
POWER		1	-	VBAT		-											
-	12	2	FTf	GPIO	34	PA13	T4C3	T16_CH1N			Y						IO
-	13	3	FTf	GPIO	37	PA14	T8C2							TX			CLK
M7	14	4	TTa	GPIO	21	PB10	T2C3						TX			VINM	
M8	15	5	TTa	GPIO	22	PB11	T2C4						RX			VINP	
I2C		6	TT	SDA	46	PB9	-	-	-	-	-						
I2C		7	TT	SCL	45	PB8	-	-	-	-	-						
DAC	19	8	TTa	GPIO	15	PA5	T2C1		A2_IN2	Y							
M6	18	9	FT	GPIO	39	PB3	T2C2	T8C1~						TX			SWO
SPI		10	FT	SS	29	PA8	T1C1										
POWER		11	-	GND		-											
POWER		12	-	GND	-												
DAC		13	TTa	GPIO	14	PA4	T3C2		A2_IN1	Y						VINM	
SPI		14	TTa	SS	25	PB12			A4_IN3			WS				VOUT	
SPI		15	TTa	SCK	26	PB13	T1_C11		A3_IN5			CK					
SPI		16	TTa	MISO	27	PB14	T1_C2~		A4_IN4			EXT_SD					
SPI		17	TTa	MOSI	28	PB15	T1_C3~	T15_C2,1~	A4_IN5			SD					
UART		18	TTa	RX	13	PA3	T2_C4	T15_C2	A1_IN4					RX			
UART		19	TTa	TX	12	PA2	T2_C3	T15_C1	A1_IN3					TX			
POWER		20		3.3V		-											

LEGEND	
	Shared pins; Use with care
	SWD debug pins, but can be used freely
	Power pins

GPIO	16
Anaglog channel	8
Timers	11
Uarts	2
SPI	1
I2C	1
DAC channels	2
Debug port	SWD

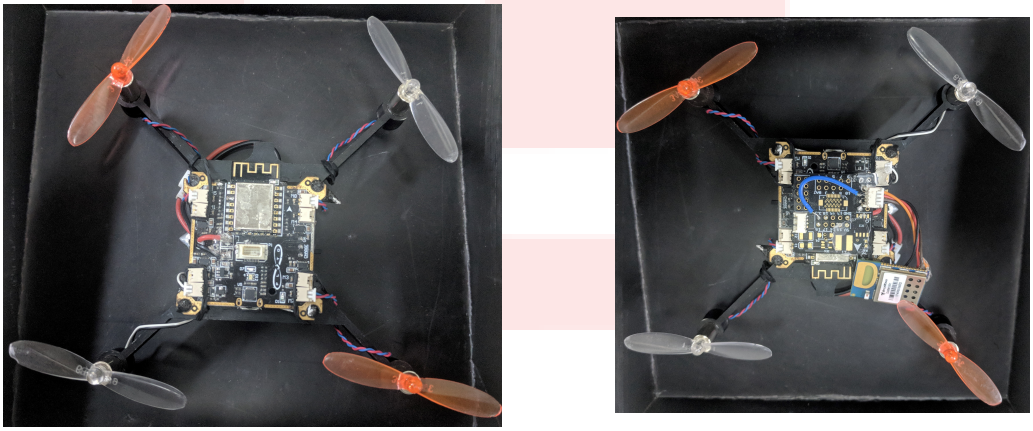
Figure 1.7: Pin Mapping of Pluto X Shield

- After attaching the tag on the gpio shield of Pluto X, place the shield on the Pluto X as shown in the Figure 1.8 and 1.9.



(a) Top View of Pluto-X shield (b) Back Side view of Pluto-X shield

Figure 1.8: Pluto X Shield



(a) Pluto X without shield (b) Pluto X with shield

Figure 1.9: Pluto X Shield Setup

- A whycon marker is also added on top of the Pluto X as shown in the Figure 1.10 below to enable landing on charging area using an overhead camera.

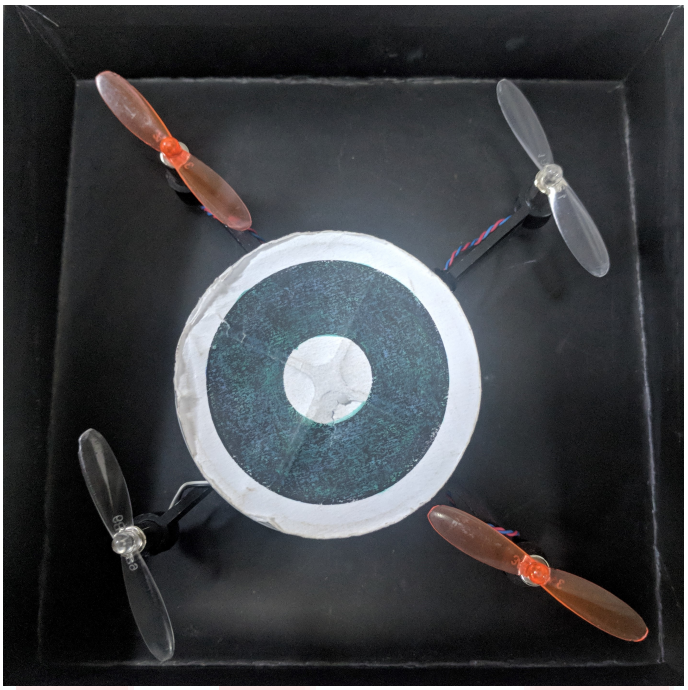


Figure 1.10: PlutoX with Whycon Marker Mounted

Outdoor Flying Sensor Node

Auto Pilot Assembly

- Raspberry Pi, Navio 2, GPS antenna, Telemetry trans-receiver, battery monitor, remote receiver through PPM encoder and electronic speed controller inputs are connected as shown here

  Note: The PPM encoder allows to encode up to 8 PWM (pulse width modulated) signals into one PPM (pulse position modulation) signal.

- The other parts of the drone such as motors and propellers are assembled in accordance to this guide

- A three cell 11.1V, 6200mAh, 40C lithium polymer battery is used to power the drone

- The completely assembled quadcopter is as shown in the Figure 1.11 .

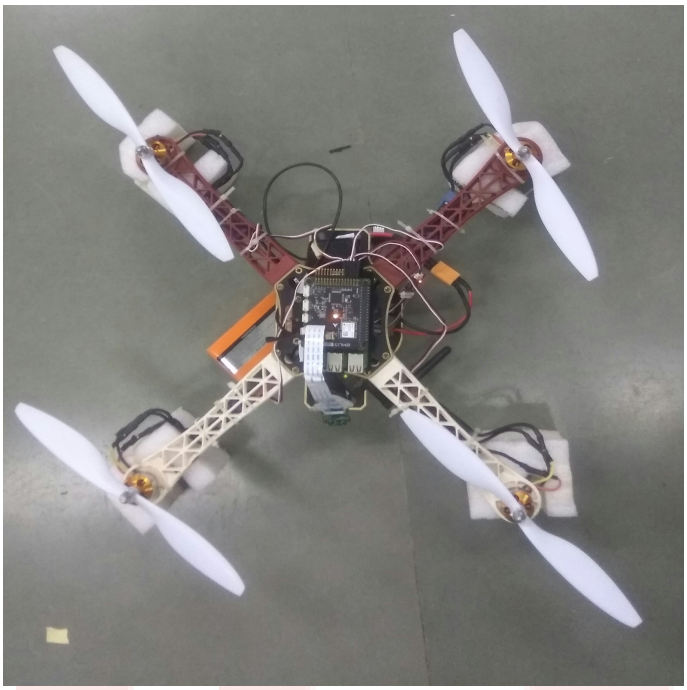
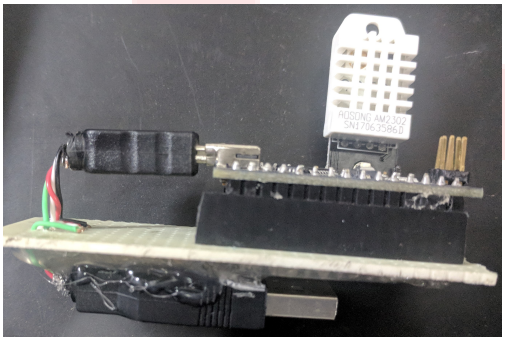
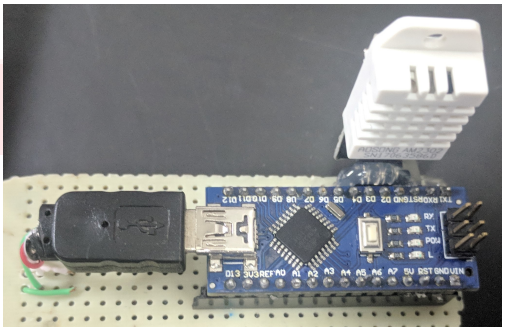


Figure 1.11: Outdoor Drone/Quadcopter

- The sensor node comprising of the arduino nano, DHT-22 and USB connector is as shown in the Figure 1.12.



(a) Sensor Node Side View



(b) Sensor node top View

Figure 1.12: Sensor Node Module

- The sensor node needs to be plugged in any of the USB ports of the raspberry pi on the drone.

1.4 Software and Code

Github link for the repository of code

Outdoor

- **Arduino Code** - Contains code to interface DHT-22 using arduino nano on a Raspberry PI.
- **Drone-Server** - Contains server and related files for drone and copy the folder on Raspberry PI.
- **Ardupilot Setup** - link

Indoor

- **Documents** - Contains API for interfacing PlutoX board.
- **Codes** - Contains testing codes for PlutoX and RF.
- **Pluto-X Navigation-Whycon,Final,Navigation** - Different Navigation scripts for PlutoX
- **PlutoX Firmware** - Firmware Changes for MSP protocol
- **Server** - Establishing server for logging data and controlling drone remotely
- **pluto_drone** - Contains changes in navigation packages

Use and Demo

Outdoor

- After setting up the raspberry with the custom image and configuring the quadcopter using mission planner do the following :-
- **Step 1** - Change directory to Drone-Server
- **Step 2** - Run command "sudo modprobe bcm2835-v4l2" to get video on web server.
- **Step 3** - Run command "sudo python SensorLog.py" to get real time sensor data i.e. temperature and humidity

- **Step 4** - Run command "sudo python GPS.py" to get real time sensor data i.e. GPS (only if the drone is outdoors)
- **Step 5** - Run command "sudo python app.py" to start the server
- **Step 6** - If the IP address of the raspberry pi is 192.168.43.238 open a browser on a device connected to the network and enter the URL 192.168.43.238:5010 to get access to the IoT platform.

Indoor

After setting the DWM1001 module,whycon on the drone and the overhead camera follow the below steps.

- **Step 1** - Make changes to the serial_msp.cpp and flash the firmware.
- **Step 2** - Add changed files to pluto_drone metapackage
  - To add your own MSP headers first define it in the firmware in the serial_msp.cpp file
  - Add the new MSP header in the service request.
  - Receive the data in global declared parameters in the Plutoservice.srv file
- **Step 3** - Run the data_via_rosservice.py file to get coordinates and other data
- **Step 4** - Run the drone_comb.launch file and on another terminal run the navigation script.
- **Step 5** - Run the rosbridge_websocket launch and open comm.html in browser to control the drone and get data
- **PS** - Play with the code to understand it better.

Youtube Link of demonstration video

1.5 Future Work

1.5.1 Outdoor

- Manual and Autonomous testing of drone in outdoor environment using Ground Control Station.

- Full fledged IOT server for large scale use using internet.
- Buildng Charging Dock for the drone.
- Using Dronekit and similar libraries for controlling drone using server.

1.5.2 Indoor

- Building a Charging Dock for PlutoX.
- External Sensor(Temperature,Humidity) Interfacing for PlutoX.
- Designing a better noise reduction filter so that navigation is much smoother.
- On-board Computing of path to reduce latency and improve performance.

1.6 Bug report and Challenges

1.6.1 Outdoor

- No heartbeat received from the Flight Controller Unit. Check if the correct mode of connection is selected and the address(UDP,TCP/IP) is correct. Also check the telemetry module for any damage on both transmitter and receiver end.
- Arming the quadcopter without GPS fix is generally not permissible but by changing failsafe parameters this can be bypassed.
- Bad logging error due to SD card. Try formatting the SD card again.Make sure to select FAT32 as the memory allocation type.
- Telemetery issues with Pixhawk flight controller. Check the telemetry module for any damage on both transmitter and receiver end.

1.6.2 Indoor

- Understanding the firmware of Pluto X. The firmware itself contains huge amount of code and to understand the flow of code was a little difficult.A more clear reference can be taken from the Cleanflight repository on github.

- Reconfiguring the firmware to accept the coordinate values coming from DWM1001. It was done using existing apis as shown by Drona Aviation team. .
- Understanding Multi Wii Serial Protocol of data transfer from drone to control station and vice versa. The protocol was studied using on-line resources and wiki page mentioned in Bibliography.
- Reconfiguring the ROS communication package for PlutoX 'plutodrone' to accept new MSP headers.
- Designing a filter to reduce noise in the coordinates.A complimentary filter fusing raw roll,pitch values with the coordinates was tried but a better filter can be used to make navigation smoother.
- Navigating drone using PID and then switching dynamically between Whycon marker and Ultra Wide Band for better navigation.
- Fine tuning of PID for smooth control of drone.
- External sensor i.e. DHT-22 interfacing was attempted but was later neglected for the following reasons :-
  - The payload i.e.the RFID tag, itself was leading to increased strain on the drone, and incorporating the sensor on the drone increases the weight further which lead to difficulty in the flight of the drone. (Reduced battery life and difficulty in taking off)
  - The DHT-22 is a sensor based on one wire protocol, and libraries and documentation for it are not readily available for ARM STM32f303CB. Also making changes in the firmware for the same, were attempted in the initial stage but led to random behavior of the drone. The source of random behavior was due to mistakes in the firmware by Drona Aviation which were later rectified. But the sensor interfacing plan was dropped due to the increased payload mentioned earlier.
  - On-board temperature sensor was later used to do the environment monitoring, but this data was inaccurate due to heat dissipation of on-board components.

Bibliography

[1] Mastering ROS for Robotics Programming - Lentin Joseph

[2] Ros tutorials- Link

[3] Programming Robots with ROS: A Practical Introduction to the Robot Operating - Brian Gerkey, Morgan L. Quigley, and William D. Smart

[4] ROS cheatsheet

[5] Navio Documentation-link

[6] Linux-Documentation

[7] Python-Documentation

[8] Multi Wii Serial Protocol-link

[9] Navio2 drivers-Github