IMPLEMENTATION AND VALIDATION OF GAUSSIAN PROCESS MODEL REFERENCE ADAPTIVE CONTROL FOR FIXED WING UNMANNED AERIAL SYSTEMS

By

Sri Theja Vuppala

Bachelor of Technology in Mechanical Engineering
Jawarharlal Nehru Technological University
Kakinada, Andhra Pradesh
India
2012

Submitted to the Faculty of the
Graduate College of
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
May, 2016

IMPLEMENTATION AND VALIDATION OF GAUSSIAN PROCESS MODEL REFERENCE ADAPTIVE CONTROL FOR FIXED WING UNMANNED AERIAL SYSTEMS

Thesis Approved:

Dr. Girish V. Chowdhary

Thesis Advisor

Dr. Jamey D. Jacob

Committee Member

Dr. He Bai

Committee Member

Name: SRI THEJA VUPPALA

Date of Degree: MAY, 2016

Title of Study: IMPLEMENTATION AND VALIDATION OF GAUSSIAN PROCESS MODEL REFERENCE ADAPTIVE CONTROL FOR FIXED WING UNMANNED AERIAL SYSTEMS

Major Field: MECHANICAL AND AEROSPACE ENGINEERING

Abstract: Over the last couple of decades, rapid development of unmanned aerial systems (UAS) has been observed. UAS are becoming an integral part of various industries such as agriculture, communications, defense, first response, geophysical surveys. This wide range of applications over different industries demands a number of mission specific vehicle platforms. The platforms must be reliable in all environments as well as in the presence of various uncertainties. Presently, the UAS that are flown autonomously rely on extensive manual tuning of control parameters. The control parameters are platform specific, hence transferring the controllers from one platform to another, is time consuming and requires extensive testing against human errors. A detailed approach to the development of an adaptive, platform independent controller which leverages Bayesian Non-parametric approach towards the adaptive control was performed in this thesis. Hardware-in-the-loop simulation is one of the effective methods for the verification of the overall control performance and safety of the UAS before conducting actual flight tests. We had developed Hardware-in-the-loop (HITL) framework to test the developed. This was done by actual implementation into different aircraft platform. Extensive testing in the HITL environment was done and results from HITL tests as well as flight test results are presented.

Acknowledgments

I would like to express my sincere gratitude to my thesis advisor and academic mentor, Dr. Girish V. Chowdhary for his continuous support and motivation during my Masters study and research. The knowledge he shared with me, the kind of trust he had instilled in me, the degree of patience he embraced when correcting my mistakes, are some things that made me admire him a lot. I am extremely fortunate to have an advisor like him. I would like to express my thanks to the Committee Members Dr. Jamey Jacob and Dr. He Bai for their support and inputs towards completion of the work.

I would like to thank all of the members of the Distributed Autonomous Systems Laboratory. Most of what I have learned has been due to the interactions and discussions with the lab members. I would like to thank Dane Johnson, Andrew Cole who helped me ease into the research at its inception and also Girish Joshi, Logan Washbourne, Nolan Repogle, Sesha Talapa Sai Radganti for the support towards the progress of the work.

I am extremely thankful to my friends Gopal Koya, Nakul Babu Maddipati, Nandu Kumar Merupula, Noel Daniel Gundi, Rakshit Allamraju, Suresh Babu Myneni, Suryakiran Chavali and Sowmya Pachipenta for their encouragement and support during my graduate study.

I would like to acknowledge, with deepest gratitude, the support and immeasurable love of my family. My mother Usha Rani Vuppala and brother Sri Harsha Vuppala have supported me at each and every phase of my career. They gave me freedom to take my own decisions and gave up many things for me to chase my dreams. I can never be grateful enough to such an amazing family.

I would like to dedicate this work to my father Ramesh Gupta Vuppala, who is my role model and inspired me towards becoming a Mechanical Engineer.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Over the last decade, Unmanned Aerial Systems (UAS) has seen rapid growth. There is a rapid growth in the technology relating to the UAS. UAS have already been a part of many industries where it applications include security, search and rescue, monitoring, disaster management, crop management, geophysical surveys and many more. To handle novels tasks with unique platforms, the onbaord control system must be robust , highly reliable and allows for deep modification of functionality. The Commercial off the shelf (COTS) autopilots are categorized into two groups : open source and closed autopilots. The former is available at a low price where the latter are relatively expensive. Unfortunately, neither of them allow modification for higher functionality.

The wide range of applications of UAS mentioned has resulted in development of numerous mission specific Unmanned Aerial Vehicles (UAV) platforms. These novel platforms must operate reliably in various environments and in presence of uncertainties. The current practice of flying the UAVs autonomously relies on extensive manual tuning of the UAV autopilot parameters or time consuming approximate modeling of the dynamics of the UAV. These methods lead to excessive development time. However, controllers cannot be simply transfer from one platform to another, which leads to each platform being tuned independently of the others in order to achieve desired performance. This process can be time intensive and a lot of money is involved. This work tackles the problem of efficiently transferring controllers between different UAV platforms using adaptive control.

The problem of control transfer is framed using the ideas of adaptive control and Rapid Controller Transfer (RCT). The primary goal is to transfer the autopilot with minimal effect on the performance from one platform to another. The main advantage of RCT is reduction in time spent on developing control system from every novel platform. The proposed method uses a new class of data driven adaptive control algorithm. It leverages Bayesian non-parametric approach to adaptation.

1.2 Outline of Contributions

The contributions of this thesis are

- Implementation of Gaussian Process Model Reference Adaptive Control (GP-MRAC) in fixed wing aircrafts, to demonstrate Rapid Controller Transfer
- Validated autopilot Stabilis, developed in house, with integrating it in the Hardware in the Loop Environment (HITL) and with real world flight testing.

1.3 Outline of the Thesis

The thesis is organized as follows. Chapter 2 will discuss the related works in the area of adaptive control and its aerospace applications. Chapter 3 will outline a brief overview of flight dynamics and the control scheme implemented using GP-MRAC formulation. Chapter 4 will discuss the design and construction of Plug-and-Adapt^{TM} Autopilot STABILIS and integration into Hardware in the Loop testing environment.

Chapter 5 will address the results that were gathered from both Hardware in the Loop Simulations and the real world testing. Chapter 6 will discuss the conclusions and future work.

Chapter 2

Related Works

The work presented in this thesis, mainly focuses on a platform independent autonomy module featuring adaptive control. This chapter presents a proper understanding of adaptive control. The usage of adaptive control techniques for transferable control has not been widely studied. Adaptive control has proved to be a reliable solution for modeling errors and system uncertainty. Adaptive Control can be classified into two categories, the first being used to track the error to modify controller parameters, whereas, the second one approximates the difference between the assumed reference model and the actual system dynamics, then uses the approximation to control the plant.

Adaptive Control has been extensively studied for Aerospace applications. First flight experiments with adaptive control systems were performed in the decade of 1960, however, without proper analysis of closed loop stability. This lead to a fatal crash of the X15A in the year 1967 and as a result, adaptive flight control systems were pushed out of focus for quite some time. Later in 1980, after Narendra provided a mathematical stability proof for MRAC system [1]. Further, important results of MRAC were summarized by Narendra and Annaswamy [2]. Following this, emphasis was put on performance and robustness of adaptive systems in presence of uncertainties and unmodeled dynamics, which resulted in various modifications of the parameter update equations.

There have been many MRAC formulations that have sought to solve some of the issues that are associated with such methods. L1 adaptive control is a well known MRAC formulation that has been widely used in aerospace guidance and control applications [3, 4], as well as others [5, 6]. The benefits of L1 adaptive control claimed by the authors are fast and robust adaptation, analytically computable performance bounds and excellent performance with minimal flight control design cost [7]. The L1 formulation differs from classical MRAC methods through the use of high adaptive gains with an input filter. The high adaptive gains help ensure the adaptive controller is responsive enough to track the uncertainty point wise in time. Another MRAC formulation known as Intelligent Excitation, seeks to mitigate the need to inject Persistent Excitation (PE) in the reference input while guaranteeing parameter convergence [8, 9]. This is done by injecting excitation only when the tracking error exceeds a desirable limit. Although this MRAC formulation reduced the need for excitation, PE is still used, thus control effort is wasted. Another MRAC formulation called Derivative Free MRAC (DF-MRAC), was presented by Yucelen et al. [10]. DF-MRAC relaxes the assumption of constant ideal weights that classical MRAC methods use and featuring a time varying set of weight parameters. This feature of the algorithm allows for a time varying system to be modeled in the face of uncertainty. The DF-MRAC formulation is shown to be uniformly ultimately bounded, and the error is shown to be ultimately bounded exponentially [11].

The most widely used technique for estimating the system uncertainty in the context of indirect MRAC methods is the neural network. Neural networks used in conjunction with adaptive control techniques are used extensively in flight control and guidance [12–16]. This formulation guarantees the existence of a set of ideal weights that guarantees optimal approximation of uncertainty, which is implied by the universal approximation property of neural networks.

There are two types of neural networks that are used in adaptive control, single hidden layer (SHL) neural networks (NN) and radial basis function (RBF) neural networks. The idea of Controller Transfer is first presented using a neural network based MRAC formulation [17], but it was not explicitly studied. Later, Chowdhary et al. extended neural networks into a formulation of MRAC that uses both recorded and instantaneous data to concurrently learn, hence called Concurrent Learning MRAC (CL-MRAC). The most notable feature of CL-MRAC is its ability to leverage the advantages of both direct and indirect adaptive control to mitigate the need for PE [18]. CL-MRAC was used for controller transfer on indoor quadcopters with promising results [19].

However, both SHL and RBF neural networks have disadvantages. One of the more notable disadvantages of RBF neural network based approaches, is that the number of centers and hyperparameters must be allocated a-priori over the operating domain. Thus controllers operating outside of the intended domain experience degraded performance [20, 21]. Also, SHL neural networks performance can suffer from getting stuck in local minimum [22].

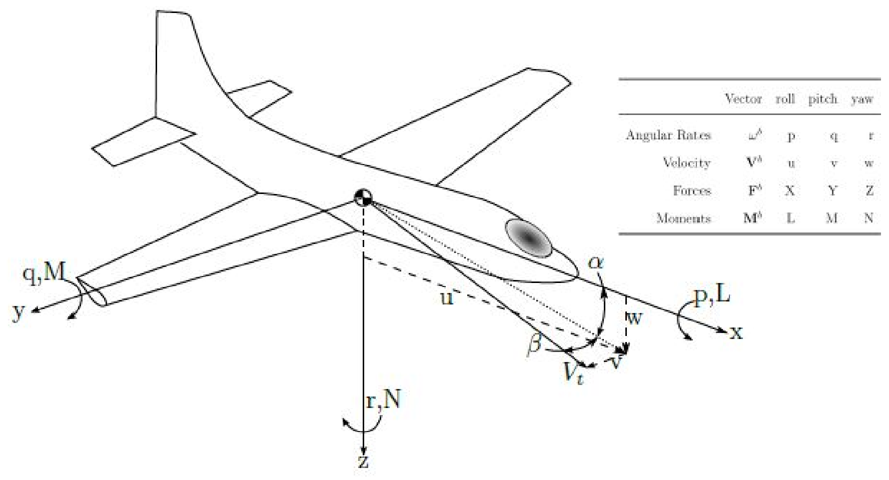
Unlike RBF Neural Networks, Gaussian Processes (GPs), can cover the entire operating domain, by dynamically allocating kernel locations based on a fixed budget of kernels. As GPs are Bayesian in nature, the model itself provides a quantified confidence metric in its predictions via the predictive variance. Previously, using online GPs to model uncertainty was computationally expensive due to large data sets. However, largely due to the derivation of sparse, online Gaussian Processes by Casato et al. [23], GPs were recently proposed as a nonparametric approach to modeling dynamical uncertainty in an adaptive controller [21]. Recently, Grande et al. proved that the hyperparameters associated with the kernels can be optimized online [24]. The flight test results presented in this research show GP-MRAC outperforms modern MRAC methods using NN.

Chapter 3

Rapid-Transferable Control for Fixed Wing Small Unmanned Aerial Vehicles

3.1 Aircraft Kinematics and Dynamics

Consider an aircraft with mass m and mass moment of inertia $\mathbf{I}^b$, where $(\cdot)^b$ represents the moment of inertia about the body axis. The position of the aircraft p^n is determined using an earth-fixed inertial frame of reference and denoted using the superscript $(\cdot)^i$. The origin is fixed at a desired home location with the x-axis pointing towards north, y-axis towards east and z-axis pointing downwards completing the right-hand rule.



	Vector	roll	pitch	yaw
Angular Rates	ω^b	p	q	r
Velocity	$\mathbf{V}^b$	u	v	w
Forces	$\mathbf{F}^b$	X	Y	Z
Moments	$\mathbf{M}^b$	L	M	N

Figure 3.1: Vehicle coordinate frame of reference

The body axis shown in figure 3.1 show that the x-axis of the body fixed frame points out the nose of the aircraft, the y-axis is directed out of the starboard wing of the aircraft, and the z-axis is oriented downward, completing the right-handed coordinate system. The origin is centered at the *center of gravity* of the aircraft as shown. The attitude of the vehicle is described using Euler angles defined, $[\ \phi \quad \theta \quad \psi\ ]$, where ϕ describes roll, θ is pitch, ψ is yaw about the inertial frame. The transformation between the inertial frame and body frame is given by the transformation matrix given in equation 3.1.

$$\mathbf{R_b^i} = \begin{bmatrix} C_\theta C_\psi & S_\phi S_\theta C_\psi - C_\phi S_\psi & C_\phi S_\theta C_\psi + S_\phi S_\psi \\ C_\theta S_\psi & S_\phi S_\theta S_\psi + C_\phi C_\psi & C_\phi S_\theta S_\psi + S_\phi C_\psi \\ -S_\theta & S_\phi C_\theta & C_\phi C_\theta \end{bmatrix} \tag{3.1}$$

Note that in equations 3.1 and 3.2, $S_\theta = \sin\theta, C_\phi = \cos\phi$, and so on. The relationship between the body fixed angular rates and inertial frame angular rates is

$$\begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix} = \begin{bmatrix}1 & S_\phi \tan_\theta & C_\phi \tan\theta \\ 0 & C_\phi & -S_\phi \\ 0 & S_\phi \sec\theta & C_\phi \sec\theta\end{bmatrix} \begin{bmatrix}p\\ q\\ r\end{bmatrix} \tag{3.2}$$

The equations of motion for the aircraft can be derived using Newton's Second Law of motion which states the summation of all external forces on the aircraft must be equal to time rate change of momentum and the summation of the external moments must be equal to time rate of change of angular momentum. Consequently these laws can be expressed in inertial frame as given in equations 3.3 and 3.4

$$\sum_i \mathbf{F}_i = m\ddot{\mathbf{p}}^n = \mathbf{g}^n + \mathbf{R}_b^n m \mathbf{a}^b \tag{3.3}$$

$$\dot{\omega}^b = \left(\mathbf{I}^b\right)^{-1}\left(\mathbf{M}^b - \omega^b \times \mathbf{I}^b\omega^b\right); \tag{3.4}$$

where, $\mathbf{g}^n = [\begin{array}{ccc}0 & 0 & g\end{array}]^T$ is the acceleration due to gravity vector in the inertial frame, and $\mathbf{a}^b = [\begin{array}{ccc}\dot{u} & \dot{v} & \dot{w}\end{array}]^T$ is the body fixed accelerations. The dynamics of the aircraft can be described in the body frame of reference using the transformations defined above in the equations 3.1 and rearranging the terms we get

$$\begin{bmatrix}\dot{p_n}\\ \dot{p_e}\\ \dot{p_d}\end{bmatrix} = \mathbf{R_b^i}\begin{bmatrix}u\\ v\\ w\end{bmatrix} \tag{3.5}$$

$$\begin{bmatrix}\dot{u}\\ \dot{v}\\ \dot{w}\end{bmatrix} = \begin{pmatrix}-g\sin\theta\\ -g\sin\phi\cos\theta\\ -g\cos\phi\cos\theta\end{pmatrix} + \frac{1}{m}\left[\begin{pmatrix}F_T\\ 0\\ 0\end{pmatrix} + \begin{pmatrix}X\\ Y\\ Z\end{pmatrix}\right] - \begin{bmatrix}qw - rv\\ ru - pw\\ pv - qu\end{bmatrix} \tag{3.6}$$

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \left(\mathbf{I^b}\right)^{-1} \left(\begin{bmatrix} L \\ M \\ N \end{bmatrix}^b - \begin{bmatrix} p \\ q \\ r \end{bmatrix}^b \times \mathbf{I^b} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \right) \quad ; \tag{3.7}$$

where, F_b and M_b are given by the aerodynamic forces on the aircraft. The aerodynamic forces are primarily dependent on the angle of attack, α, and side slip, β, in steady states. However, the body fixed angular rates can significantly change the aerodynamic forces as shown in equations 3.8 and 3.9.

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} C_X(\alpha, \beta) \\ C_Y(\beta) \\ C_Z(\alpha) \end{pmatrix} QS \tag{3.8}$$

$$\begin{pmatrix} L \\ M \\ N \end{pmatrix} = \begin{pmatrix} C_L(\delta_a, \beta, \tilde{p}, \tilde{r}) QSb \\ C_M(\delta_e, \alpha, \tilde{q}) QS\bar{c} \\ C_N(\delta_r, \beta, \tilde{r}) \end{pmatrix} \quad ; \tag{3.9}$$

where, $\tilde{p} = \dfrac{bp}{2V_t}, \tilde{q} = \dfrac{\bar{c}q}{2V_t}, \tilde{r} = \dfrac{br}{2V_t}$. Since body fixed forces and moments are functions of multiple variables, they are the most complex part of the aircraft to be modeled. Usually, linear approximations are used for aerodynamics forces. In-depth explanation of the reference frames , flight dynamics and control are referred to [25–28]

3.1.1 Kinematic Guidance Models

The Guidance model assumes that the autopilot controls the airspeed (V_a), altitude (h) and the course angle(χ). The corresponding equations of motion are given by

$$\begin{aligned}
\dot{p}_n &= V_a \cos\psi + w_n \\
\dot{p}_e &= V_a \sin\psi + w_e \\
\ddot{\chi} &= b_{\dot{\chi}}(\dot{\chi}_c - \dot{\chi}) + b_{\chi}(\chi_c - \chi) \\
\ddot{h} &= b_{\dot{h}}(\dot{h}_c - \dot{h}) + b_h(h_c - h)
\end{aligned}$$

$$\dot{V}_a = b_{V_a}(V_a^c - V_a) \tag{3.10}$$

where the inputs are the commanded altitude h_c, the commanded airspeed V_a^c and the commanded course χ_c and ψ.

3.2 Autopilot Design

In the autopilot design, the foremost task is to control the inertial position (p_n, p_e, h) and the attitude (ϕ, θ, ψ) of the aircraft. In the design of the autopilot, we use the technique called successive loop closure which assumes that the lateral and the longitudinal dynamics of the aircraft are decoupled. This assumption simplifies the development of the control scheme. In successive loop closure, the principle is to close several feedback loops in succession around the open loop plant dynamics. The control value calculated from the outer loop, based on the feedback signal is used as an input to the inner loop and the output of the inner loop controller is used as the control actuation for the plant. Guidance Models 3.1.1 are used to calculate commanded signal for the outer loop.

The lateral autopilot design is shown in the figure 3.2. In successive loop closure of the lateral autopilot design the inner loop controls the roll angle (ϕ), while the outer loop controls the course heading (χ) of the aircraft. As shown in the figure 3.2, the lateral guidance mechanism generates the desired course angle (χ_c), which is passed to outer loop course controller. The outer loop course control mechanism uses a generic purpose PI control to generate the commanded roll angle (ϕ_c), such that the course (χ) asymptotically tracks the commanded course angle. The output of the course hold is

$$\phi_c = k_{p_\chi}(\chi_c - \chi) + \frac{K_{i_\chi}}{s}(\chi_c - \chi) \tag{3.11}$$

The inner loop lateral autopilot controls the roll dynamics of the aircraft. The inner loop controller uses the feedback information about the roll and uses the desired roll angle generated by the outer loop to calculate the control surface deflection. However as described in the equations of motion in the previous section, the roll dynamics are highly non-linear and a generic PID controller is incapable of adapting to uncertainties and requires certain degree of tuning to adapt it to different platforms.

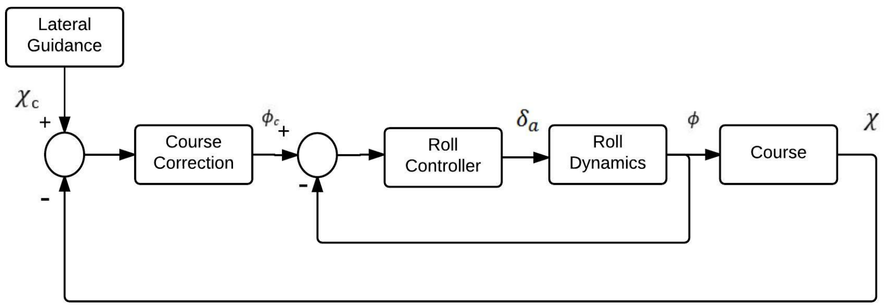


Figure 3.2: Lateral Motion Control Using Successive Loop Closure

The longitudinal autopilot design is shown in the figure 3.3. Similarly, in the longitudinal autopilot design, the inner loop controls the pitch angle(θ), while the outer loop handles the altitude(h) of the aircraft. As shown in the figure 3.3, the longitudinal guidance mechanism generates the desired altitude (h_c), which is passes to outer loop altitude controller.The outer loop altitude controller uses a generic PI control to generate the commanded pitch angle(θ_c), such that aircrafts maintains the commanded altitude. The output of the altitude hold is

$$\theta_c = k_{p_h}(h_c - h) + \frac{K_{i_h}}{s}(h_c - h) \tag{3.12}$$

Similar to the lateral control, in longitudinal control the inner loop controls the pitch dynamics of the aircraft. However, as described in the equations of motion, the pitch dynamics are highly non-linear and is similar to the case of the roll, incapable of adapting to uncertainties and requires certain degree of tuning to adapt to different platforms.

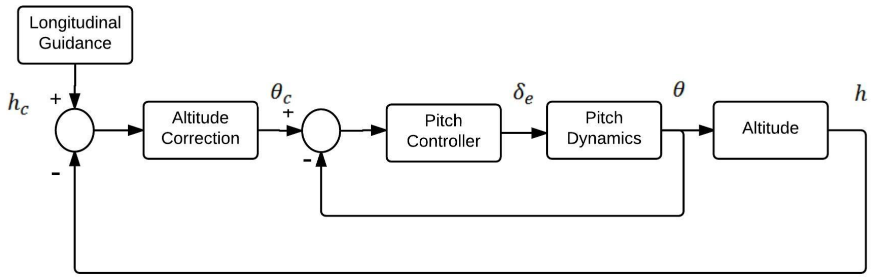


Figure 3.3: Longitudinal Motion Control Using Successive Loop Closure

3.3 Model Reference Adaptive Control

Approximate Model Inversion based Model Reference Adaptive Control is an MRAC method that allows the design of adaptive controllers for a general class of nonlinear plants where an inversion model exists. Let $x(t) \in R^n$ be the state vector, let $\delta(t) \in R^m$ denote the control input and consider the following nonlinear uncertain dynamical system

$$\begin{aligned} \dot{x_1}(t) &= x_2(t), \\ \dot{x_2}(t) &= f(x(t), \delta(t)) \end{aligned} \tag{3.13}$$

The non linear system can also be represented as

$$\dot{x}(t) = Ax(t) + B(u + \Delta)(t) \tag{3.14}$$

where Δ is a smooth non-linear function.

In AMI-MRAC a pseudo control input is designed $\nu(t) \in R^m$ that can be used to find the control input δ such that the system states track the output of the reference model. Since, the exact system model is usually not known, ν is considered to be the output of an approximate inversion model $\hat{f}$ where

$$\delta = \hat{f}^{-1}(x, \nu) \tag{3.15}$$

The use of an approximate inversion model results in a model error of the form

$$\dot{x_2} = \nu + \Delta(x, \nu) \tag{3.16}$$

where Δ is the modeling error given by

$$\Delta = f - \hat{f} \tag{3.17}$$

A reference model can be designed that characterizes the desired response of the system

$$\begin{aligned} \dot{x}_{1rm} &= x_{2rm}, \\ \dot{x}_{2rm} &= f_{rm}(x_{rm}, r) \end{aligned} \tag{3.18}$$

where $f_{rm}(x_{rm}(t), r(t))$ denotes the reference model dynamics. The command $r(t)$ is assumed to be bounded and piecewise continuous.

The pseudo-control input ν conssits of a linear feedback, a linear feedforward and an adaptive part which is in the following form

$$\nu(t) = \nu_{rm}(t) + \nu_{pd}(t) - \nu_{ad}(t) \tag{3.19}$$

Defining the tracking error $e(t) = x_{rm}(t) - x(t)$, the tracking error dynamics can be written as

$$\dot{e} = Ae + B(\Delta - u_{ad}) \tag{3.20}$$

The baseline full state feedback controller ν_{pd} is selected such that A is Hurwitz. Hence for any positive definite matrix $Q \in R^{mxm}$, a positive definite solution $P \in R^{mxm}$ exits to the Lyapunov equation.

$$A^T P + PA + Q = 0 \tag{3.21}$$

Consider Γ_w to denote positive definite learning rate and considering gradient based adaptation law [29] $\dot{W}(t) = -\Gamma_w \Phi(t) e^T(t) PB$ that minimizes a cost on the instantaneous tracking $e^T e$ guarantees that the tracking error is uniformly bounded for the adaptive controller framework described above. However, this adaptive law guarantees that the parameters (W) stay bounded within a neighborhood of the ideal parameters (W^*) only if $\Phi(t)$ is persistently excited [30]. Narendra and Annaswamy introduced the e-modification [31]. The adaptive law with the e-modification follows the update law

$$\dot{W}(t) = -\Gamma_w \Phi(t) e^T(t) PB - \sigma \mid e(t) \mid W \tag{3.22}$$

The rational for using a error-dependent damping is that it tends to zero, as the regulated output error diminishes. Also e-modification helps in keeping the weights bounded through out the entire operating domain [31].

3.3.1 Gaussian Process Model Reference Adaptive Control

Gaussian Process Model Reference Adaptive Control is widely studied upon in [32], [21]. It was implemented successfully on quadrotors and the results yielded were very impressive [24]. However, there is very less study on implementation of GP-MRAC in Fixed Wing Aircrafts [34]. A detailed overview of Gaussian Processes can be found in the section 3.3.1

To achieve the tracking objective, the adaptive element attempts to learn the mean of the stochastic process online.

From the equation 3.14

$$\Delta = B^{-1}(\dot{x} - Ax) - u \tag{3.23}$$

Traditionally, in GP-MRAC [32], [21], [24], in order to estimate the $\dot{x}$, the exact value of control effectiveness matrix B is to be known. This poses to be a problem as $\dot{x}$ can be noisy and B changes with the acceleration. To overcome this issue, an alternates solution is presented for the implementation of GP-MRAC, that is to use the pointwise estimation of $Delta$ from a traditional high-gain MRAC as

$$\dot{W}(t) = -\Gamma_w \Phi(t) e^T(t) PB - \sigma \mid e(t) \mid W \tag{3.24}$$

Note that even if the learning rate is increasing, there is no effect on the controller as the weights from the baseline adaptive controller are not utilized. Instead we use the $\hat{Delta}$ that is trained from the Gaussian Process.

$$\hat{\Delta} = W^T \Phi(x) \tag{3.25}$$

The mean of the estimate of the uncertainty trained on the GP is assigned to the adaptive element ν_{ad} which is used in the calculating the pseudo control input ν.

The benefits of this implementation are that the estimation of the control effectiveness matrix is not necessary as the system uncertainty is being captured. This implementation of the GP-MRAC is done and tested in the Hardware-in-the-loop environment.

Gaussian Processes

A Gaussian Process is a supervised learning technique, Typically, Gaussian Process Regression (GPR) is used to learn input-output mapping function f from the training data set $\mathcal{D}$ of n observations, $\mathcal{D} = \{(x_i, y_i) | i = 1, \ldots, n\}$, where x denotes the input vector of dimension D, and y is the scalar output (or target); the column vector inputs for all n cases are aggregated in the $D \times n$, matrix X. Once the mapping function f is known for the set of inputs X, it can then be used to make predictions for all possible set of test values X_* through the derivation of the posterior function $f(X_*)$.

By definition, a Gaussian process describes distribution over functions and is completely specified by its mean function $m(x)$ and covariance function $k(x, x')$ of a real process $f(x)$ as

$$\begin{aligned} m(x) &= \mathbb{E}[f(x)], \\ k(x, x') &= \mathbb{E}[(f(x) - m(x))(f(x) - m(x'))] \end{aligned}$$

which can be denoted as

$$f(x) \sim \mathcal{GP}(m(x), k(x, x')). \tag{3.26}$$

In present work, we use squared exponential covariance function defined as,

$$cov(f(x), f(x')) = k(x, x') = exp(-\frac{\|x - x'\|^2}{2\sigma}) \tag{3.27}$$

To derive $f(X_*)$ using GPR given the dataset $\mathcal{D}$, we begin by defining a zero mean prior over the functions as

$$f \sim \mathcal{N}(0, K(X, X)) \tag{3.28}$$

where $K(X, X)$ is a covariance matrix, with entries $k(x_i, x_j)$ for $i, j = 1, \ldots, n$. Next, we incorporate measurement noise in the output as $y = f(x) + \epsilon$, assuming additive independent identically distributed Gaussian noise ϵ with variance σ_n^2 , hence the prior on the noisy observations now becomes $f \sim \mathcal{N}(0, K(X, X) + \sigma_n^2 I)$. The joint distribution of the measured target values and the function values at the test locations according to the prior is

$$\begin{bmatrix} y \\ f_* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K(X, X) + \sigma_n^2 I & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{bmatrix}\right) \tag{3.29}$$

where $f_* = f(X_*)$. The posterior conditioned on the observations gives the key predictive equations for Gaussian process regression as

$$f_*|X, y, X_* \sim \mathcal{N}(\bar{f}_*, cov(\bar{f}_*)) \tag{3.30}$$

$$\bar{f}_* = K(X_*, X)[K(X, X) + \sigma_n^2 I]^{-1} y \tag{3.31}$$

$$cov(\bar{f}_*) = K(X_*, X_*) - K(X_*, X)[K(X, X) + \sigma_n^2 I]^{-1} K(X, X_*) \tag{3.32}$$

where $\bar{f}_*$ is the mean prediction at locations X_* and $cov(\bar{f}_*)$ is the predictive uncertainty. Hence, the mean is directly estimated from the set of available data.

The main strength of the GPR is that it does not need to assume an a-priori allocation of the RBF centres. However, the main disadvantage of using the traditional GPR techniques is the covariance matrix increases in size as the size of $\mathcal{D}$ increases. In online applications, this can quickly become intractable as computing the inverse of the covariance matrix can become computationally intractable. It was shown that this problem can be alleviated in [21] using online sparsification techniques,budgeted online Sparse Gaussian Process regression technique [23]. This technique only includes valuable data points in an active Basis Vector set $\mathcal{BV}$. When new data is observed, the sparsification algorithm computes how well the new data point can be approximated by the existing basis vectors using a comparative test called the *kernel independence test* defined by

$$\gamma = \| \sum_{i=1}^{t} \alpha_i \psi(x_i) - \psi(x_{t+1}) \|_{\mathcal{H}}^2 \tag{3.33}$$

The γ_{t+1} gives the residual distance between $\psi(x_i)$ and the GP generated by elements in $\mathcal{BV}$. An existing element ψ_m in the basis vector set which minimizes $D(\mathcal{GP} \parallel \mathcal{BV}) - D(\mathcal{GP} \parallel \mathcal{BV} \backslash \{\psi_m\})$ is removed and the new sample is added to the set. Given the basis vector set, the approximate mean and variance can be written as:

$$\bar{f}_* = K(X_*, \mathcal{BV})[K(\mathcal{BV}, \mathcal{BV}) + \sigma_n^2 I]^{-1} y \tag{3.34}$$

$$cov(\bar{f}_*) = K(X_*, X_*) - K(X_*, \mathcal{BV})[K(\mathcal{BV}, \mathcal{BV}) + \sigma_n^2 I]^{-1} K(\mathcal{BV}, X_*) \tag{3.35}$$

Ref. [33] provides a complete analysis of the properties of GPs.

Chapter 4

AutoPilot Design & Development

4.1 Hardware Design

The design and development of the autopilot takes a new approach other than the conventional autopilot design by modularizing the subsystems in the autopilot. Using this process, the system can be prevented from becoming obsolete with the advancements in the technology. Being modular also helps in developing the autopilot to be mission specific. Furthermore, any faulty subsystems can be easily replaced individually without affecting the whole system and reducing the effort of rebuilding the system from scratch.

The components that were selected to feature modularity are: the flight control computer, the inertial navigation system and the wireless ground control communications module. When selecting the components for aerospace design, the form, the weight and the power consumption of all the components play a major role.

4.1.1 Modular Components

Flight Control Computer

The flight control computer handles all the operations such as interacting with all the components on-board the aircraft, as well as communicating with the ground control

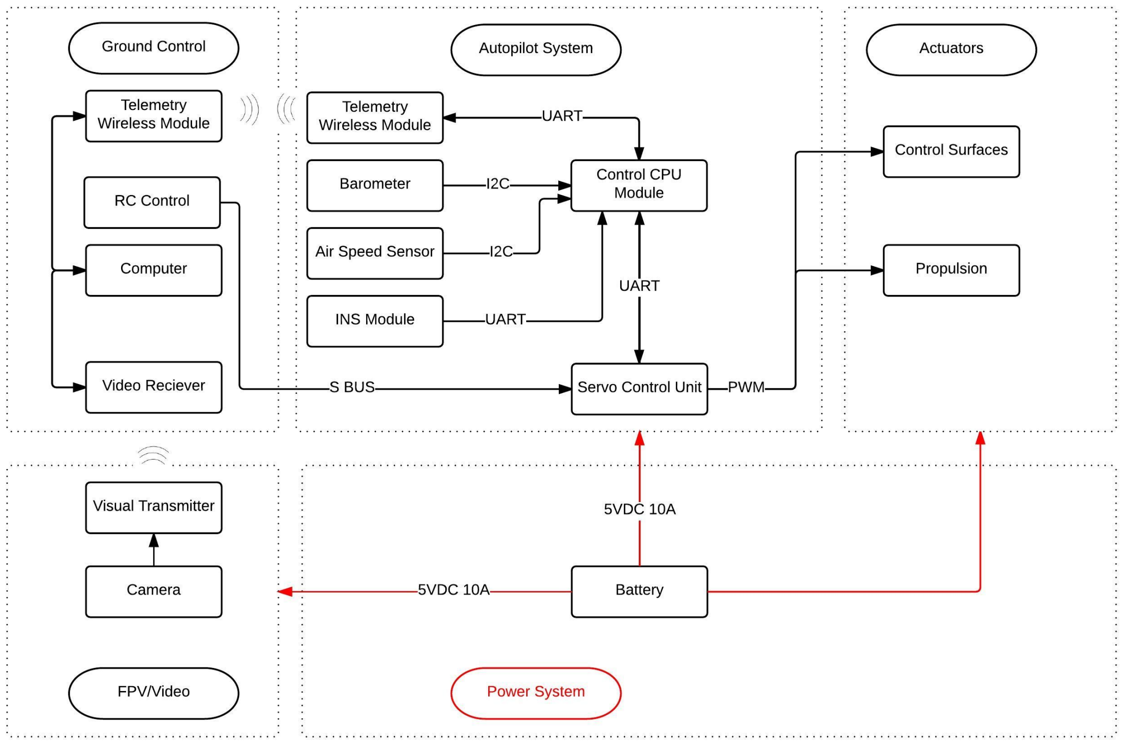


Figure 4.1: The block diagram showing the different components and their communication protocols.

station. Its primary functions include

- Analyzing the data received from the onboard sensors.
- Executing the flight controls
- Communicating with the Ground Control Station
- Logging flight data for post-flight analysis

A market survey was conducted in order to identify the most suitable computer, as special attention was needed with considering the size, weight, power consumption and input/output(I/O) ports configurations.The details of the market survey can be found in Table C.4 from Appendix C. The final choice was the BeagleBone Black, an embedded computer board as shown in Figure 4.2.

The BeagleBone Black features

- Sitara AM3358 1Ghz ARM ®- A8 32-Bit Processor
- 512 MB DDR3 RAM
- 4GB 8-bit eMMC on-board flash storage
- 2x PRU 32-bit microcontrollers

The complete specifications of BeagleBone Black can be found in Appendix C.

Figure 4.2: BeagleBone Black

Usually, the autopilots are designed and developed around the selection of the central computer. But in our approach to the design of the autopilot, the flight control computer is also modular since the selection of the subcomponents can be easily adapted to fit other similar linux-based embedded computers by simply modifying the routing and connections of the Systems Integration Board. However, the makers of BeagleBone Black, have not changed the form factor for 4 generations of development. Therefore, it is safe to assume, BeagleBone Black can be easily be replaced with an upgraded version from BeagleBone in the future with no modifications.

Navigation Sensors

Navigation sensors provide reliable measurement for the flight status of the flying vehicle. Many commercial navigation sensors are available on the market.Some of them are listed in Table C.5. All of them vary in the material, manufacturing technology, measuring range, size, weight, estimation algorithm, positional accuracies. Based on the working principle, a navigation solution falls into one of the categories.

- INS (Inertial Navigation System)
- INS/GPS (INS calibrated by GPS)
- GPS-aided AHRS (Altitude Heading Reference System)

It is a common practice to integrate the INS in the autopilot to reduce the wiring footprint and maintain the same overall form factor of the autopilot. With the advancements of Microelectromechanical Systems(MEMS), INS are increasing in precision and accuracy very rapidly. But this being chosen as a modular unit, the INS was not integrated in the Systems Integration Board, as it allows the user to select one that matches the required form factor, the budget allowance and can be easily swapped, if necessary. Also, most of the COTS navigation sensors come in rugged, self-contained packages which gives freedom to the user to place the unit where it is inconvenient to place the flight control computer. VectorNav's VN-200 Rugged GPS/INS shown in Figure 4.3, has been selected as it a miniature high performance INS that features MEMS inertial sensors, a high-sensitivity GPS receiver, advanced Kalman filtering algorithms to provide optimal estimates of position, velocity and orientation. The complete specifications of VectorNav's VN-200 Rugged GPS/INS can be found in Appendix C.

Figure 4.3: VectorNav's VN-200 Rugged GPS/INS

Wireless Communication Device

Communication range and reliability are most important factors when the wireless communication device is selected. The Ground Control Station is the relay for all of the relevant information on-board the UAV. Similar to the navigation sensors, wireless communication technology is advancing rapidly and is becoming much more efficient. This component is placed off board the autopilot, this way it reduces the Electromagnetic Interference (EMI) caused by the other systems. Three different low-cost, serial wireless communication modules were tested to determine the connection strength and its robustness. The modules that were tested were

- XBee - 900 working at 915 MHz
- 3DR Telemetry Radios working at 915 MHz
- jDrones jD-RF900Plus Longrange working at 915 MHz

The jDrones jD-RF900Plus shown in the figure 4.4 has been selected as the connection strength and the performance was better.

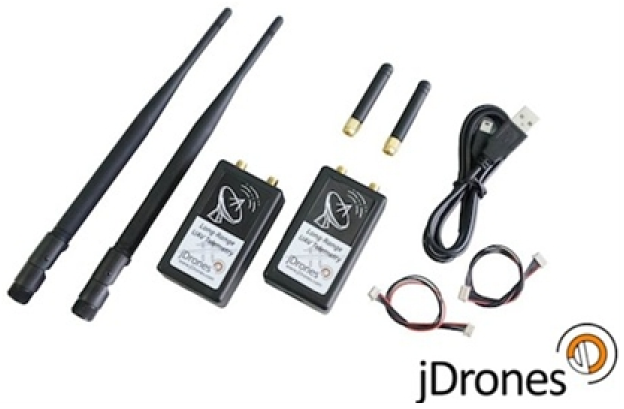
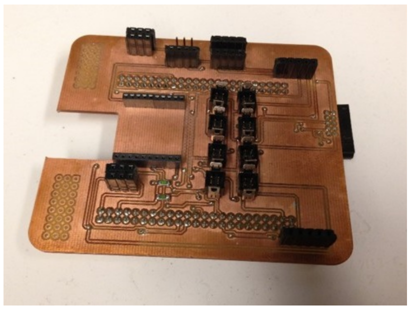


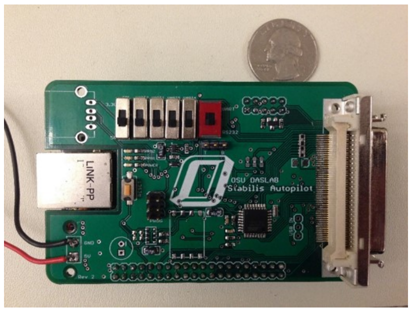
Figure 4.4: jDrones jD-RF900Plus Longrange telemetry set

Systems Integration Board

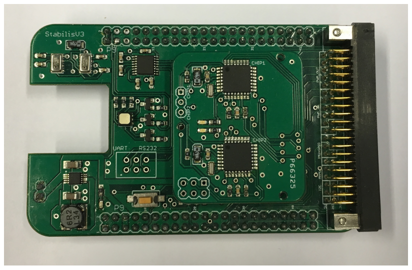
The main purpose of the Systems Integration Board, SIB in short, is the integration of the Flight Control Computer with the other sensors and components on board the aircraft. The SIB was designed with the form factor and robustness in mind. The design of SIB has improved over the iterations as shown below in Figure 4.5 . In order to eliminate the various issues such as loose or faulty connections, as well as to easily use the autopilot like a plug and play device for quick connect/disconnect the iterations were developed.



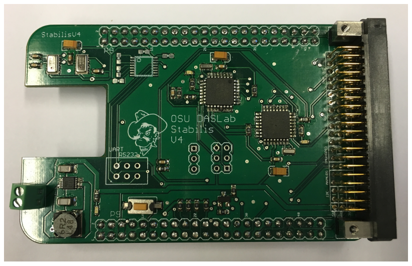
(a) Prototype SIB



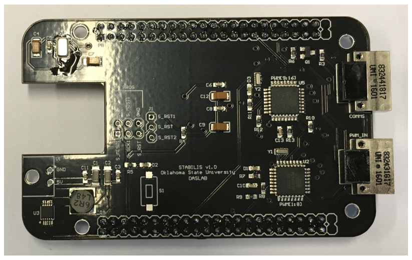
(b) First iteration for SIB



(c) Second iteration for SIB



(d) Third iteration of SIB



(e) Latest iteration of SIB

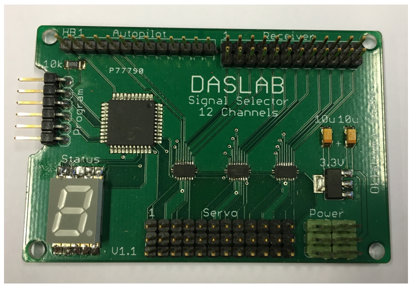
Figure 4.5: Different Iterations of the Systems Integration Board(SIB)

Peripheral Sensors

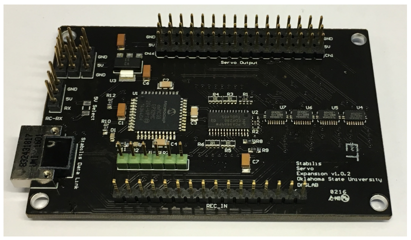
There were two sensors that have been chosen to go on the Systems Integration Board (SIB). The Honeywell, HSCMRRN001PD2A3, was chosen for its superior resolution, accuracy and form factor to provide the differential pressure reading from the Airspeed sensor. Additionally, we have the Freescale MPL3115A2 Absolute Digital Pressure Sensor on the SIB to provide accurate pressure [Pascals]/altitude [meters] and temperature [°C].

Fail-Safe Servo Driver

The Fail-Safe Servo Driver or Servo Driver in short, is another important part of the Autopilot Design to guarantee the airborne safety of the small UAV. It is mainly responsible for decoding both piloted and computer generated servo control commands and selecting desired decode signals to drive multiple servo actuators. In case of any malfunction of the any component or accidents during autonomous flight, with the Servo Driver, the human pilot has a chance to retrieve the UAV to safety. As the SIB improved over iterations, even there were improvised iterations of the Servo Driver as shown in the Figure 4.6.



(a) First Iteration Servo Driver supporting Second and Third Iteration SIB



(b) Second Iteration Servo Driver supporting Fourth Iteration SIB

Figure 4.6: Different Iterations of the Fail-Safe Servo Driver

4.2 Software Design

4.2.1 Multi-Threaded Design

The software system for the autopilot is developed based on multi-threaded architecture to ensure integrity and robustness of the system. The thread structure is employed to execute multiple tasks based on the functionality and hardware components. The threads are shown in the Figure 4.7. This design aligns well with the practices observed in past works [28]. To execute the threads, the *main()* function is tasked with the initialized with several parameters such as the system gains, actuator limits and sensor profiles. In a multi-threaded system, the tasks for each thread must be scheduled such that the control is executed properly. A detailed explanation of the software design can be found in [34].

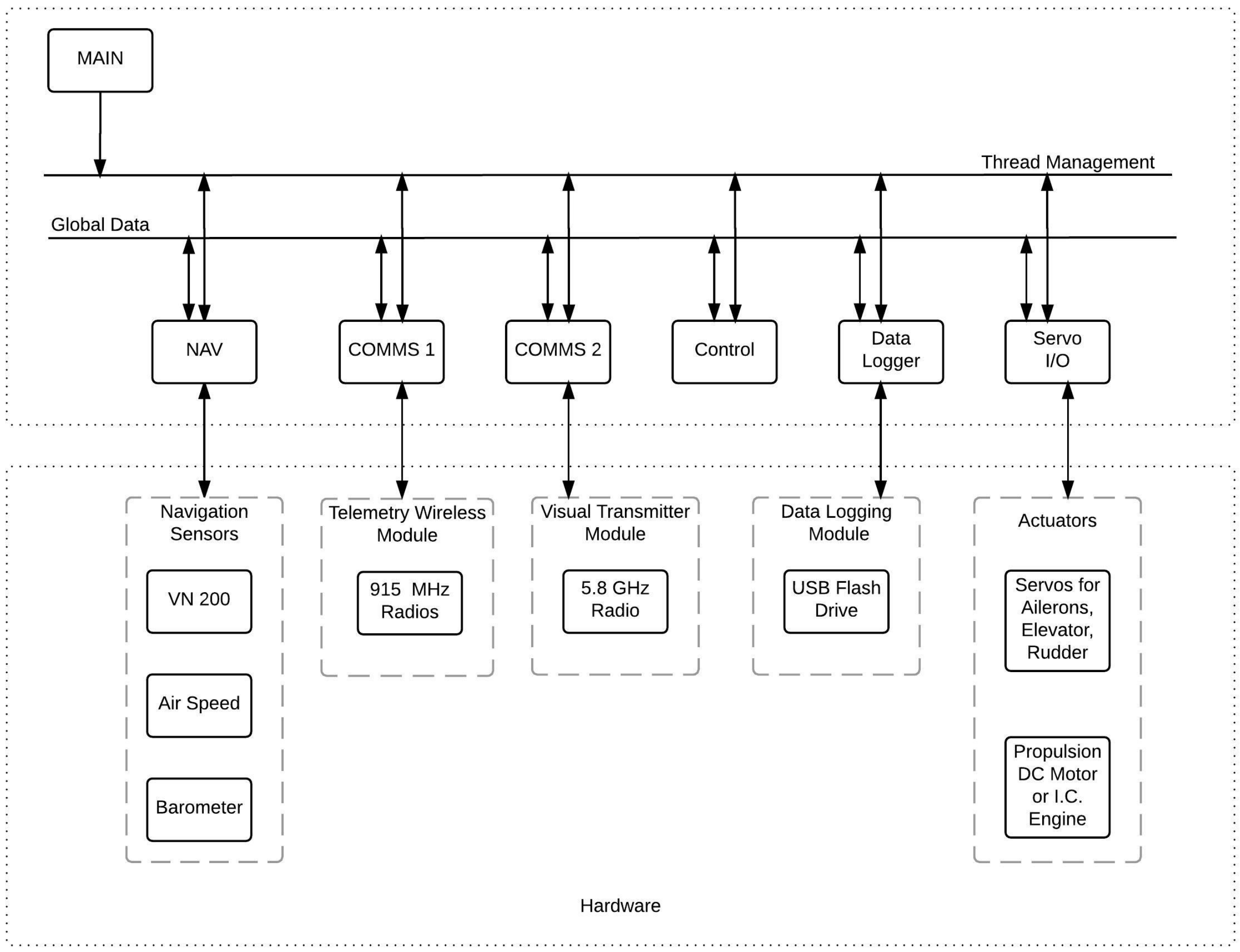


Figure 4.7: Thread design block Design

4.2.2 Ground Control Station Software

The Ground Control Station plays the primary role as the means by which operators plan, execute and monitor UAS missions through a wireless communication channel.

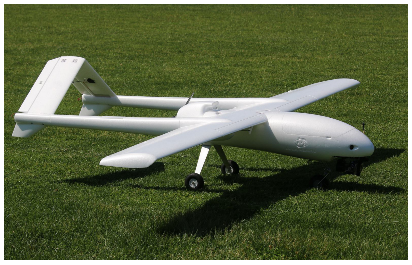
The task of the ground station is to provide a realistic interface for users to monitor the performance of the UAV during the flight tests. Many ground control software platforms exist but QGROUNDCONTROL(QGC) is a well documented, platform independent and community supported ground station software package. QGC software is compatible with the major Operating Systems (Windows, Linux, Mac OS X). It also features serial, UDP, TCP and mesh networks communication compatibility. It also has real-time plotting and logging capabilities of onboard parameters. It also features the ability to change onboard parameters relevant for the Control law. QGC utilizes a highly efficient communication protocol called MAVLINK. MAVLINK is an extensively tested and possibly the most widely used communication protocol in the UAS research community.

4.3 Airframes

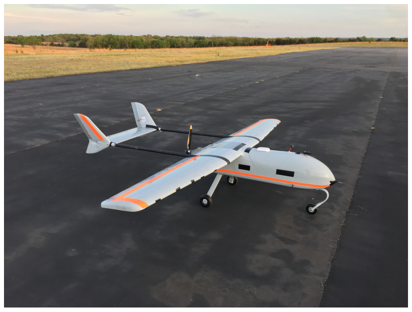
The various aircrafts in the fixed wing class, which were used to test out the autopilot in the HITL environment are

- Skyhunter
- Anaconda
- Mugin
- Penguin-B

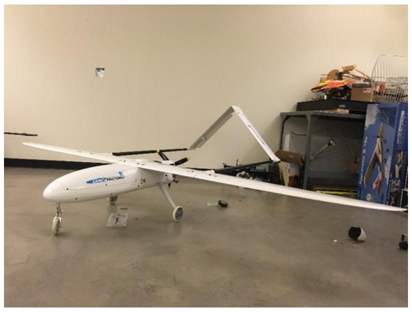
(a) Skyhunter



(b) Anaconda



(c) Mugin



(d) Penguin B

Figure 4.8: Different Aircrafts used for Hardware in the Loop Testing

The aircrafts' specifications are as follows:

Table 4.1: Aircraft Specifications

Vehicle	Skyhunter	Anaconda	Mugin	Penguin-B
Wing Span	1.8 m	2.06 m	4.45 m	3.3 m
Body Length	1.4 m	1.41 m	3.67 m	2.27 m
Wing Area	0.362 sq.m	0.49 sq.m	1.1sq.m	0.79sq.m
Engine	Electric	Electric	Gasoline	Gasoline
Weight	9 lbs	12 lbs	44 lbs	28 lbs

4.4 Hardware in the Loop

The flight tests are conducted after intensive simulations executed on the Hardware-in-the-loop (HITL) simulation system.Real-time HITL simulation is an effective method for the verification of the overall performance and safety of the unmanned systems before conducting the flight tests. In the HITL simulation, the different modules, which include the onboard hardware system, automatic flight control system, ground control station and the software architecture are included the simulation.The simulation is done using Laminar Research X-Plane, a high fidelity simulator, utilized to simulate aircraft dynamics in order to evaluate the autopilot Stabilis.

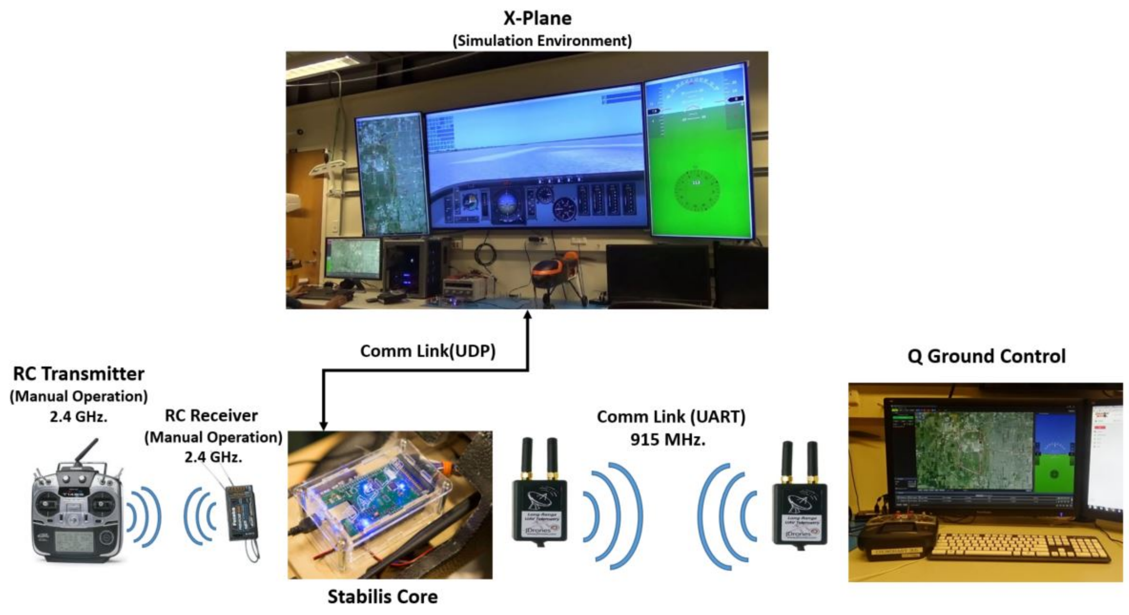


Figure 4.9: Hardware-in-the-Loop Environment Setup

The procedure to setup the Hardware-in-the-loop environment is clearly explained in Appendix D. The procedure to run the HITL testing is elaborated in Appendix E.

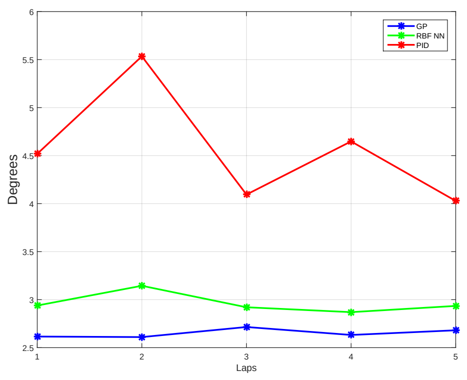
Chapter 5

Results

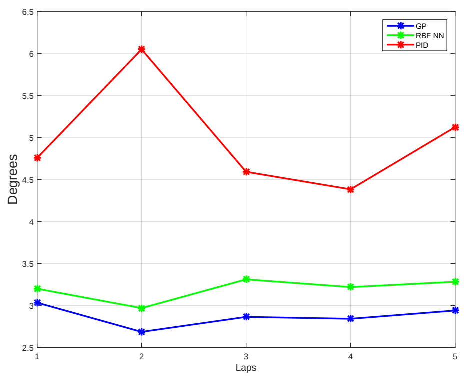
5.1 Hardware in the Loop Simulation Results

The Hardware in the loop tests presented here were executed on 4 different aircrafts: Skyhunter, Mugin, Anaconda and the Penguin-B. All the aircrafts are flown with the same mission of tracking 8 waypoints. The aircrafts are flown in similar weather conditions in the Simulator, average cross winds of 18 knots gusting upto 25 knots. The aircraft is taken off in the attitude hold mode and then put into autonomous mode where it tracks the waypoints in laps such that repeatability is ensured. The Inner Loop dynamics, roll (ϕ) and pitch (θ) are augmented with the adaptive controllers RBF-NN MRAC and GP-MRAC and are compared with the baseline PID Controller.

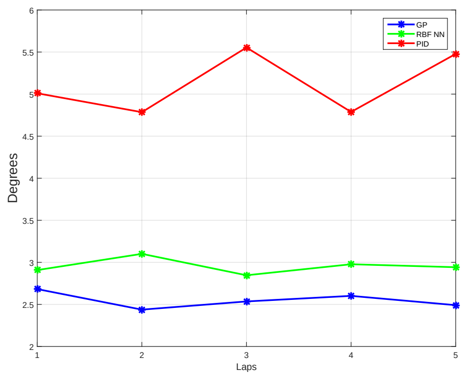
The Root Mean Square error for roll (ϕ) and pitch (θ) is calculated for the each lap with the three different controllers, implemented and tested with the four different airframes.



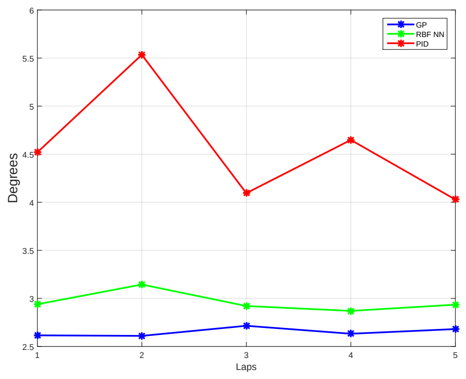
(a) Skyhunter



(b) Mugin

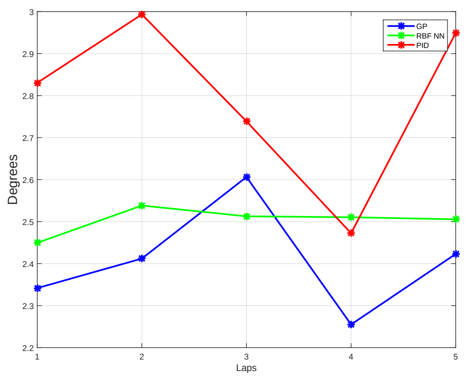


(c) Anaconda

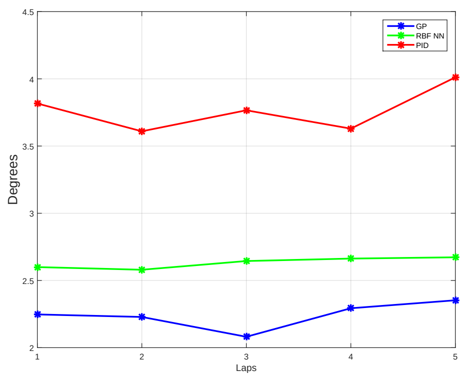


(d) Penguin-B

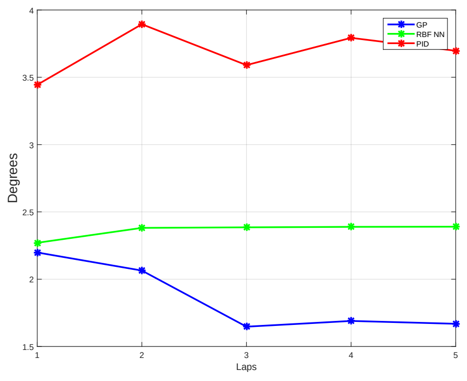
Figure 5.1: Comparison of Root Mean Squared Error in Tracking Roll with the different controllers in different Aircrafts



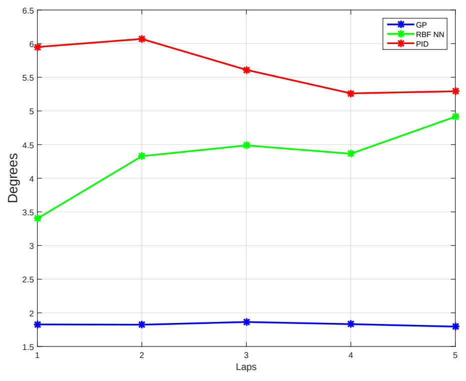
(a) Skyhunter



(b) Mugin



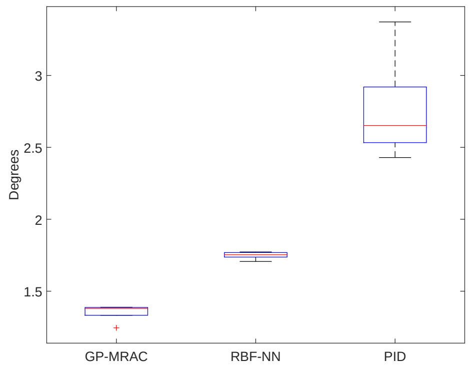
(c) Anaconda



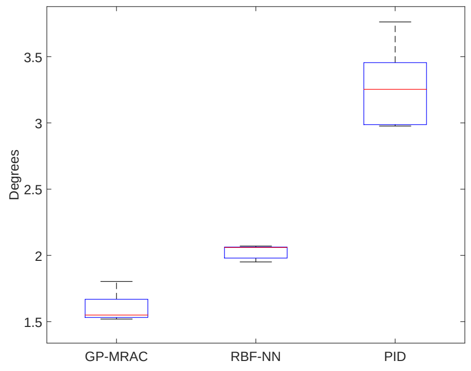
(d) Penguin-B

Figure 5.2: Comparison of Root Mean Squared Error in Tracking Pitch with the different controllers in different Aircrafts

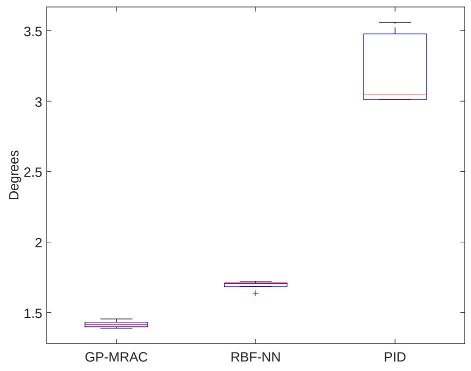
The Mean error for roll (ϕ) and pitch (θ) is calculated for each lap with the three different controllers, implemented and tested with the four different airframes.



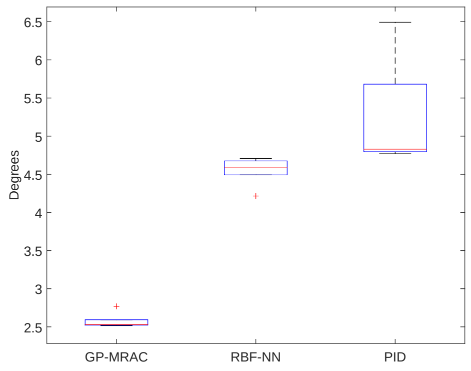
(a) Skyhunter



(b) Mugin

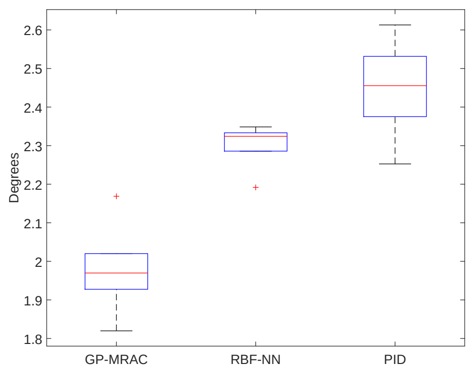


(c) Anaconda

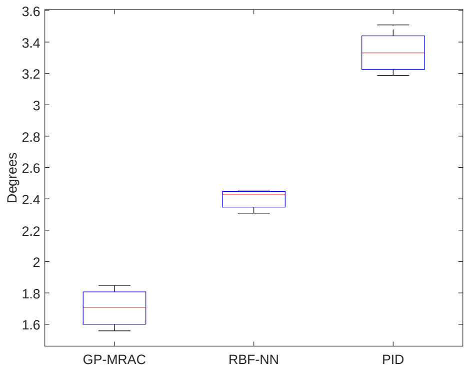


(d) Penguin-B

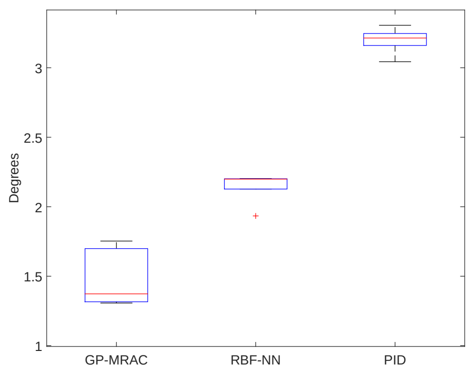
Figure 5.3: Comparison of Mean Absolute Error in Tracking Roll with the different controllers in different Aircrafts



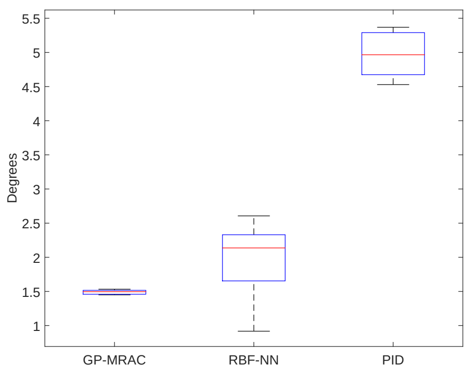
(a) Skyhunter



(b) Mugin



(c) Anaconda



(d) Penguin-B

Figure 5.4: Comparison of Mean Absolute Error in Tracking Pitch with the different controllers in different Aircrafts

From the figures 5.1,5.2,5.3,5.4 it is observed that GP-MRAC outperforms RBF-NN MRAC and the PID controller in terms of the Root Mean Squared Error and Mean Absolute Errors of the Roll and Pitch tracking.

All the plots presented describes the performance of the controllers with the aircraft Skyhunter. The results and plots of the remaining aircrafts are presented in the Appendix A .

The Waypoint Tracking performance with the three different controllers for the aircraft Skyhunter is shown. The circles represents the waypoints that the aircraft has to fly autonomously. It is observed that the aircraft Skyhunter tracks the given waypoint course well in the Autonomous mode with the three different controllers implemented in the inner loop dynamics of the aircraft. Even in the presence of high cross winds the aircraft performs well in tracking the waypoints.

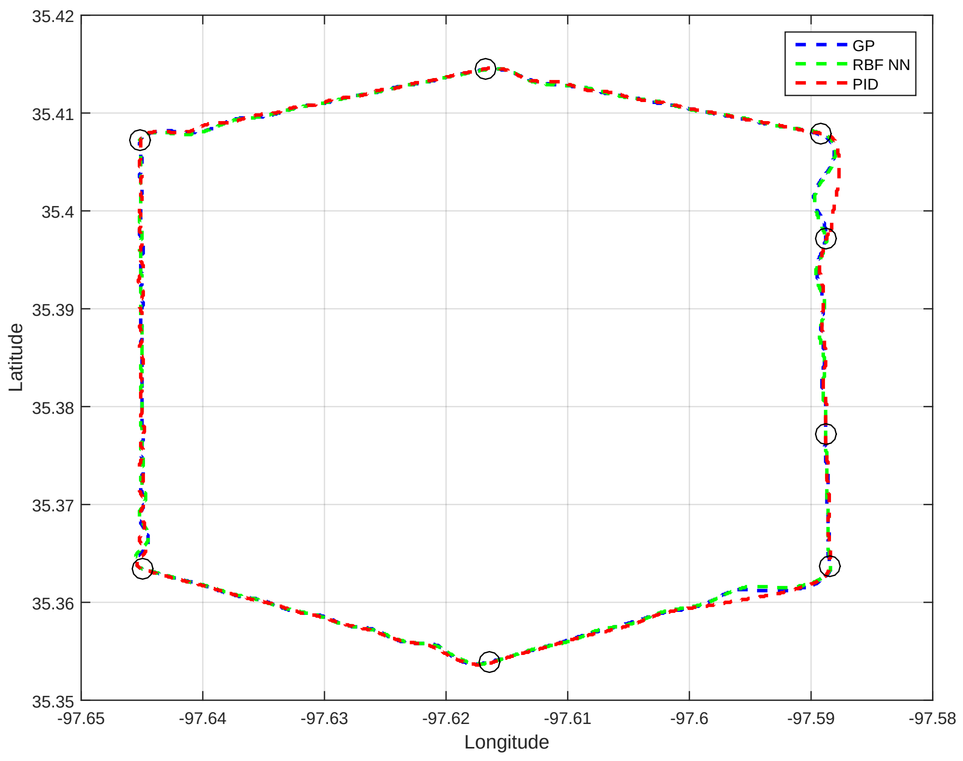
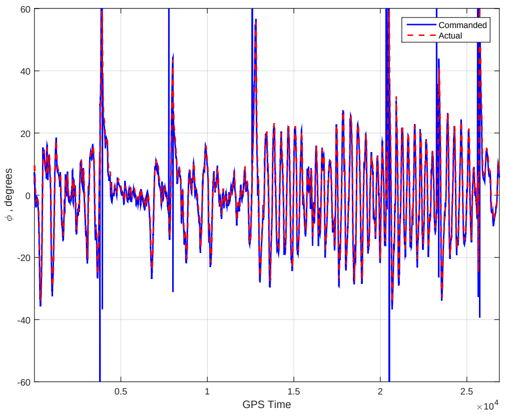
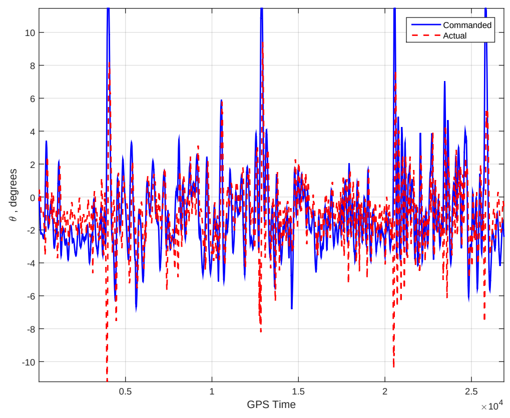


Figure 5.5: Comparison of Waypoint Tracking in Skyhunter with the different controllers

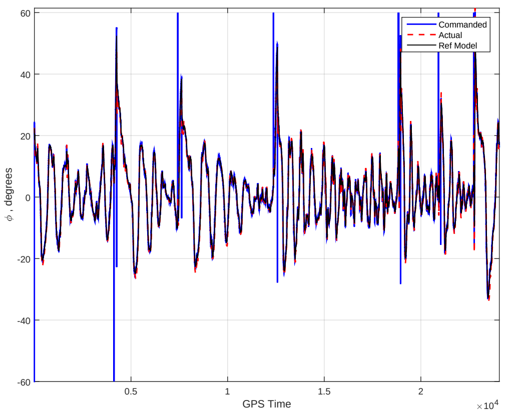
The tracking of commanded input form the outer loops for both roll (ϕ) and pitch (θ) with the different controllers for a single lap are shown.



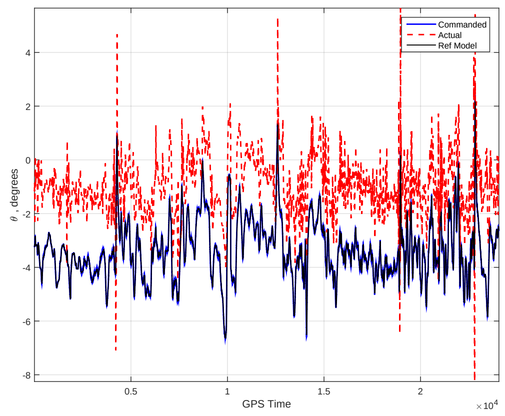
(a) Roll Control using PID



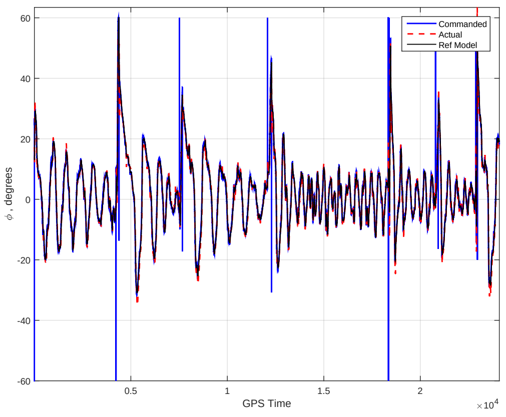
(b) Pitch Control using PID



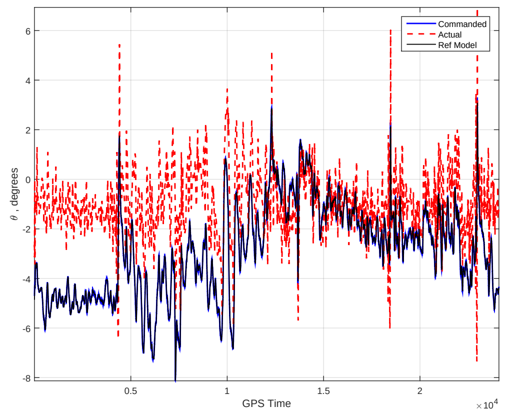
(c) Roll Control using RBF-NN MRAC



(d) Pitch Control using RBF-NN MRAC



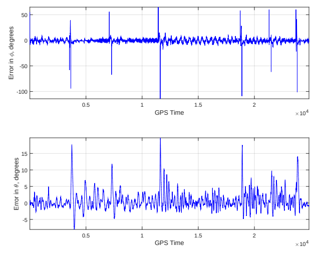
(e) for Roll Control using GP-MRAC



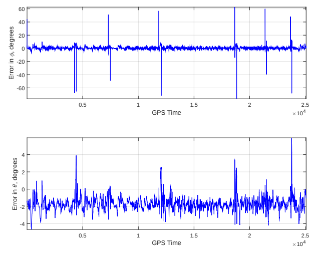
(f) Pitch Control using GP-MRAC

Figure 5.6: Tracking Performance of Roll and Pitch with the different controllers in Skyhunter

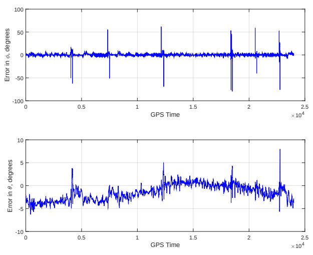
The Evolution of the Inner Loop errors and Outer loop errors are shown.



(a) Inner Loop Errors using the PID Controller

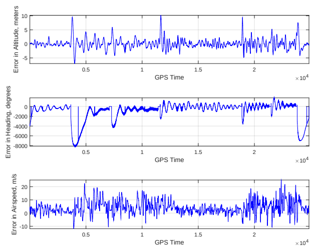


(b) Inner Loop Errors using RBF-NN MRAC

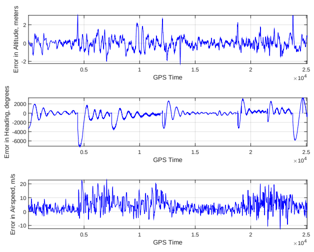


(c) Inner Loops Errors using GP-MRAC

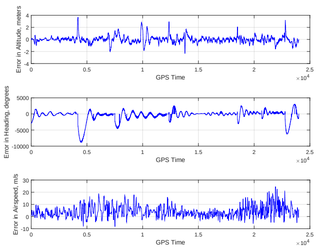
Figure 5.7: Evolution of Inner Loop Errors in Skyhunter with the various Controllers



(a) Outer Loop Errors using the PID Controller



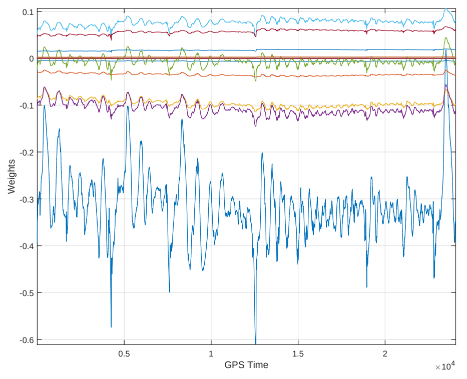
(b) Outer Loop Errors using RBF-NN MRAC



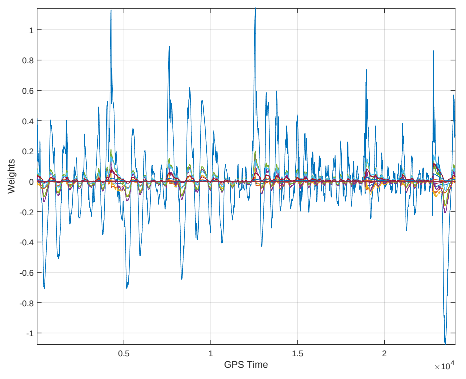
(c) Outer Loops Errors using GP-MRAC

Figure 5.8: Evolution of Outer Loop Errors in Skyhunter with the various Controllers

The Evolution of the RBF-NN weights for roll (ϕ) and pitch (θ) for a single lap while the aircraft is tracking the waypoints from the mission.



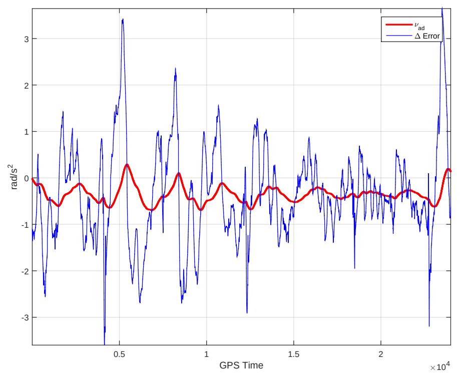
(a) Evolution of MRAC weights in Roll Dynamics



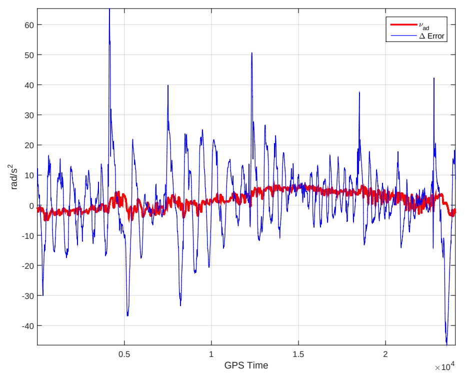
(b) Evolution of MRAC weights in Pitch Dynamics

Figure 5.9: Evolution of MRAC weights in Inner Loop Dynamics

Performance of Gaussian Processes in capturing the uncertainty for a single lap while the aircraft is tracking the waypoints from the mission.



(a) Modeling Error in Roll Dynamics in GP-MRAC



(b) Modeling Error in Pitch Dynamics in GP-MRAC

Figure 5.10: Performance of GPs in capturing the uncertainty

5.2 Flight Test Results

Real world flight tests were conducted with the Skyhunter aircraft using the baseline adaptive controller (RBF-NN MRAC). The aircraft was flown in weather conditions, average winds at 17 knots and gusting upto 23 knots.

The tracking of the commanded input from the outer loop is observed for both roll and pitch shown in figure 5.11 and 5.12.

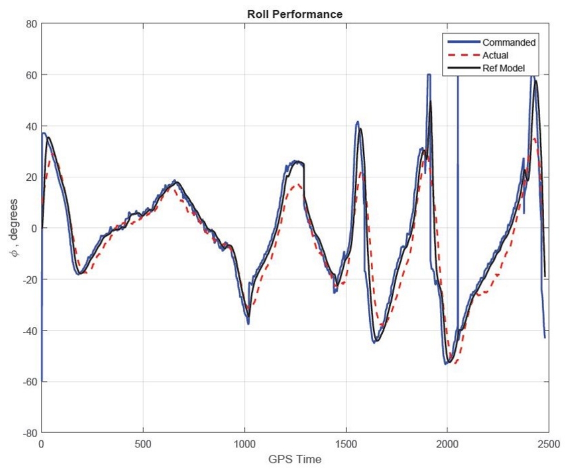


Figure 5.11: Tracking Performance for Roll Control using RBF-NN MRAC using e-mod

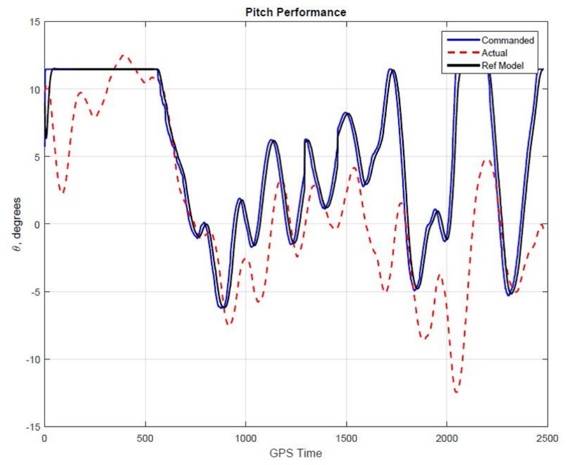


Figure 5.12: Tracking Performance for Pitch Control using RBF-NN MRAC using e-mod

The inner loop and outer loop error performance are shown in figure 5.13 and 5.14

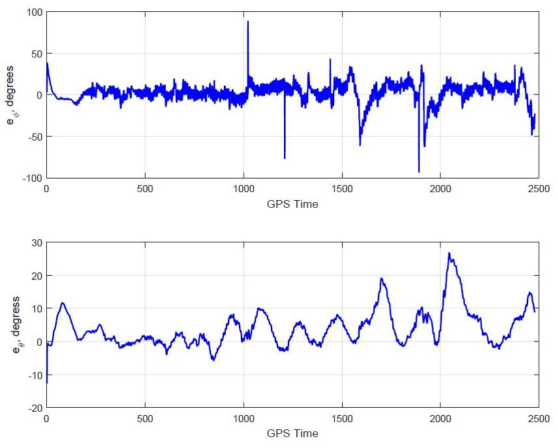


Figure 5.13: Evolution of Inner Loop Errors

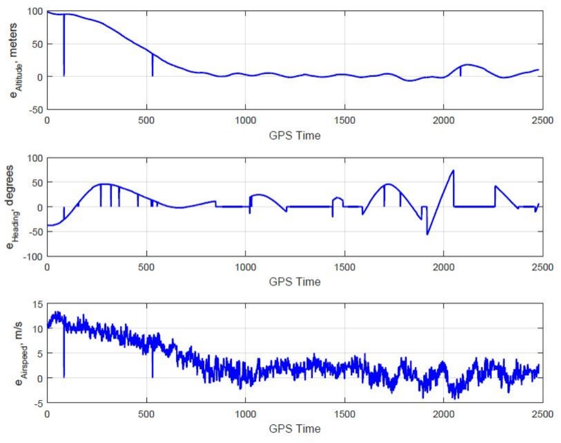
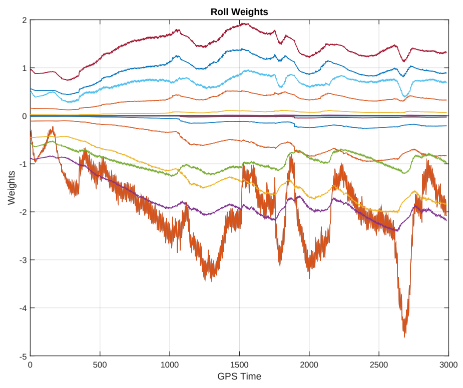
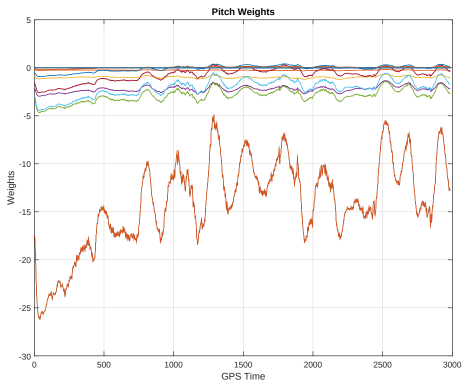


Figure 5.14: Evolution of Outer Loop Errors

The weights for roll and pitch from the MRAC remained Uniformly Bounded across the entire flight 5.15a and 5.15b

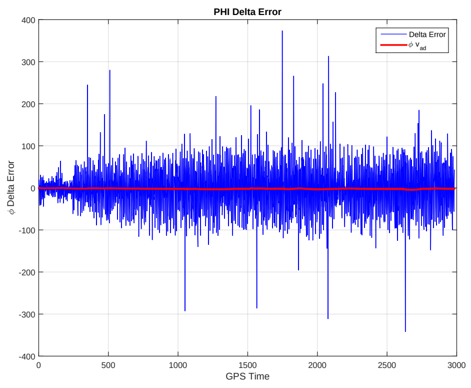


(a) Evolution of MRAC weights in Roll Dynamics

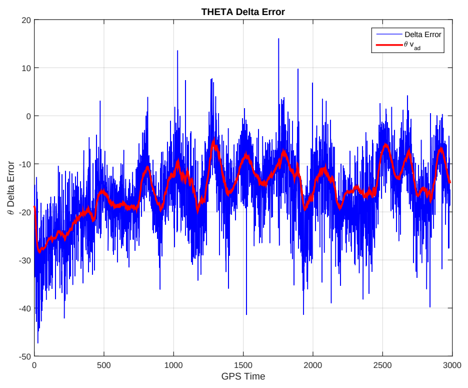


(b) Evolution of MRAC weights in Pitch Dynamics

Figure 5.15: Evolution of MRAC weights in Inner Loop Dynamics



(a) Online Disturbance Approximation of Roll Dynamics



(b) Online Disturbance Approximation of Pitch Dynamics

Figure 5.16: Adaptive Controller Performance in capturing the uncertainty

From the Figure 5.16, it is observed that the uncertainty of the modeling error is being captured very well.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

The main contribution of this thesis was the extensive Hardware in the Loop testing results comparing the tracking performance of GP-MRAC to the RBF-NN MRAC as well as the baseline Proportional-Integral-Derivative (PID) Controller. The first experimental results of the RBF Neural Network MRAC were presented. Validation of autopilot Stabilis, developed in house, is done with the flight testing with RBF Neural Network MRAC is unfavorable flying conditions such as cross winds of 17 knots, gusting upto 23 knots integrating with an aircraft not specified to fly at these conditions. Results show that GP-MRAC outperforms RBF-NN MRAC and PID in terms of the tracking error. The results from the Hardware in the loop testing demonstrates the feasibility to transfer controllers from one platform to another using the adaptive controller GP-MRAC.

6.2 Future Work

The recommendations for future work are as follows

- Derivation of bounds and stable regions for GP - MRAC with $\hat{\Delta}$ estimation using a high gain traditional MRAC.

- Flight Testing with GP - MRAC Architecture with fixed wing UAV, as intensive HITL testing (more than 300 hours) is performed with the different airframes
- Characterize effectiveness of the Control Transfer using GP-MRAC by flight testing with various fixed-wing aircrafts
- Extending the Control Transfer to a different class of airframes such as quadcopters, helicopters, unconventional airframes, etc.

APPENDIX A

Hardware in the Loop Testing Results

The results of the Hardware in the loop testing with the other aircrafts are presented in this section.

A.1 Mugin

The Waypoint Tracking performance with the three different controllers for the aircraft Mugin is shown. The circles represents the waypoints that the aircraft has to fly autonomously.

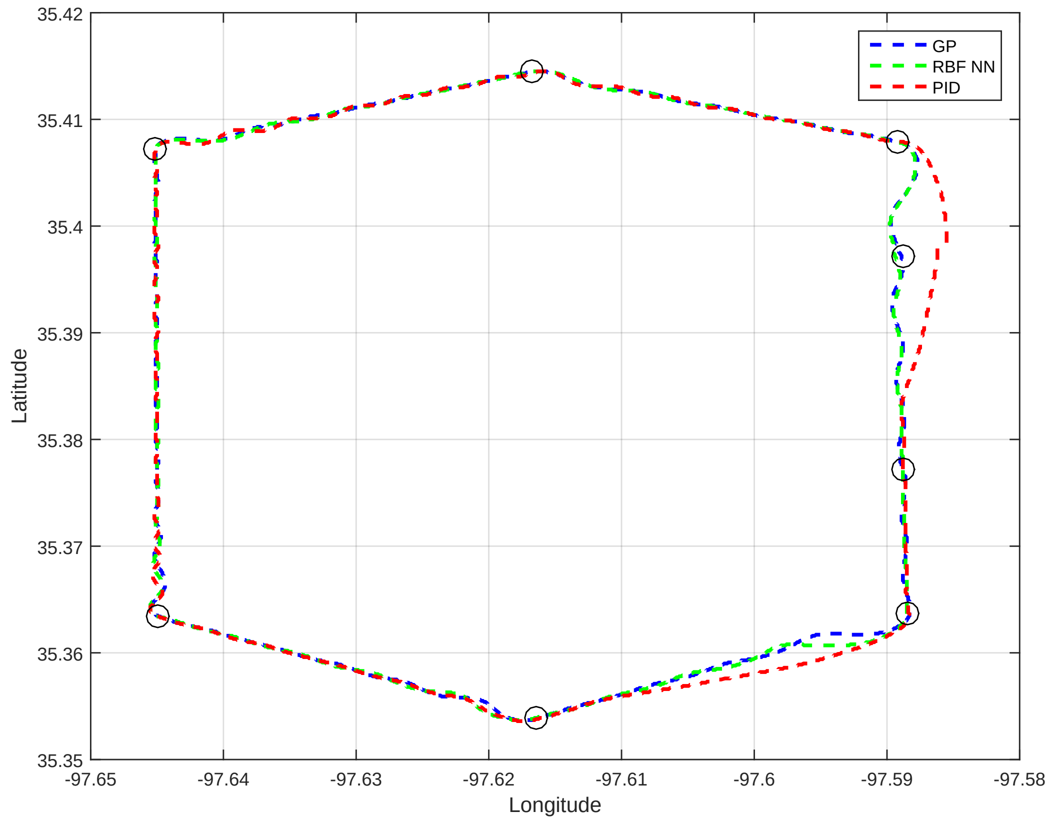
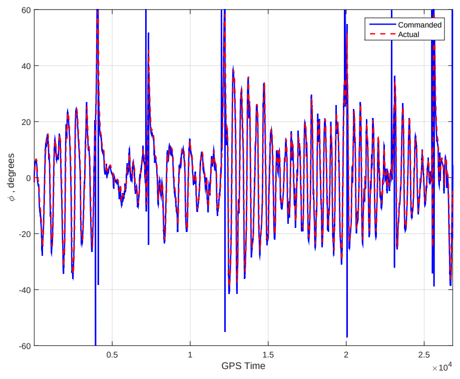
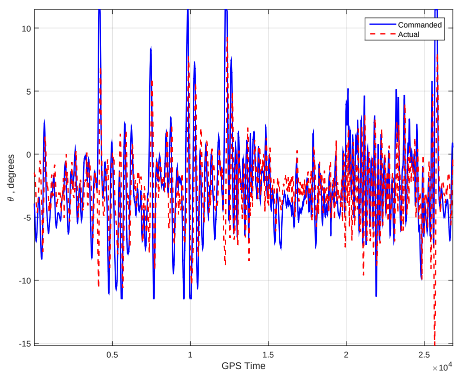


Figure A.1: Comparison of Waypoint Tracking in Mugin with the different controllers

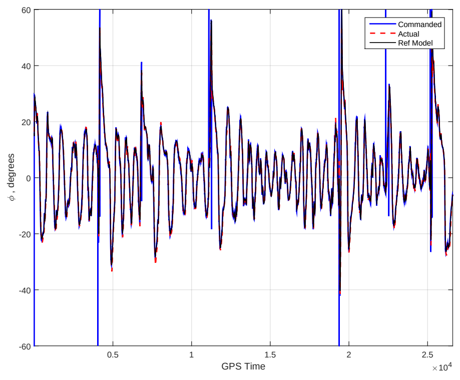
The tracking of commanded input form the outer loops for both roll (ϕ) and pitch (θ) with the different controllers for a single lap are shown.



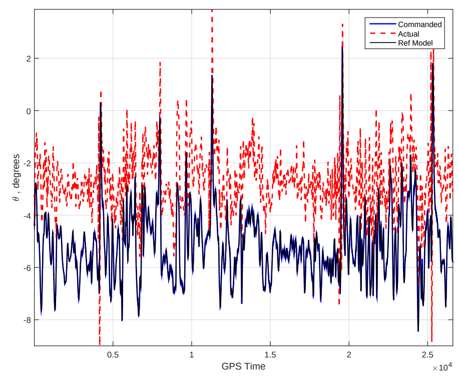
(a) Tracking Performance for Roll Control using PID



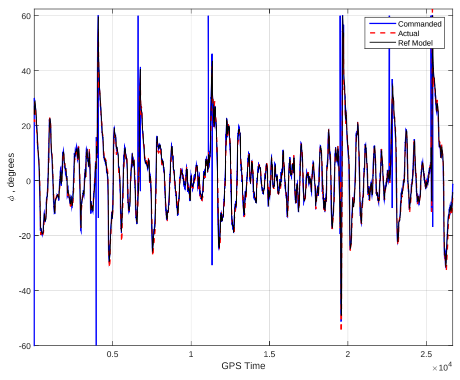
(b) Tracking Performance for Pitch Control using PID



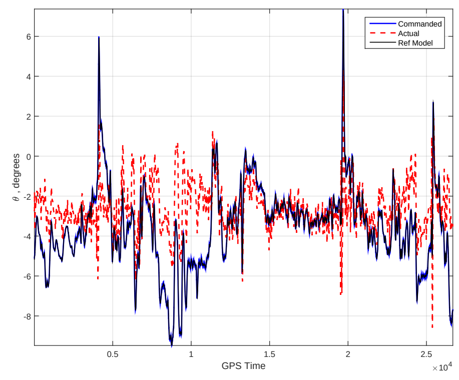
(c) Tracking Performance for Roll Control using RBF-NN MRAC



(d) Tracking Performance for Pitch Control using RBF-NN MRAC



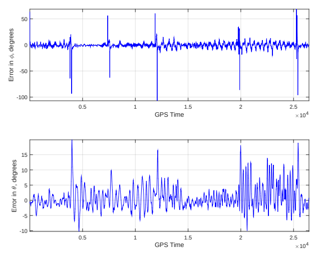
(e) Tracking Performance for Roll Control using GP-MRAC



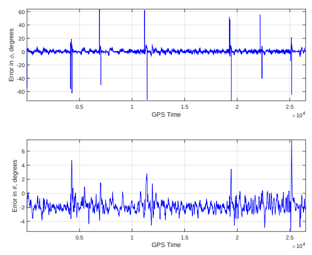
(f) Tracking Performance for Pitch Control using GP-MRAC

Figure A.2: Tracking Performance of Roll and Pitch with the different controllers in Mugin

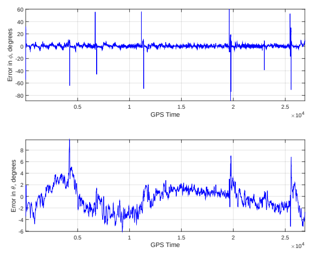
The Evolution of the Inner Loop errors and Outer loop errors are shown.



(a) Inner Loop Errors using the PID Controller

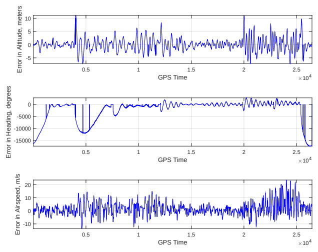


(b) Inner Loop Errors using RBF-NN MRAC

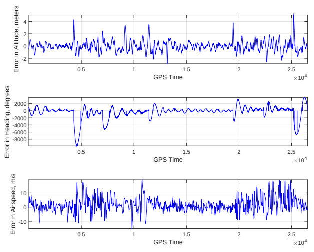


(c) Inner Loops Errors using GP-MRAC

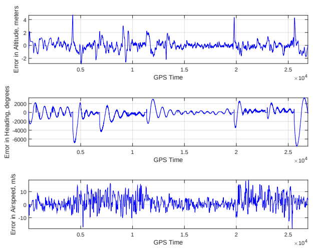
Figure A.3: Evolution of Inner Loop Errors in Skyhunter with the various Controllers



(a) Outer Loop Errors using the PID Controller



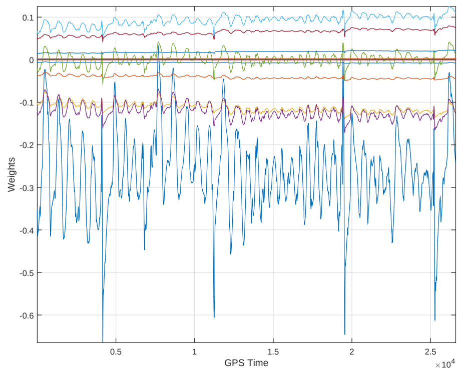
(b) Outer Loop Errors using RBF-NN MRAC



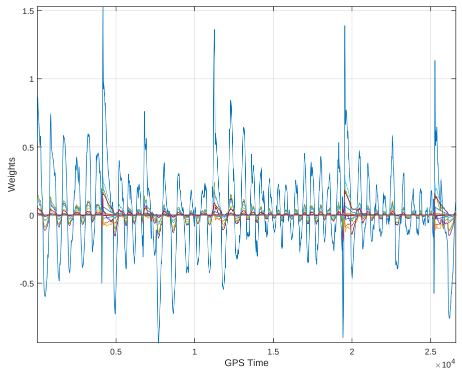
(c) Outer Loops Errors using GP-MRAC

Figure A.4: Evolution of Outer Loop Errors in Mugin with the various Controllers

The Evolution of the RBF-NN weights for roll (ϕ) and pitch (θ) for a single lap while the aircraft is tracking the waypoints from the mission.



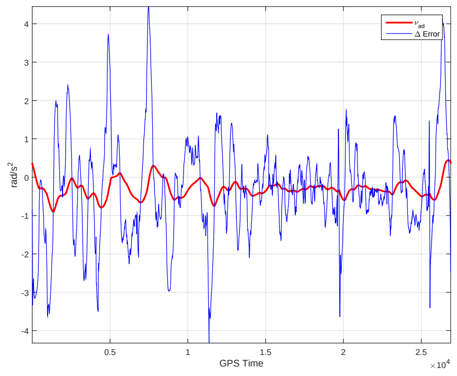
(a) Evolution of MRAC weights in Roll Dynamics



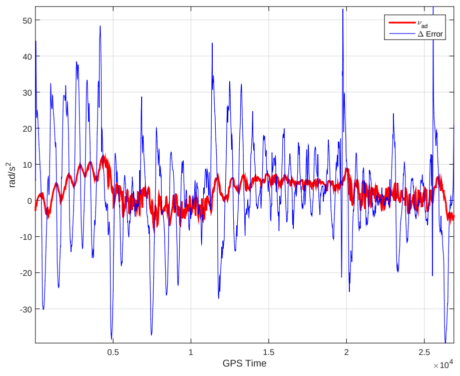
(b) Evolution of MRAC weights in Pitch Dynamics

Figure A.5: Evolution of MRAC weights in Inner Loop Dynamics

Performance of Gaussian Processes in capturing the uncertainty for a single lap while the aircraft is tracking the waypoints from the mission.



(a) Modeling Error in Roll Dynamics in GP-MRAC



(b) Modeling Error in Pitch Dynamics in GP-MRAC

Figure A.6: Performance of GPs in capturing the uncertainty

A.2 Anaconda

The Waypoint Tracking performance with the three different controllers for the aircraft Anaconda is shown. The circles represents the waypoints that the aircraft has to fly autonomously.

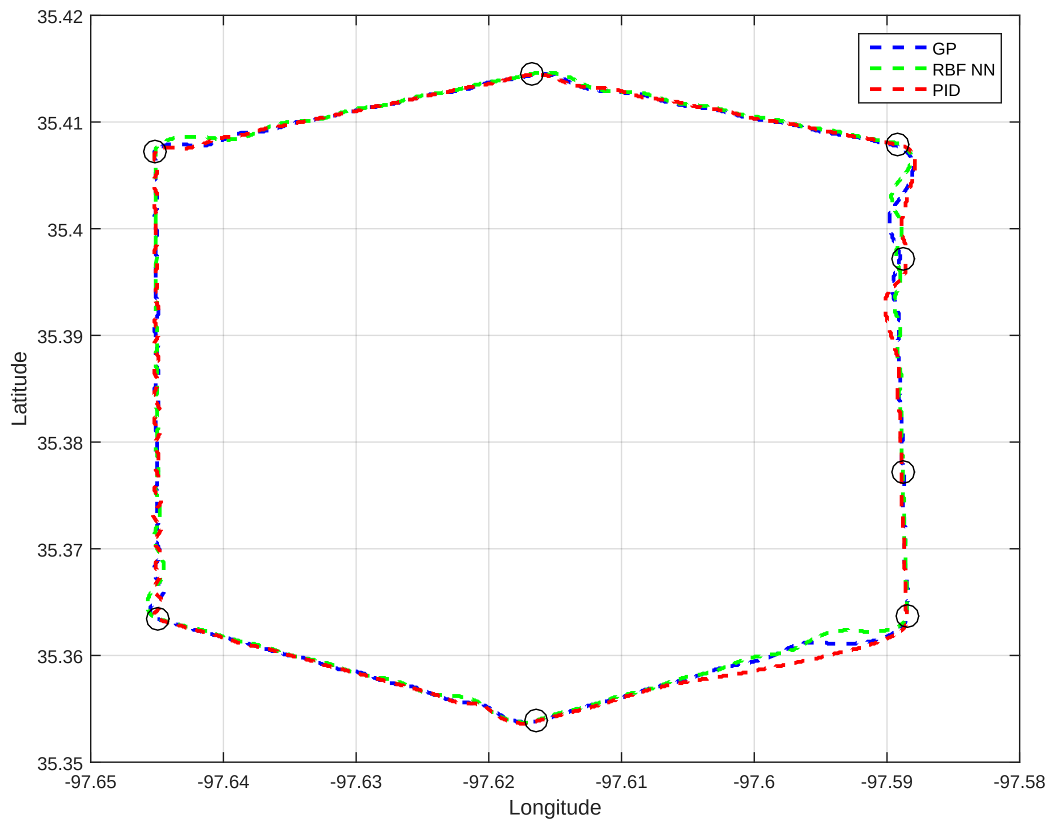
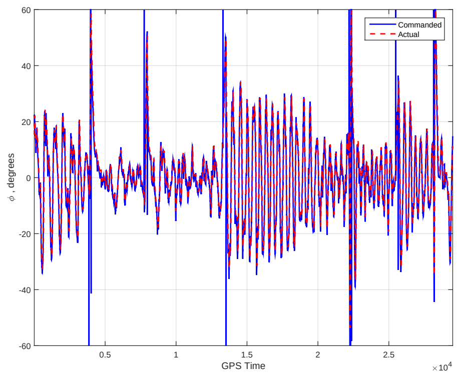
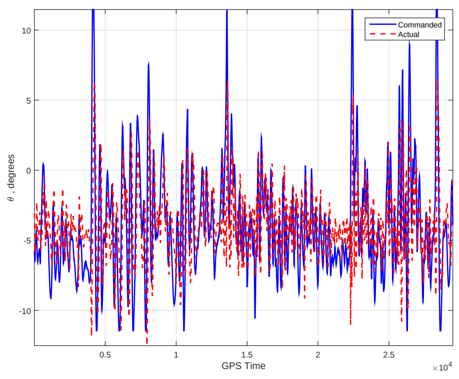


Figure A.7: Comparison of Waypoint Tracking in Anaconda with the different controllers

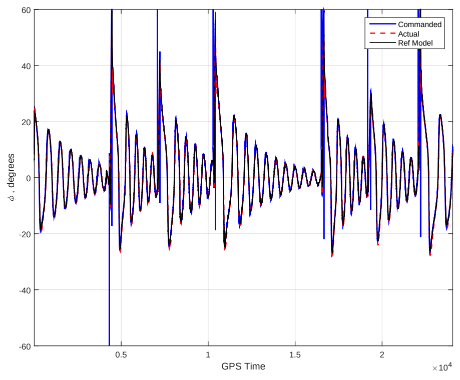
The tracking of commanded input form the outer loops for both roll (ϕ) and pitch (θ) with the different controllers for a single lap are shown.



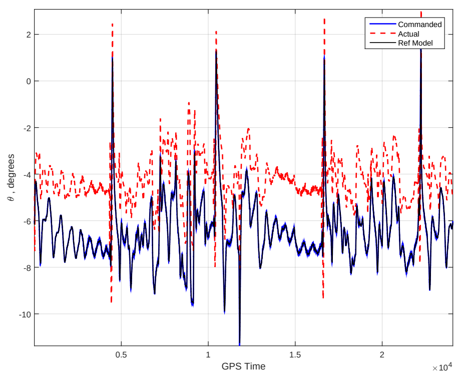
(a) Tracking Performance for Roll Control using PID



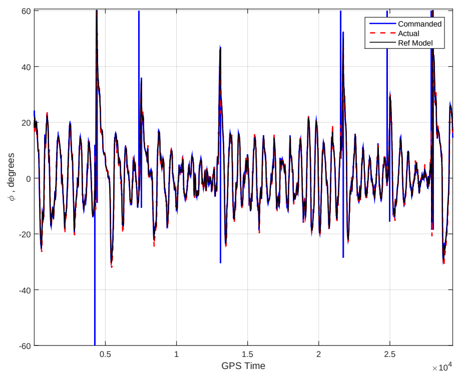
(b) Tracking Performance for Pitch Control using PID



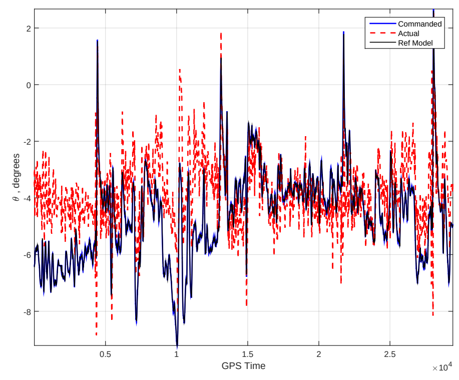
(c) Tracking Performance for Roll Control using RBF-NN MRAC



(d) Tracking Performance for Pitch Control using RBF-NN MRAC



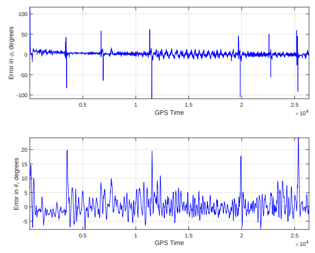
(e) Tracking Performance for Roll Control using GP-MRAC



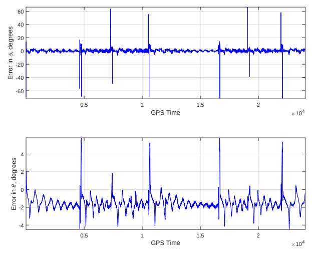
(f) Tracking Performance for Pitch Control using GP-MRAC

Figure A.8: Tracking Performance of Roll and Pitch with the different controllers in Anaconda

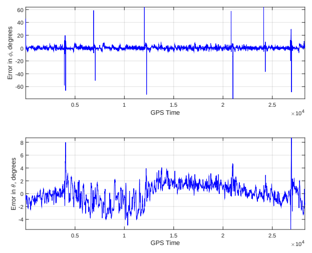
The Evolution of the Inner Loop errors and Outer loop errors are shown.



(a) Inner Loop Errors using the PID Controller

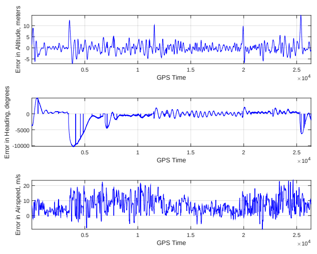


(b) Inner Loop Errors using RBF-NN MRAC

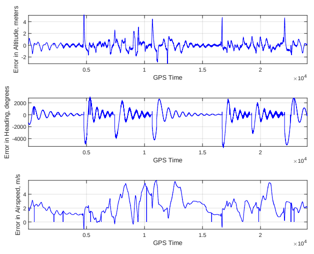


(c) Inner Loops Errors using GP-MRAC

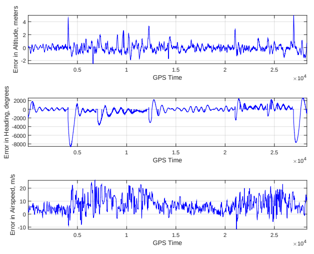
Figure A.9: Evolution of Inner Loop Errors in Anaconda with the various Controllers



(a) Outer Loop Errors using the PID Controller



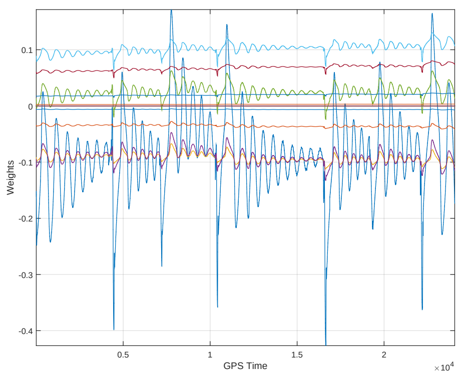
(b) Outer Loop Errors using RBF-NN MRAC



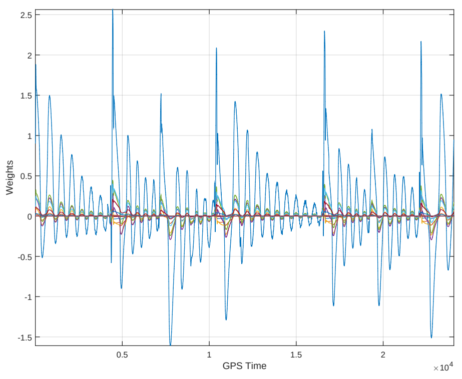
(c) Outer Loops Errors using GP-MRAC

Figure A.10: Evolution of Outer Loop Errors in Anaconda with the various Controllers

The Evolution of the RBF-NN weights for roll (ϕ) and pitch (θ) for a single lap while the aircraft is tracking the waypoints from the mission.



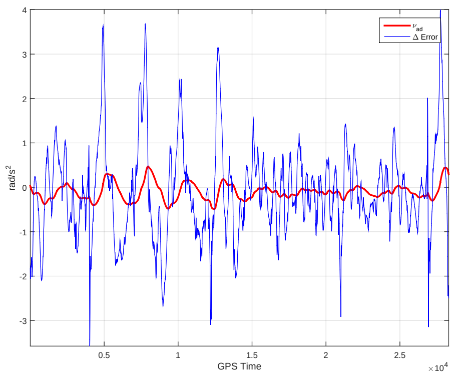
(a) Evolution of MRAC weights in Roll Dynamics



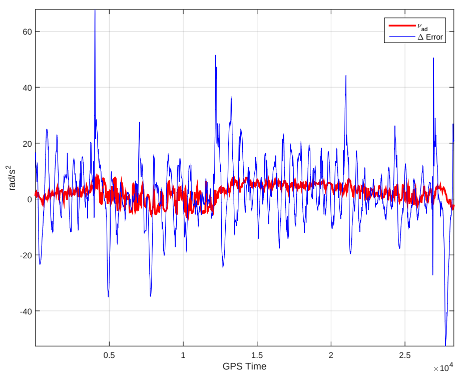
(b) Evolution of MRAC weights in Pitch Dynamics

Figure A.11: Evolution of MRAC weights in Inner Loop Dynamics

Performance of Gaussian Processes in capturing the uncertainty for a single lap while the aircraft is tracking the waypoints from the mission.



(a) Modeling Error in Roll Dynamics in GP-MRAC



(b) Modeling Error in Pitch Dynamics in GP-MRAC

Figure A.12: Performance of GPs in capturing the uncertainty

A.3 Penguin - B

The Waypoint Tracking performance with the three different controllers for the aircraft oenguin-B is shown. The circles represents the waypoints that the aircraft has to fly autonomously.

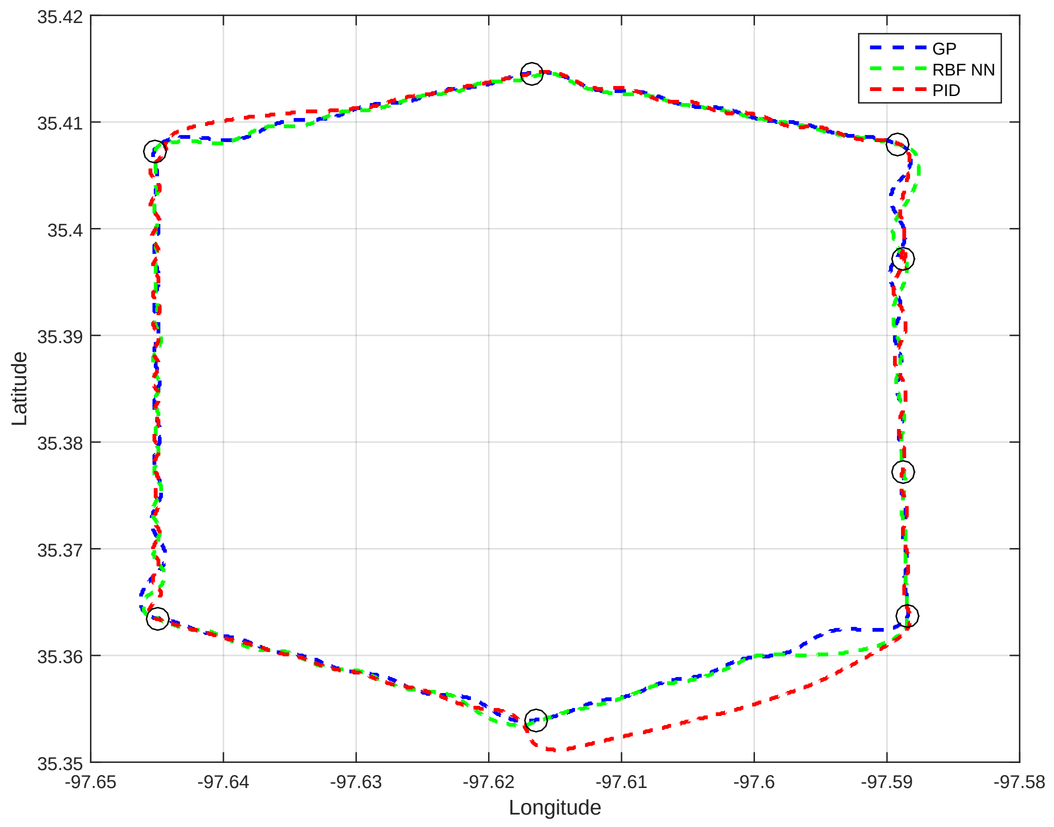
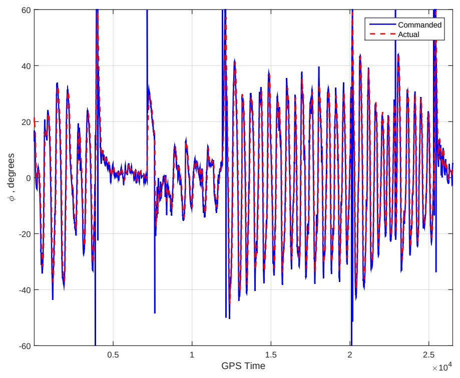
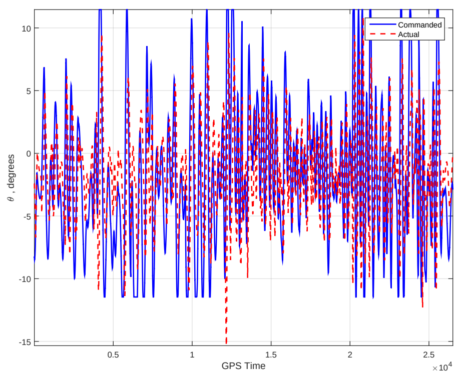


Figure A.13: Comparison of Waypoint Tracking in Penguin-B with the different controllers

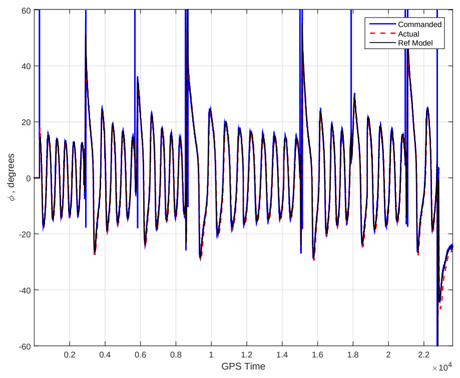
The tracking of commanded input form the outer loops for both roll (ϕ) and pitch (θ) with the different controllers for a single lap are shown.



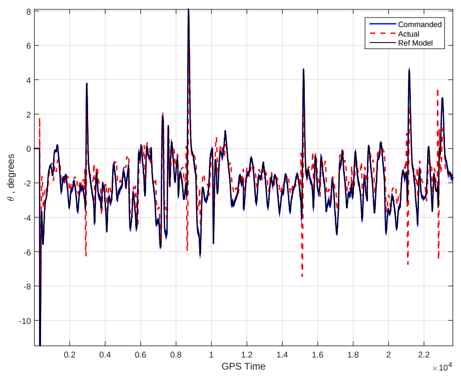
(a) Tracking Performance for Roll Control using PID



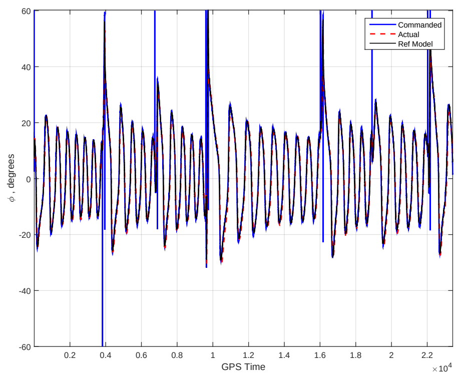
(b) Tracking Performance for Pitch Control using PID



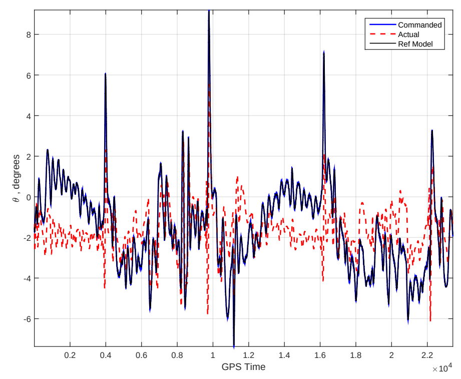
(c) Tracking Performance for Roll Control using RBF-NN MRAC



(d) Tracking Performance for Pitch Control using RBF-NN MRAC



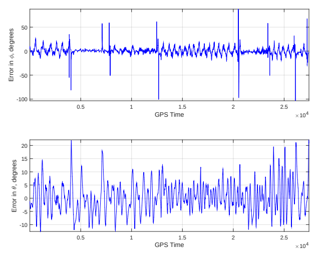
(e) Tracking Performance for Roll Control using GP-MRAC



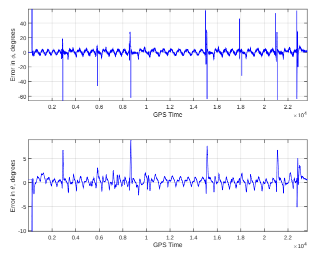
(f) Tracking Performance for Pitch Control using GP-MRAC

Figure A.14: Tracking Performance of Roll and Pitch with the different controllers in Penguin-B

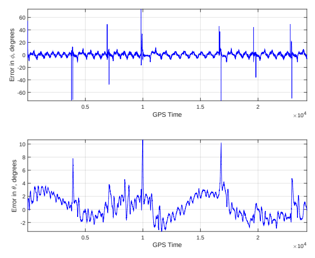
The Evolution of the Inner Loop errors and Outer loop errors are shown.



(a) Inner Loop Errors using the PID Controller

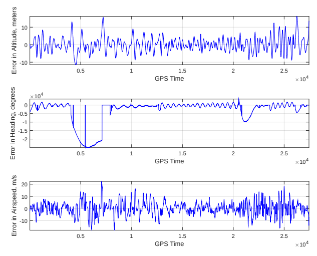


(b) Inner Loop Errors using RBF-NN MRAC

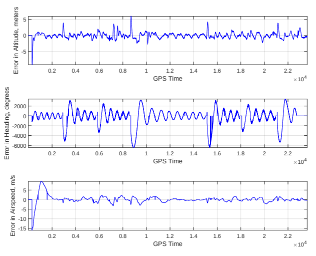


(c) Inner Loops Errors using GP-MRAC

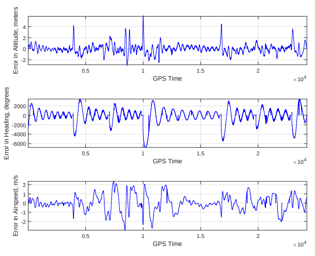
Figure A.15: Evolution of Inner Loop Errors in Anaconda with the various Controllers



(a) Outer Loop Errors using the PID Controller



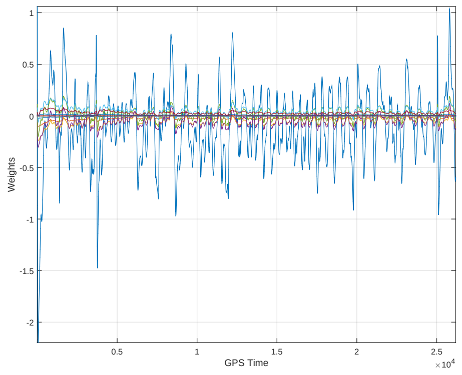
(b) Outer Loop Errors using RBF-NN MRAC



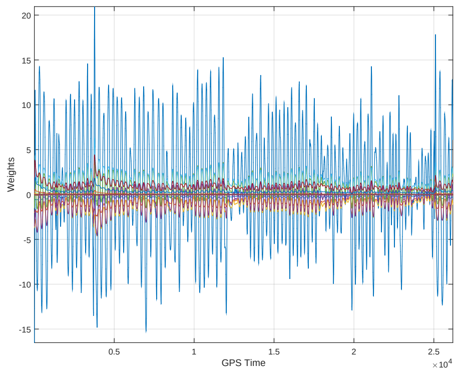
(c) Outer Loops Errors using GP-MRAC

Figure A.16: Evolution of Outer Loop Errors in Anaconda with the various Controllers

The Evolution of the RBF-NN weights for roll (ϕ) and pitch (θ) for a single lap while the aircraft is tracking the waypoints from the mission.



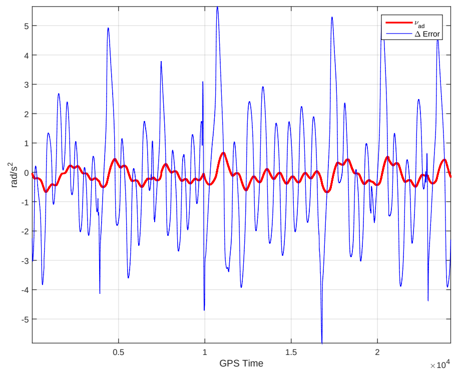
(a) Evolution of MRAC weights in Roll Dynamics



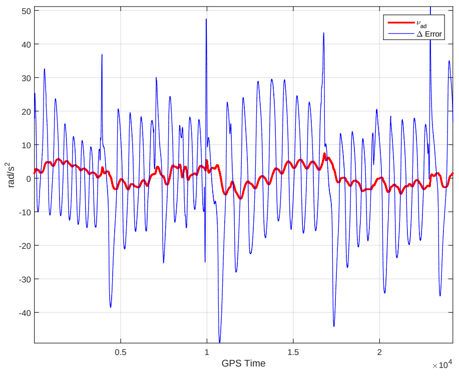
(b) Evolution of MRAC weights in Pitch Dynamics

Figure A.17: Evolution of MRAC weights in Inner Loop Dynamics

Performance of Gaussian Processes in capturing the uncertainty for a single lap while the aircraft is tracking the waypoints from the mission.



(a) Modeling Error in Roll Dynamics in GP-MRAC



(b) Modeling Error in Pitch Dynamics in GP-MRAC

Figure A.18: Performance of GPs in capturing the uncertainty

APPENDIX B

Autopilot Specifications

Specifications for the autopilots benchmarked in Table **??** are provided below. These specifications were used to aid in selecting components for Stabilis. It should be noted that many of the autopilot companies do not readily advertise the specifications of their product. Thus, unfortunately, a significant amount of information was not provided since it was not disclosed.

Table B.2: Commercial Off the Shelf Autopilots Specifications - Availability of I/O

	Serial I/O								Digitial I/O			ADC
	RS232	RS422	RS485	UART	I2C	SPI	CAN	Eth.	PWM	PPM	SBUS	
Kestrel 2.2	4 Serial Ports (STD, SPI, I2C)								10	-	-	2 - 12bit
MP 2028g	-	-	-	-	-	-	-	-	24	-	-	>1
Piccolo Nano	3	-	-	-	-	-	1	-		-	-	-
Piccolo LT	3	-	-	-	-	-	1	-		-	-	-
Piccolo II	3	-	-	-	-	-	1	-	16	-	-	up to 4
Unav3521	-	-	-	-	-	-	-	-	4	-	-	-
osflexPilot[2]	>1	>1	>1	>1	>1	>1	>1	-	>1	-	-	>1
osnanoPilot[2]	-	-	-	>1	>1	>1	>1	-	8	-	-	-
osflexQuad[2]	-	-	-	-	>1	-	>1	-	8	-	-	-
Slugs	-	-	-	-	-	-	-	-	-	-	-	-
PixHawk	-	-	-	5	1	1	2	-	-	1	1	1 - 12bit
Ardupilot	-	-	-	2	1	-	-	-	8	-	-	12
Swiftpilot	-	-	-	-	-	-	-	-	6	-	-	-
wePilot1000/3000	1	-	-	-	-	-	-	-	10	-	-	6 - 12bit
SkyCircuit-SC2[1]	-	-	-	-	-	-	-	1	6	-	-	-
SmartAP	-	-	-	1	-	-	-	-	6	-	-	2
Paparazzi	-	-	-	3	2	2	1	-	6	1	-	1
GNC1000	2	1	8	-	-	-	2	4	6	-	-	-

Note 1) Expansion boards available.

Note 2) Specific values not advertised

Table B.3: Commercial Off the Shelf Autopilots Specifications - Availability of Sensors

	Rate Gyroscopes			Accelerometers			Air Speed		Max	
	Range	Res.	Bias	Range	Bias	Noise	Range	Res.	Alt.	Op. Temp
	(°/s)	(°/s)	(°/vHr)	(g)	(mg/LSB)	(g/vHz)	(kts)	(kts)	(ft)	(°C)
Kestrel 2.2	300	0.0318	6	10	1.50	200	252	0.05	22k	-40 to $+85$
MP 2128g	-	-	-	-	-	-	500	-	12k	-
Piccolo Nano	-	-	-	-	-	-	-	-	-	-30 to $+80$
Piccolo LT	300	-	-	6	-	-	192	-	-	-30 to $+80$
Piccolo II	300	-	-	10	-	-	155	-	-	-40 to $+80$
Unav3521	300	-	-	-	-	-	-	-	-	-
osflexPilot	-	-	-	-	-	-	-	-	-	-
osnanoPilot	-	-	-	-	-	-	-	-	-	-
osflexQuad	-	-	-	-	-	-	-	-	-	-
Slugs	-	-	-	-	-	-	-	-	-	-
PixHawk[1]	250	0.0088	1.8	8	0.244	~600	193	1.1	>60k	-30 to $+70$
Ardupilot[1]	1000	32.8	3	8	0.2	400	60	0.5	>60k	-30 to $+70$
Swiftpilot	2000	-	-	-	-	-	-	-	-	0 to 70
wePilot1000/3000	100	-	-	10	-	-	-	-	10k	-
SkyCircuit-SC2	-	-	-	-	-	-	-	-	15	-20 to $+60$
SmartAP[1]	1000	32.8	3	8	0.2	400	External		>60k	-30 to $+70$
Paparazzi[2]	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
GNC1000	200/1000	-	3	10	5	-	450	1	60k	-

Note 1) FS values assumed; the resolution and range are dependent on this assumption.

APPENDIX C

Component Benchmarking

C.1 Flight Control Computer Survey

A summary of some of the flight control computers that were considered are provided in C.4.

Table C.4: Embedded system specifications

Model	Clock Speed	Memory	Price	Peripherals
Arduino Due	84MHz	128Kb	40	I2C(2), UART(2), GPIO(28), 12 bit ADC(2), UDP/TCP, USB
Arietta-G25	400 MHz	128Kb	30	I2C x 2, UART x 2, GPIO x 28, 12 bit ADC x 2
Beaglebone	400 MHz	128Kb	40	I2C x 2, UART x 2, GPIO x 28, 12 bit ADC x 2
Beaglebone Black	1 GHz	128Kb	55	I2C x 2, UART x 2, GPIO x 28, 12 bit ADC x 2
RaspberryPi 3	700 MHz	128Kb	40	I2C x 2, UART x 2, GPIO x 28, 12 bit ADC x 2

C.2 Inertial Sensor Survey

The following list is a compilation of available COTS IMU/INS/AHRS sensors. Table C.5 was used primarily in the early design phases of Stabilis in order to characterize and select an appropriate inertial navigation system. It is provided as a reference to the reader.

Table C.5: List of Inertial Sensors

Model	Manufacturer	Type	GPS	Cost
Daisy-7	ACME Systems	MEMS	Yes	128.7
Adjacent Reality	open hardware	MEMS	no	
Spatial	Advanced Navigation	MEMS	Yes	3000
Spatial Dual	Advanced Navigation	MEMS	Yes	10000
Spatial FOG	Advanced Navigation	FOG	Yes	35000
ARN-NS0535	Aeron Systems	MEMS		
impactAIMS	AIMS	MEMS		
uMotion	AIMS	MEMS		
Navigation	AIMS	MEMS	yes	
FOG	AIMS	MEMS/FOG		
AHRS/INS	American GNC	MEMS		
AHRS/INS/GPS	American GNC	MEMS	yes	
AHRS/INS/DGPS	American GNC	MEMS	yes	
ADIS16355	Analog Devices	MEMS		600
Opal	APDM	MEMS		
AHR150A-1	Archangel Sys.	MEMS		
3D-Bird	Ascension	MEMS		1768
INU	Atair Aerospace	MEMS	yes	
Micro INS	Athena (Rockwell)	MEMS	yes	
SensorPac	Athena (Rockwell)	MEMS	yes	
SilMU 01	UTC Aerospace (BAE)	MEMS		
SilMU 02	UTC Aerospace (BAE)	MEMS		
SiNAV 02	UTC Aerospace (BAE)	MEMS	yes	
MMQ 50	BEI Systron Donner	MEMS		
MMQ-G	BEI Systron Donner	MEMS	yes	
C-MIGITS III	BEI Systron Donner	MEMS	yes	
MiniSense 2	CDLTD	MEMS	input	
TOGS	CDLTD	RLG	input	
MiniRLG2	CDLTD	RLG	input	
MiniPOS2	CDLTD	RLG	input	

Continued on next page

Table C.5 – *Continued from previous page*

Model	**Manufacturer**	**Type**	**GPS**	**Cost**
MiniPOS/NAV	CDLTD	RLG	input	
MiniTilt	CDLTD	MEMS		
MicroTilt	CDLTD	MEMS		
MiniSense	CDLTD	MEMS		
ADAHRS	Chelton Avionics	MEMS	input	26000
CHR-6d	CH Robotics	MEMS	no	125
CHR-6dm	CH Robotics	MEMS	no	
CHR-6um6	CH Robotics	MEMS	no	
GP9	CH Robotics	MEMS	yes	320
Crista	Cloudcap	MEMS		2000
Piccolo	Cloudcap	MEMS	yes	6000
Terrella 6	Clymer Tech.	MEMS		1300
NAV 420	Crossbow	MEMS	yes	
NAV 425EX	Crossbow	MEMS	yes	
NAV 440	Crossbow	MEMS	yes	6000
AHRS500	Crossbow	MEMS		14200
IMU440	Crossbow	MEMS	yes	
IMU700CB	Crossbow	FOG		12000
Landmark 10 IMU	Gladiator Tech.	MEMS		2495
Landmark 10 IMU/GPS	Gladiator Tech.	MEMS	yes	4995
Landmark 20 IMU	Gladiator Tech.	MEMS		3995
Landmark 20 IMU/GPS	Gladiator Tech.	MEMS	yes	5995
Landmark 10 GPS/AHRS	Gladiator Tech.	MEMS	yes	
Landmark 10 AHRS	Gladiator Tech.	MEMS		
Landmark 30	Gladiator Tech.			6600
HG 1700	Honeywell	RLG		9000
AG-1	Icewire	MEMS	No	199
iNAV-FMS-T	iMAR	FOG	input	
iIMU-FSAS	iMAR	FOG		
iIMU-FR-M1	iMAR		input	
iVRU-FAS-C167-IGS	iMAR	FOG/MEMS	input	
iVRU-FC-C167-MSL	iMAR	FOG/MEMS	input	

Continued on next page

Table C.5 – *Continued from previous page*

Model	**Manufacturer**	**Type**	**GPS**	**Cost**
iVRU-SSA-C167	iMAR	FOG/MEMS	input	
iVRU-SSKS-C167	iMAR	MEMS	input	
iVRU-SBA1-C167	iMAR	FOG/MEMS	input	
iVRU-FA-C167	iMAR	MEMS	input	
iVRU-FKS-C167	iMAR	FOG/MEMS	input	
iTGAC-FK	iMAR	FOG/MEMS		
iHRP(Y)	iMAR	FOG/RLG	input	
iNAV-FMS	iMAR	FOG/RLG	yes	
iDIS-FMS	iMAR	FOG	yes	
iFLY	iMAR			
iuIMU-02	iMAR	MEMS	yes	
iTraceRT-F200	iMAR	FOG	yes	
OptoAHRS	Inertial Labs	Optical/MEMS	no	7499
AHRS-1	Inertial Labs	MEMS	no	3499
AHRS-2	Inertial Labs	MEMS	no	2999
VG	Inertial Labs	MEMS	no	2699
OS3D	Inertial Labs	MEMS	no	999
OS3DM	Inertial Labs	MEMS	no	999
ISIS-IMU	Inertial Science	MEMS		
ISIS-GPS	Inertial Science	MEMS	yes	
DMARS-R	Inertial Science	MEMS	input	
DMARS-I	Inertial Science	MEMS	input	
DMARS-GARS	Inertial Science	MEMS	yes	
InertiaCube2	InterSense	MEMS		1500
InertiaCube3	InterSense	MEMS		1800
MPU-9150	Invensense	MEMS	no	80
MPU-6000	Invensense	MEMS		15
KN-4072	Kearfott	RLG		
KN-4072A	Kearfott	RLG	yes	
KN-4073B	Kearfott	RLG		
KN-4074	Kearfott	RLG	yes	
KN-4075	Kearfott	RLG	yes	

Continued on next page

Table C.5 – *Continued from previous page*

Model	**Manufacturer**	**Type**	**GPS**	**Cost**
KN-4077	Kearfott	RLG	yes	
KN-4051/2/3	Kearfott			
KN-4051/2/3G	Kearfott		yes	
KI-4801	Kearfott	RLG		
KI-4901	Kearfott	RLG	input	
KI-4902	Kearfott	RLG	input	
TG-6000	KVH	FOG		25000
CNS-5000	KVH	FOG/MEMS	yes	30250
LPMS-B	LP Research	MEMS	no	500
LPMS-CU	LP Research	MEMS	no	400
micro IMU	Memsense	MEMS		
nano IMU	Memsense	MEMS		2730
MIDG II	Microbotics	MEMS	yes	6750
MP 2028g	MicroPilot	MEMS	yes	5000
3DM-GX1	MicroStrain	MEMS		1500
3DM-GX2	MicroStrain	MEMS		
3DM-GX3-25	MicroStrain	MEMS		2295
3DM-GX3-35	MicroStrain	MEMS	yes	3075
3DM	MicroStrain	MEMS		
3DM-DH	MicroStrain	MEMS		
INERTIA-LINK	MicroStrain	MEMS		
LN-200	Northrop-Grumman			
Summit 34203A	Omni Instr.	MEMS		
Falcon GX	O-Navi	MEMS		1000
Phoenix AX	O-Navi	MEMS	yes	1200
UM6	Pololu	MEMS	input	200
FreeIMU	open hardware	MEMS		
AHRS200A	Rotomotion	MEMS		
AHPRS200A	Rotomotion	MEMS	yes	
CHIMU	Ryan Mechatronics	MEMS	No	299
Nav Board M3	Ryan Mechatronics	MEMS	No	299
IG-500A	SBG Systems	MEMS	no	2208.7

Continued on next page

Table C.5 – *Continued from previous page*

Model	**Manufacturer**	**Type**	**GPS**	**Cost**
IG-500N	SBG Systems	MEMS	yes	4483.7
IG-500E	SBG Systems	MEMS	yes	
Ekinox INS	SBG Systems	MEMS	yes	32500
Ekinox AHRS	SBG Systems	MEMS	no	
MoveIt Senspod	Sensaris	MEMS	yes	
STIM300	Sensaris	MEMS	yes	7800
STM32F3	ST Semiconductors	MEMS	no	10.66
65210A	Summit Instr.		yes	
65210E	Summit Instr.		yes	
CompaNav 2	Teknol	MEMS	input	
CompaNav 2T	Teknol	MEMS	input	
Autopilot	Teknol	MEMS	input	6000
Nanosatellite	Tethers Unlimited	MEMS	input	
CC2541 DevKit	Texas Instruments	MEMS	no	25
Colibri	Trivisio	MEMS	no	550
Colibri wireless	Trivisio	MEMS	no	800
Atom	UAV NAvigation	MEMS	input	
Polar	UAV NAvigation	MEMS	yes	
Vector	UAV NAvigation	MEMS	yes	
Proton	UAV NAvigation	MEMS	yes	
VN-100	Vectonav	MEMS	no	500
VN-200	Vectonav	MEMS	yes	500
VN-200 dev. kit	Vectonav	MEMS	yes	2900
x-IMU	x-io	MEMS	no	249
MTi-10 IMU	Xsens	MEMS		1170
MTi-20 VRU	Xsens	MEMS		2080
MTi-30 AHRS	Xsens	MEMS		2340
MTi-100 IMU	Xsens	MEMS		1820
MTi-200 VRU	Xsens	MEMS		3380
MTi-300 AHRS	Xsens	MEMS		4940
MTi-G-700 GPS/INS	Xsens	MEMS	yes	4940
3-Space USB	YEI	MEMS	no	145

Continued on next page

Table C.5 – *Continued from previous page*

Model	**Manufacturer**	**Type**	**GPS**	**Cost**
3-Space Embedded	YEI	MEMS	no	99
3-Space Wireless 2.4G	YEI	MEMS	no	220
3-Space Bluetooth	YEI	MEMS	no	290
3-Space Data-logging	YEI	MEMS	no	180
3-Space Data-logging HH	YEI	MEMS	no	192

APPENDIX D

Setup of Hardware in the Loop

D.1 Introduction

This chapter will detail in procuring and setting up the different softwares used for running the Hardware in The Loop (HITL) Demonstration for the autopilot Stabilis. This will begin with the list of softwares required, then a short tutorial on installing the aforementioned softwares, then having setup the softwares, the required hardware to do the demonstration.

D.2 Softwares

D.2.1 Softwares Required

- BeagleBone Black Drivers
- `PuTTY`
- WinSCP
- Eclipse IDE for C/C++
- X-Plane Flight Simulator
- QGroundControl Ground Control Station

Note: The following Instructions are shown with the softwares X-Plane v.10.41, QGroundControl v.2.71 and Eclipse Luna

D.3 Installation of Softwares

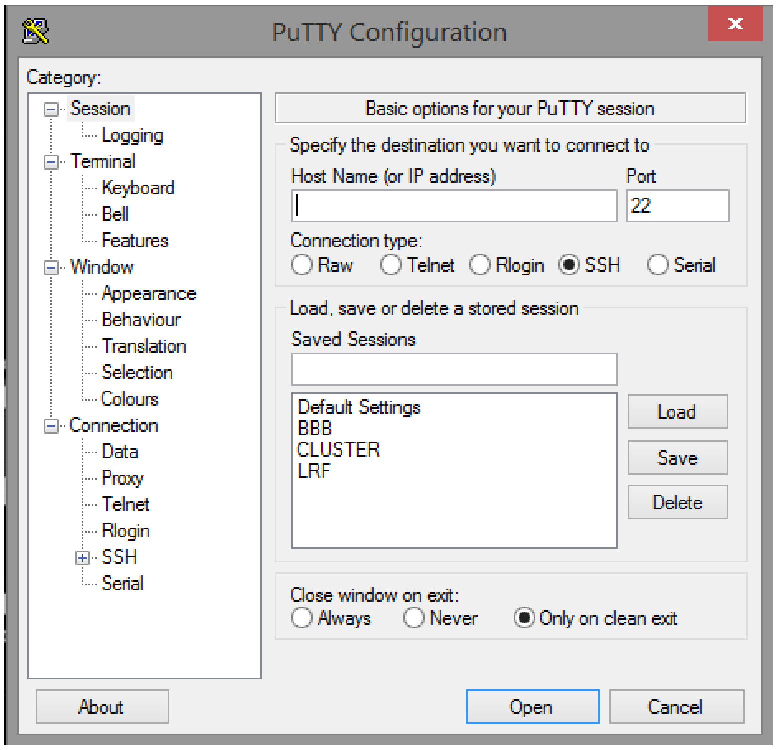
D.3.1 BeagleBone Black Drivers

To connect to STABILIS, as the module is using a BeagleBone Black, we need to install the drivers for the BeagleBlack Bone. The drivers can be downloaded at `http://beagleboard.org/getting-started` and follow the steps given in the website for BeagleBone.

D.3.2 PuTTY

To connect to STABILIS using secure shell (ssh) protocol, open an ssh client (`putty.exe`) for Windows, which can be downloaded at: `http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html`, the `PuTTY` download page.

When you open `PuTTY`, the following screen will appear:



Specify the destination you want to connect to

Host Name (or IP address): Port: 22

Connection type: Raw Telnet Rlogin [✓ SSH] Serial

Load, save or delete a stored session

Saved Sessions

Default Settings
BBB
CLUSTER
LRF

Close window on exit: Always Never [✓ Only on clean exit]

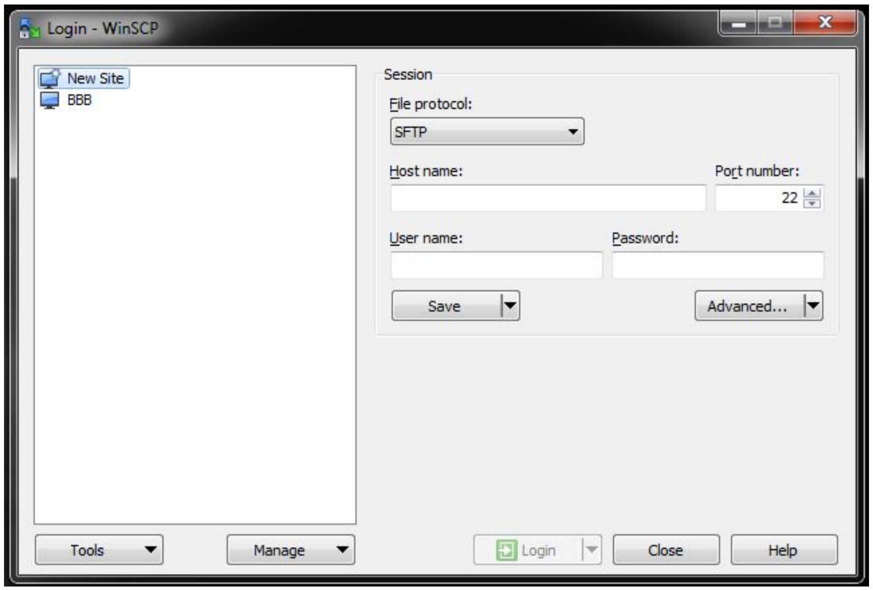
Figure D.1: `PuTTY` Connection

Enter the **STABILIS IP ADDRESS: 192.168.7.2** in the **Host Name** field. Then under Saved Sessions, type a name you would like to identify the connection to the stabilis by (I used ‘BBB’) and click **save**. From that point, you can simply double-click the name you gave to secure shell into the stabilis any time you open your `PuTTY` client.

D.3.3 WinSCP

To transfer files securely to and from STABILIS, open a STFP client(WinSCP.exe) which can be downloaded at: `https://winscp.net/eng/download.php`. You can either use the Installation Package or the Portable Executables.

When you open WinSCP, the following screen will appear(Figure : D.2):



File protocol: SFTP

Host name:

Port number: 22

User name:

Password:

Figure D.2: WinSCP Connection

Enter the **STABILIS IP ADDRESS: 192.168.7.2** in the **Host Name** field. Then under Saved Sessions, type a name you would like to identify the connection to the stabilis by (I used ‘BBB’), enter the **username** and **password**, then click **save**. From that point, you can simply double-click the name you gave to secure shell into the stabilis any time you open your WinSCP client.

D.3.4 Eclipse IDE for C/C++

To view and modify the files on **STABILIS**, open a remote system explorer(Eclipse IDE for C/C++) which can be downloaded at : `http://www.eclipse.org/downloads/`

, where there is a link for the Eclipse IDE for C/C++. Proceed to the download link and download the package and install it.

After installing the Eclipse IDE for C/C++, open the IDE and set up a workspace. Then the following steps will guide you through to setup the Remote System Explorer in the Eclipse IDE.

Opening the Remote System Explorer Perspective

- Go to $Windows \rightarrow OpenPerspective \rightarrow Other...$
- Select **Remote System Explorer** in the shown menu
- Click OK (Figure : D.3)

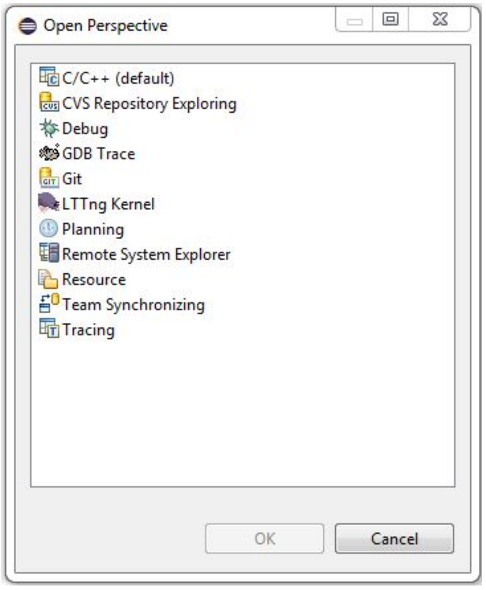


Figure D.3: Open Perspective Window

- The window will now populate the Remote System Explorer(RSE) view in Eclipse IDE (Figure : D.4)

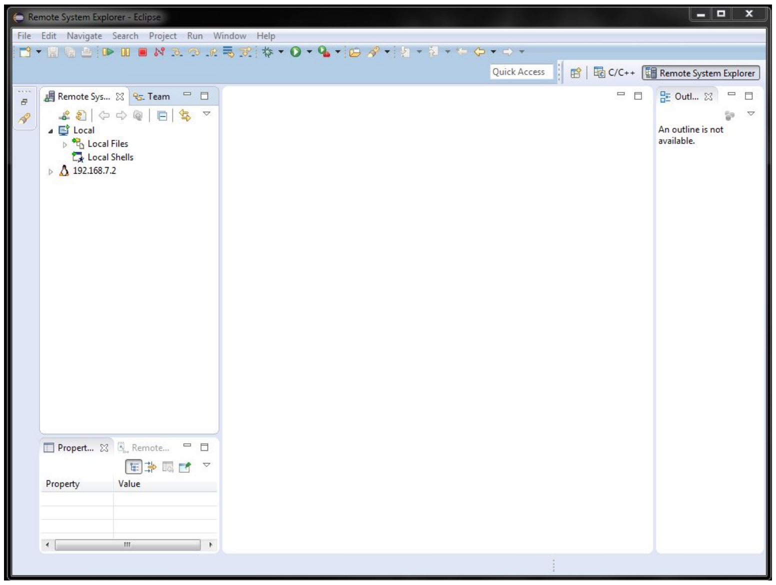


Figure D.4: Remote System Explorer View

Creating a New Connection

- Go to *File* → *New* → *Other...*
- Select *RemoteSystemExplorer* → *Connection*
- Click **Next** (Figure : D.5)

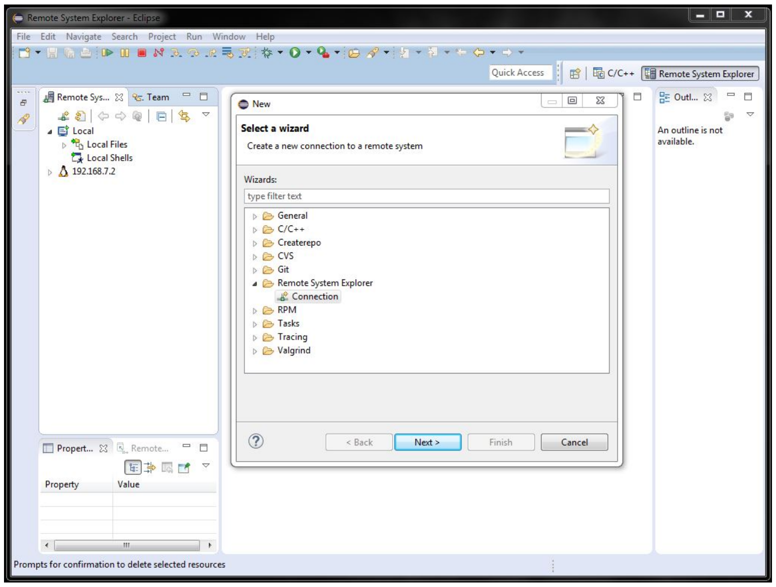


Figure D.5: New Remote Connection

- In the **Select Remote System Type** window, select **Linux** and Click **Next** (Figure : D.6)
- Enter the **STABILIS IP ADDRESS: 192.168.7.2** in the **Host Name** field.
- Fill the Connection Name (**STABILIS**) and Click **Next** (Figure : D.7)
- Select **ssh.files** and click **Next** (Figure : D.8)
- Select the **processes.shell.linux** and click **Next** (Figure : D.9)
- Select the **ssh.shells** and click **Next** (Figure : D.10)
- Select **ssh.terminals** and Click **Finish** (Figure : D.11)
- Now you can observe that the Remote System Explorer for **STABILIS** is setup in the Remote Systems Tab (Figure : D.12)

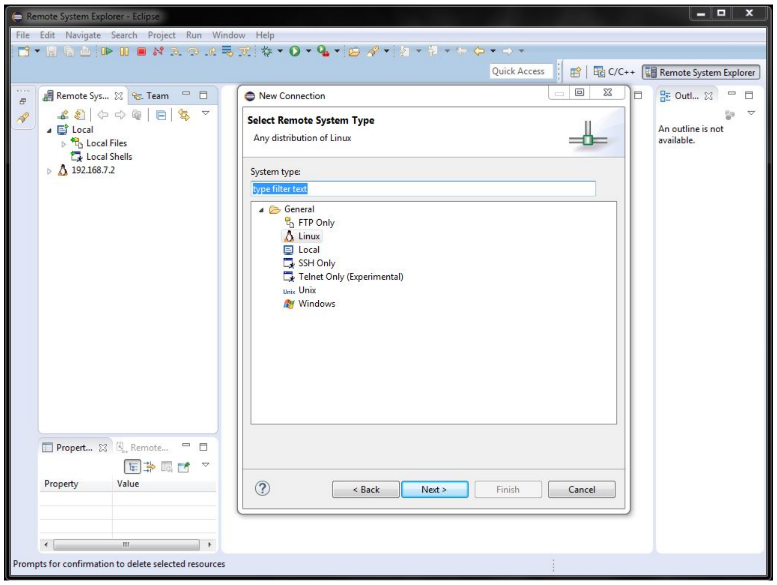


Figure D.6: Select Remote Sytem Type : Linux

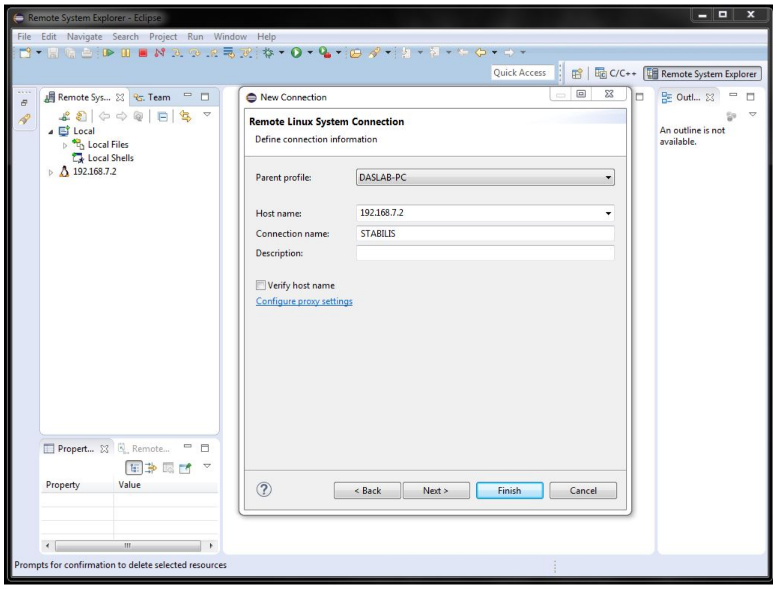
D.3.5 X-Plane Fight Simulator

The X-Plane Fight Simulator by Laminar Research is utilized to do the Hardware in the Loop (HITL) Demonstration. Generally, flight simulators emulate the real world performance of an aircraft by using empirical data to determine aerodynamic forces such as drag or lift, which vary in different flight conditions. X-Plane can model fairly complex aircraft designs, including helicopters, rockets, rotor crafts and tilt rotor crafts.

Establishing Net Connections with X-Plane

After installing X-Plane Flight Simulator, Run the X-Plane program.

- Goto to **Settings** in the Menu bar and click **Net Connections** which would load up the following screen (Figure : D.13)
- Select the **Data** Tab which would populate the following screen (Figure : D.14)
- Fill the **IP of data receiver** as **192.168.7.2** and the **Port** as **49001**.(Figure : D.15)
- For the UDP Ports fill the following values to the ports mentioned respectively (Figure : D.16)



Remote Linux System Connection

Define connection information

Parent profile: DASLAB-PC

Host name: 192.168.7.2

Connection name: STABILIS

Description:

Verify host name

Configure proxy settings

Prompts for confirmation to delete selected resources

Figure D.7: Remote Linux System Connection

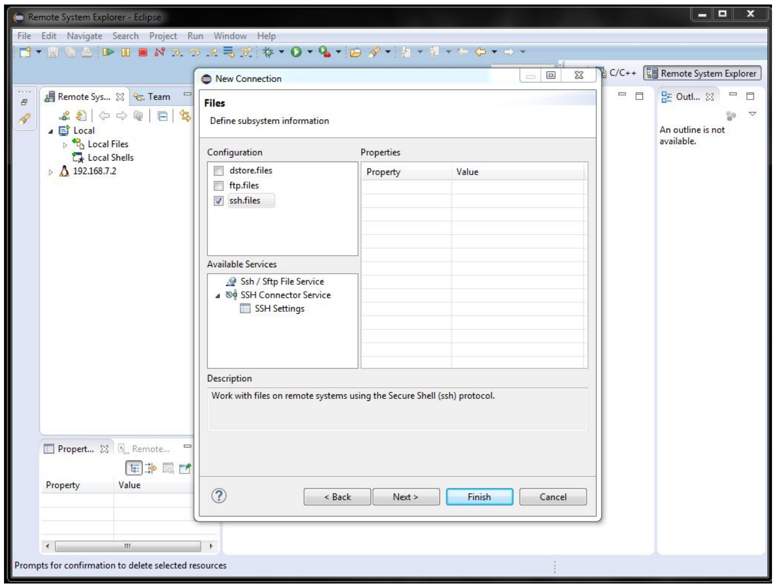
- ports that we receive on : **49,000**
- ports that we send from : **49,001**

Selecion of Data Input & Output from X-Plane to STABILIS

- Goto to **Settings** in the Menu bar and click **Data Input & Output** which would load up the following screen (Figure : D.17)

There are four checkboxes shown for each parameter in the window

- The first checkbox represents **Internet via UDP**
- The second checkbox represents **Disk file 'data.txt'**
- The third checkbox represents **Graphical Display in 'Data See'**
- The fourth checkbox represents **Cockpit During Flight**



New Connection

Files

Define subsystem information

Configuration
- [ ] dstore.files
- [ ] ftp.files
- [x] ssh.files

Properties

Property	Value

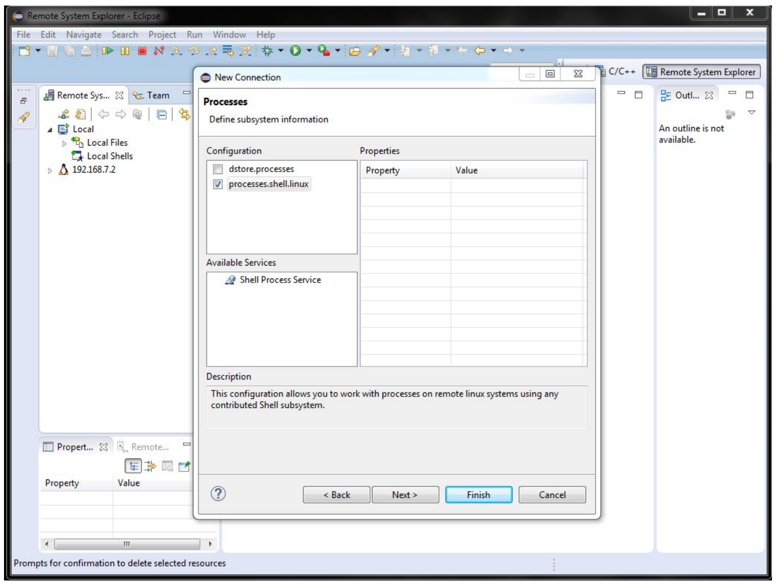
Available Services
- Ssh / Sftp File Service
- SSH Connector Service
  - SSH Settings

Description

Work with files on remote systems using the Secure Shell (ssh) protocol.

Prompts for confirmation to delete selected resources

Figure D.8: Defining the subsytem information(Files) for the new connection



New Connection

Processes

Define subsystem information

Configuration
- [ ] dstore.processes
- [x] processes.shell.linux

Properties

Property	Value

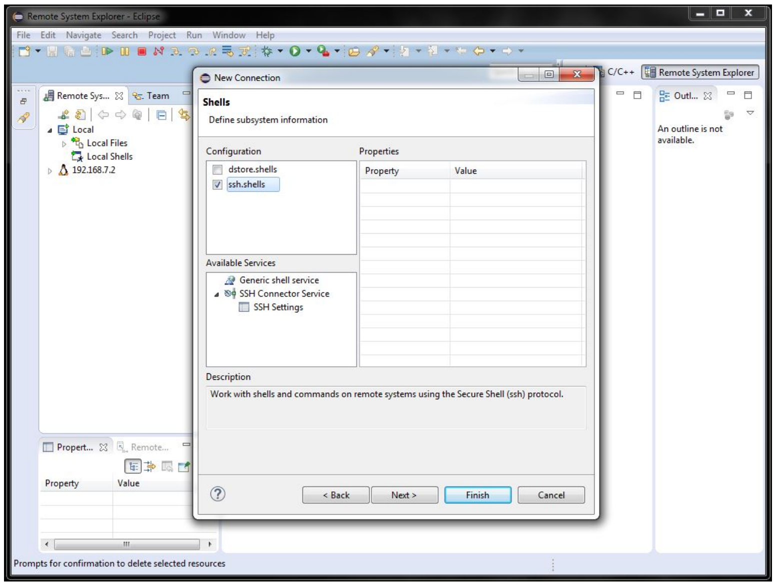
Available Services
- Shell Process Service

Description

This configuration allows you to work with processes on remote linux systems using any contributed Shell subsystem.

Prompts for confirmation to delete selected resources

Figure D.9: Defining the subsytem information(Processes) for the new connection



Shells

Define subsystem information

Configuration

- [ ] dstore.shells
- [✓] ssh.shells

Available Services

- Generic shell service
- SSH Connector Service
  - SSH Settings

Properties

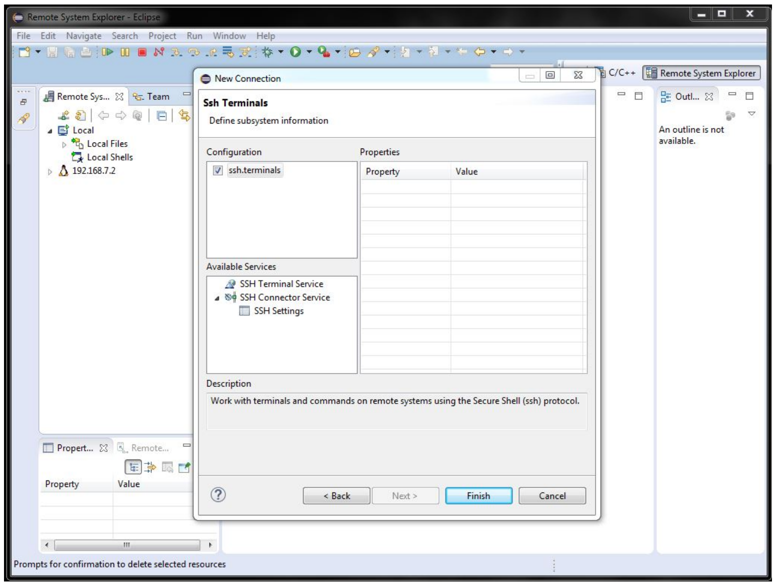
Property	Value

Description

Work with shells and commands on remote systems using the Secure Shell (ssh) protocol.

Prompts for confirmation to delete selected resources

Figure D.10: Defining the subsytem information(Shells) for the new connection



Ssh Terminals

Define subsystem information

Configuration

- [✓] ssh.terminals

Available Services

- SSH Terminal Service
- SSH Connector Service
  - SSH Settings

Properties

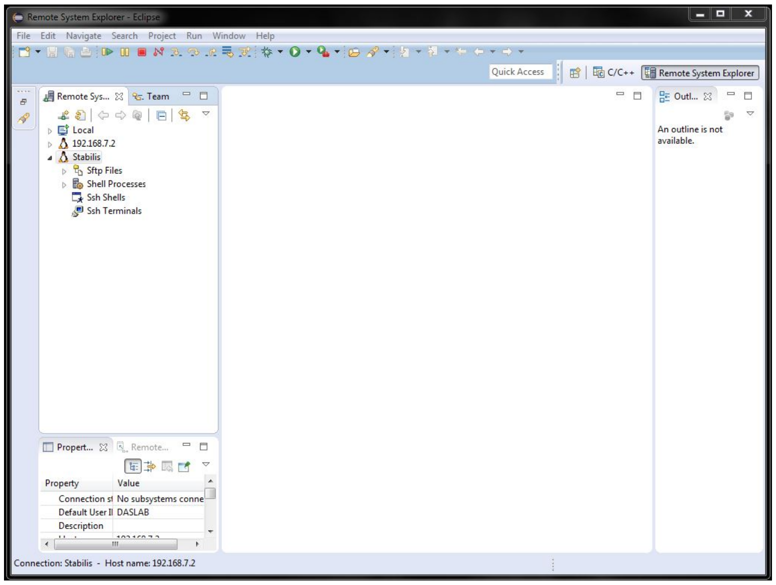
Property	Value

Description

Work with terminals and commands on remote systems using the Secure Shell (ssh) protocol.

Prompts for confirmation to delete selected resources

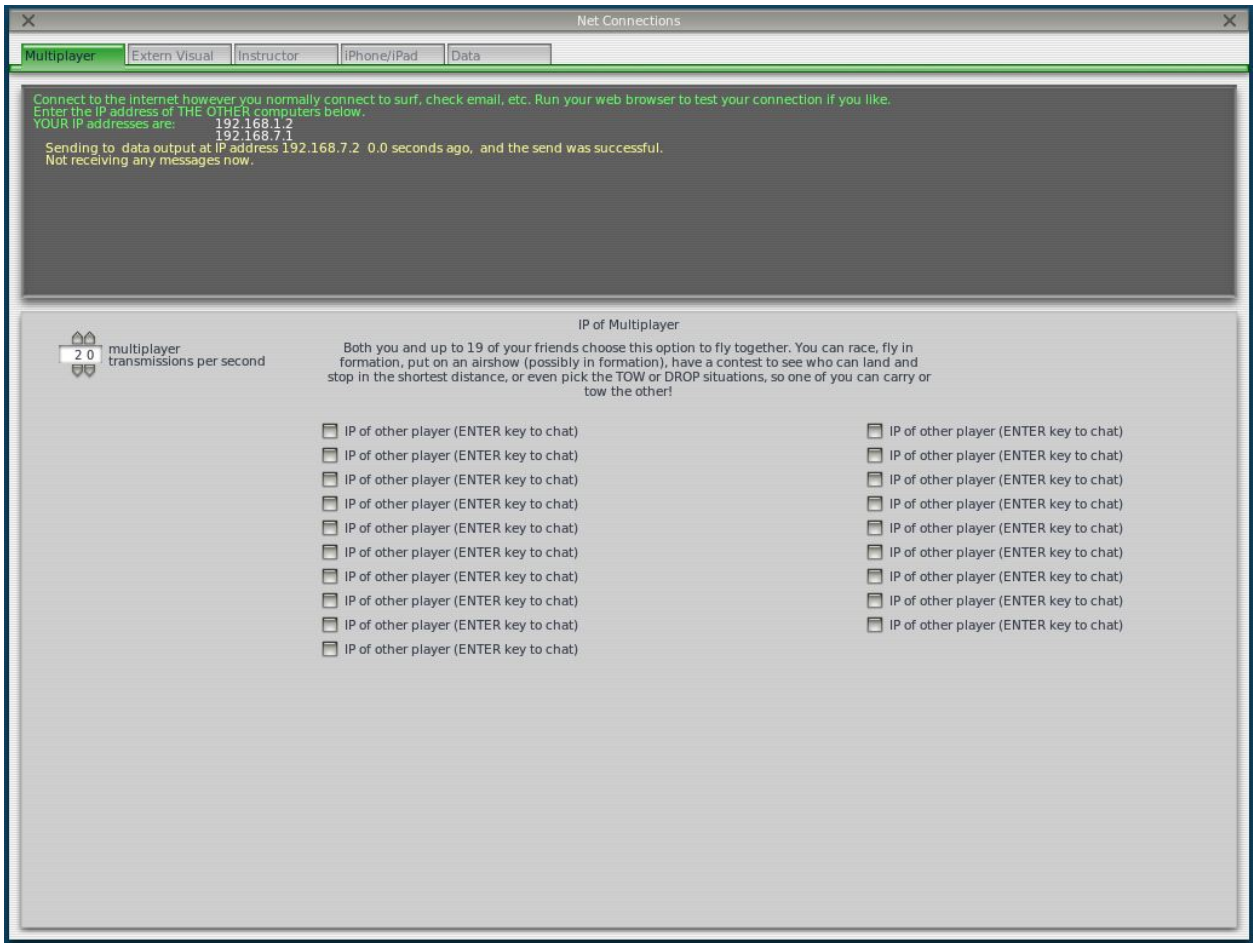
Figure D.11: Defining the subsytem information(SSH Terminals) for the new connection



Property	Value
Connection st	No subsystems conne
Default User II	DASLAB
Description	

Connection: Stabilis - Host name: 192.168.7.2

Figure D.12: Remote System Explorer Setup



Connect to the internet however you normally connect to surf, check email, etc. Run your web browser to test your connection if you like.
Enter the IP address of THE OTHER computers below.
YOUR IP addresses are: 192.168.1.2
192.168.7.1
Sending to data output at IP address 192.168.7.2 0.0 seconds ago, and the send was successful.
Not receiving any messages now.

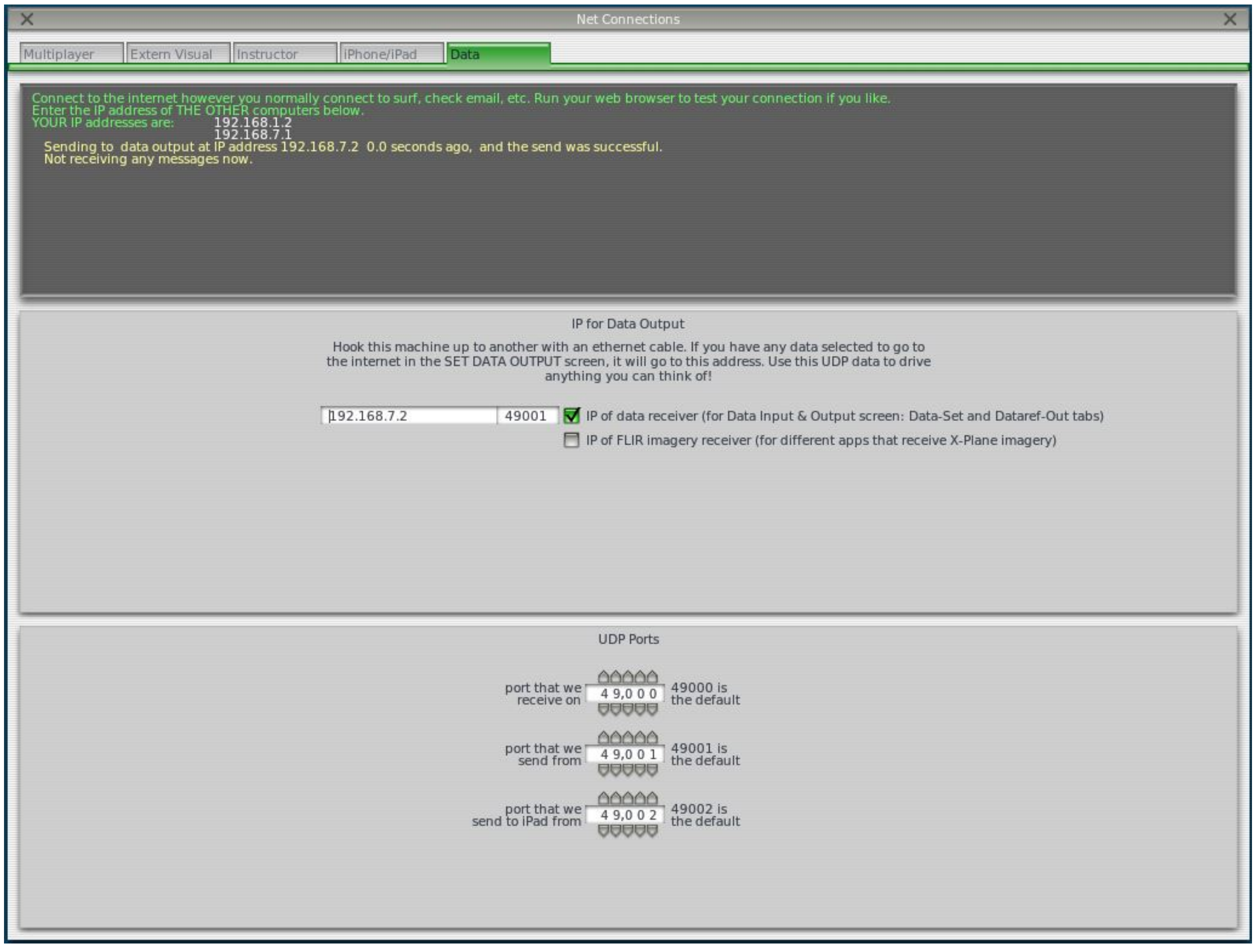
20 multiplayer transmissions per second

IP of Multiplayer

Both you and up to 19 of your friends choose this option to fly together. You can race, fly in formation, put on an airshow (possibly in formation), have a contest to see who can land and stop in the shortest distance, or even pick the TOW or DROP situations, so one of you can carry or tow the other!

☐ IP of other player (ENTER key to chat)	☐ IP of other player (ENTER key to chat)
☐ IP of other player (ENTER key to chat)	☐ IP of other player (ENTER key to chat)
☐ IP of other player (ENTER key to chat)	☐ IP of other player (ENTER key to chat)
☐ IP of other player (ENTER key to chat)	☐ IP of other player (ENTER key to chat)
☐ IP of other player (ENTER key to chat)	☐ IP of other player (ENTER key to chat)
☐ IP of other player (ENTER key to chat)	☐ IP of other player (ENTER key to chat)
☐ IP of other player (ENTER key to chat)	☐ IP of other player (ENTER key to chat)
☐ IP of other player (ENTER key to chat)	☐ IP of other player (ENTER key to chat)
☐ IP of other player (ENTER key to chat)	☐ IP of other player (ENTER key to chat)
☐ IP of other player (ENTER key to chat)	

Figure D.13: Net Connections - Multiplayer in X-Plane



Connect to the internet however you normally connect to surf, check email, etc. Run your web browser to test your connection if you like.
Enter the IP address of THE OTHER computers below.
YOUR IP addresses are: 192.168.1.2
192.168.7.1
Sending to data output at IP address 192.168.7.2 0.0 seconds ago, and the send was successful.
Not receiving any messages now.

IP for Data Output

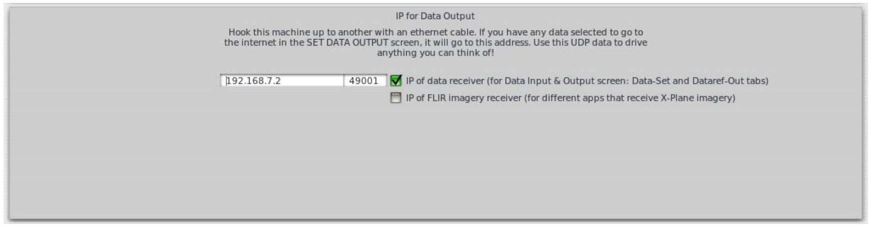
Hook this machine up to another with an ethernet cable. If you have any data selected to go to the internet in the SET DATA OUTPUT screen, it will go to this address. Use this UDP data to drive anything you can think of!

192.168.7.2 49001 [✓] IP of data receiver (for Data Input & Output screen: Data-Set and Dataref-Out tabs)
[ ] IP of FLIR imagery receiver (for different apps that receive X-Plane imagery)

UDP Ports

port that we receive on 4 9,0 0 0 49000 is the default
port that we send from 4 9,0 0 1 49001 is the default
port that we send to iPad from 4 9,0 0 2 49002 is the default

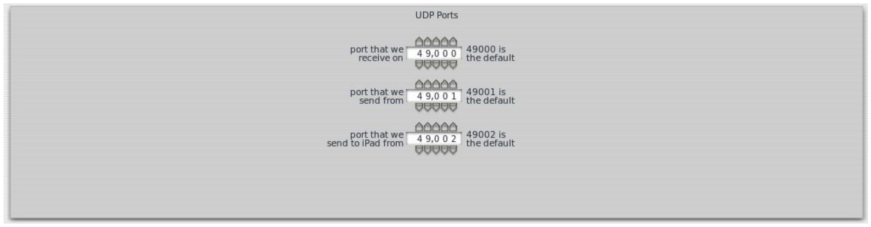
Figure D.14: Net Connections - Data in X-Plane


IP for Data Output

Hook this machine up to another with an ethernet cable. If you have any data selected to go to the internet in the SET DATA OUTPUT screen, it will go to this address. Use this UDP data to drive anything you can think of!

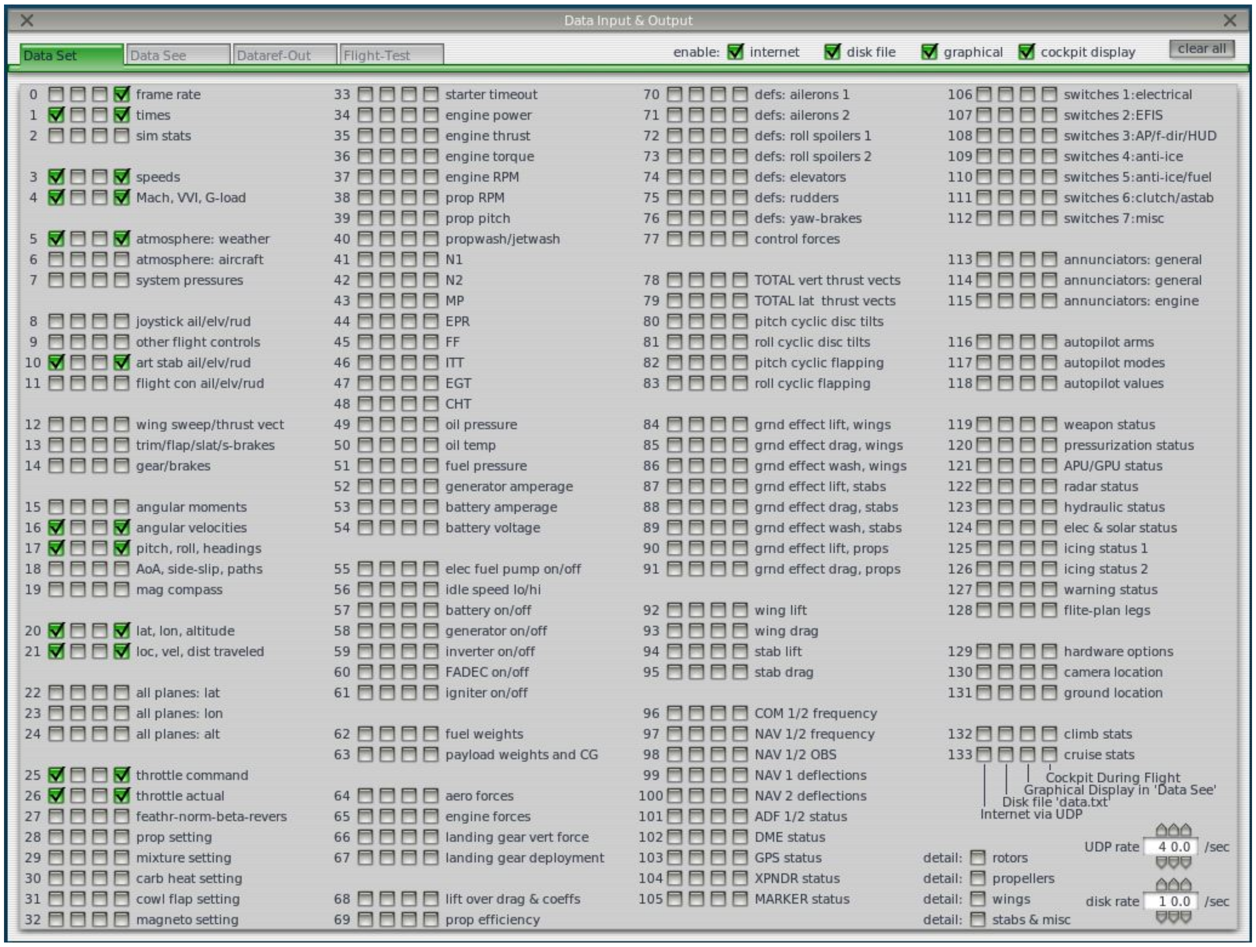
192.168.7.2 49001 [✓] IP of data receiver (for Data Input & Output screen: Data-Set and Dataref-Out tabs)
[ ] IP of FLIR imagery receiver (for different apps that receive X-Plane imagery)

Figure D.15: Net Connections - IP for Data Output


UDP Ports

port that we receive on 4 9,0 0 0 49000 is the default
port that we send from 4 9,0 0 1 49001 is the default
port that we send to iPad from 4 9,0 0 2 49002 is the default

Figure D.16: Net Connections - UDP Ports for data transfer



Data Input & Output

Data Set | Data See | Dataref-Out | Flight-Test

enable: ✓ internet ✓ disk file ✓ graphical ✓ cockpit display — clear all

0	☐	☐	☐	✓	frame rate
1	✓	☐	☐	✓	times
2	☐	☐	☐	☐	sim stats
3	✓	☐	☐	✓	speeds
4	✓	☐	☐	✓	Mach, VVI, G-load
5	✓	☐	☐	✓	atmosphere: weather
6	☐	☐	☐	☐	atmosphere: aircraft
7	☐	☐	☐	☐	system pressures
8	☐	☐	☐	☐	joystick ail/elv/rud
9	☐	☐	☐	☐	other flight controls
10	✓	☐	☐	✓	art stab ail/elv/rud
11	☐	☐	☐	☐	flight con ail/elv/rud
12	☐	☐	☐	☐	wing sweep/thrust vect
13	☐	☐	☐	☐	trim/flap/slat/s-brakes
14	☐	☐	☐	☐	gear/brakes
15	☐	☐	☐	☐	angular moments
16	✓	☐	☐	✓	angular velocities
17	✓	☐	☐	✓	pitch, roll, headings
18	☐	☐	☐	☐	AoA, side-slip, paths
19	☐	☐	☐	☐	mag compass
20	✓	☐	☐	✓	lat, lon, altitude
21	✓	☐	☐	✓	loc, vel, dist traveled
22	☐	☐	☐	☐	all planes: lat
23	☐	☐	☐	☐	all planes: lon
24	☐	☐	☐	☐	all planes: alt
25	✓	☐	☐	✓	throttle command
26	✓	☐	☐	✓	throttle actual
27	☐	☐	☐	☐	feathr-norm-beta-revers
28	☐	☐	☐	☐	prop setting
29	☐	☐	☐	☐	mixture setting
30	☐	☐	☐	☐	carb heat setting
31	☐	☐	☐	☐	cowl flap setting
32	☐	☐	☐	☐	magneto setting
33	☐	☐	☐	☐	starter timeout
34	☐	☐	☐	☐	engine power
35	☐	☐	☐	☐	engine thrust
36	☐	☐	☐	☐	engine torque
37	☐	☐	☐	☐	engine RPM
38	☐	☐	☐	☐	prop RPM
39	☐	☐	☐	☐	prop pitch
40	☐	☐	☐	☐	propwash/jetwash
41	☐	☐	☐	☐	N1
42	☐	☐	☐	☐	N2
43	☐	☐	☐	☐	MP
44	☐	☐	☐	☐	EPR
45	☐	☐	☐	☐	FF
46	☐	☐	☐	☐	ITT
47	☐	☐	☐	☐	EGT
48	☐	☐	☐	☐	CHT
49	☐	☐	☐	☐	oil pressure
50	☐	☐	☐	☐	oil temp
51	☐	☐	☐	☐	fuel pressure
52	☐	☐	☐	☐	generator amperage
53	☐	☐	☐	☐	battery amperage
54	☐	☐	☐	☐	battery voltage
55	☐	☐	☐	☐	elec fuel pump on/off
56	☐	☐	☐	☐	idle speed lo/hi
57	☐	☐	☐	☐	battery on/off
58	☐	☐	☐	☐	generator on/off
59	☐	☐	☐	☐	inverter on/off
60	☐	☐	☐	☐	FADEC on/off
61	☐	☐	☐	☐	igniter on/off
62	☐	☐	☐	☐	fuel weights
63	☐	☐	☐	☐	payload weights and CG
64	☐	☐	☐	☐	aero forces
65	☐	☐	☐	☐	engine forces
66	☐	☐	☐	☐	landing gear vert force
67	☐	☐	☐	☐	landing gear deployment
68	☐	☐	☐	☐	lift over drag & coeffs
69	☐	☐	☐	☐	prop efficiency
70	☐	☐	☐	☐	defs: ailerons 1
71	☐	☐	☐	☐	defs: ailerons 2
72	☐	☐	☐	☐	defs: roll spoilers 1
73	☐	☐	☐	☐	defs: roll spoilers 2
74	☐	☐	☐	☐	defs: elevators
75	☐	☐	☐	☐	defs: rudders
76	☐	☐	☐	☐	defs: yaw-brakes
77	☐	☐	☐	☐	control forces
78	☐	☐	☐	☐	TOTAL vert thrust vects
79	☐	☐	☐	☐	TOTAL lat thrust vects
80	☐	☐	☐	☐	pitch cyclic disc tilts
81	☐	☐	☐	☐	roll cyclic disc tilts
82	☐	☐	☐	☐	pitch cyclic flapping
83	☐	☐	☐	☐	roll cyclic flapping
84	☐	☐	☐	☐	grnd effect lift, wings
85	☐	☐	☐	☐	grnd effect drag, wings
86	☐	☐	☐	☐	grnd effect wash, wings
87	☐	☐	☐	☐	grnd effect lift, stabs
88	☐	☐	☐	☐	grnd effect drag, stabs
89	☐	☐	☐	☐	grnd effect wash, stabs
90	☐	☐	☐	☐	grnd effect lift, props
91	☐	☐	☐	☐	grnd effect drag, props
92	☐	☐	☐	☐	wing lift
93	☐	☐	☐	☐	wing drag
94	☐	☐	☐	☐	stab lift
95	☐	☐	☐	☐	stab drag
96	☐	☐	☐	☐	COM 1/2 frequency
97	☐	☐	☐	☐	NAV 1/2 frequency
98	☐	☐	☐	☐	NAV 1/2 OBS
99	☐	☐	☐	☐	NAV 1 deflections
100	☐	☐	☐	☐	NAV 2 deflections
101	☐	☐	☐	☐	ADF 1/2 status
102	☐	☐	☐	☐	DME status
103	☐	☐	☐	☐	GPS status
104	☐	☐	☐	☐	XPNDR status
105	☐	☐	☐	☐	MARKER status
106	☐	☐	☐	☐	switches 1:electrical
107	☐	☐	☐	☐	switches 2:EFIS
108	☐	☐	☐	☐	switches 3:AP/f-dir/HUD
109	☐	☐	☐	☐	switches 4:anti-ice
110	☐	☐	☐	☐	switches 5:anti-ice/fuel
111	☐	☐	☐	☐	switches 6:clutch/astab
112	☐	☐	☐	☐	switches 7:misc
113	☐	☐	☐	☐	annunciators: general
114	☐	☐	☐	☐	annunciators: general
115	☐	☐	☐	☐	annunciators: engine
116	☐	☐	☐	☐	autopilot arms
117	☐	☐	☐	☐	autopilot modes
118	☐	☐	☐	☐	autopilot values
119	☐	☐	☐	☐	weapon status
120	☐	☐	☐	☐	pressurization status
121	☐	☐	☐	☐	APU/GPU status
122	☐	☐	☐	☐	radar status
123	☐	☐	☐	☐	hydraulic status
124	☐	☐	☐	☐	elec & solar status
125	☐	☐	☐	☐	icing status 1
126	☐	☐	☐	☐	icing status 2
127	☐	☐	☐	☐	warning status
128	☐	☐	☐	☐	flite-plan legs
129	☐	☐	☐	☐	hardware options
130	☐	☐	☐	☐	camera location
131	☐	☐	☐	☐	ground location
132	☐	☐	☐	☐	climb stats
133	☐	☐	☐	☐	cruise stats

Cockpit During Flight
Graphical Display in 'Data See'
Disk file 'data.txt'
Internet via UDP

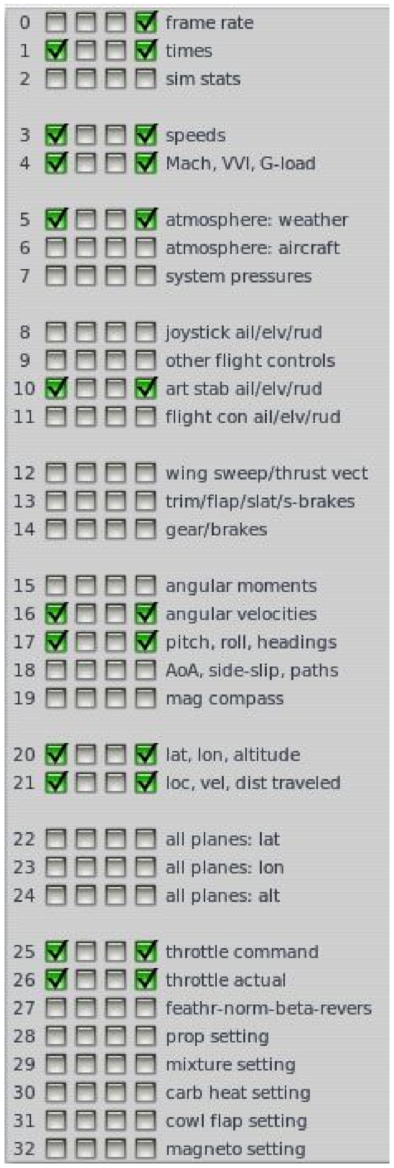
detail: ☐ rotors
detail: ☐ propellers
detail: ☐ wings
detail: ☐ stabs & misc

UDP rate 40.0 /sec
disk rate 10.0 /sec

Figure D.17: Net Connections - UDP Ports for Data Transfer

The following parameters are selected to send to **STABILIS** over the UDP connection(First Checkbox)

- 1 times
- 3 speeds
- 4 Mach,VVI,G-load
- 5 atmosphere:weather
- 10 art stab ail/elv,rud
- 16 angular velocities
- 17 pitch, roll, headings
- 20 lat,lon,altitude
- 21 loc,vel,dist traveled
- 25 throttle command
- 26 throttle actual

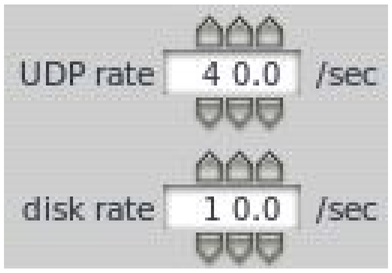


0	☐	☐	☐	☑	frame rate
1	☑	☐	☐	☑	times
2	☐	☐	☐	☐	sim stats
3	☑	☐	☐	☑	speeds
4	☑	☐	☐	☑	Mach, VVI, G-load
5	☑	☐	☐	☑	atmosphere: weather
6	☐	☐	☐	☐	atmosphere: aircraft
7	☐	☐	☐	☐	system pressures
8	☐	☐	☐	☐	joystick ail/elv/rud
9	☐	☐	☐	☐	other flight controls
10	☑	☐	☐	☑	art stab ail/elv/rud
11	☐	☐	☐	☐	flight con ail/elv/rud
12	☐	☐	☐	☐	wing sweep/thrust vect
13	☐	☐	☐	☐	trim/flap/slat/s-brakes
14	☐	☐	☐	☐	gear/brakes
15	☐	☐	☐	☐	angular moments
16	☑	☐	☐	☑	angular velocities
17	☑	☐	☐	☑	pitch, roll, headings
18	☐	☐	☐	☐	AoA, side-slip, paths
19	☐	☐	☐	☐	mag compass
20	☑	☐	☐	☑	lat, lon, altitude
21	☑	☐	☐	☑	loc, vel, dist traveled
22	☐	☐	☐	☐	all planes: lat
23	☐	☐	☐	☐	all planes: lon
24	☐	☐	☐	☐	all planes: alt
25	☑	☐	☐	☑	throttle command
26	☑	☐	☐	☑	throttle actual
27	☐	☐	☐	☐	feathr-norm-beta-revers
28	☐	☐	☐	☐	prop setting
29	☐	☐	☐	☐	mixture setting
30	☐	☐	☐	☐	carb heat setting
31	☐	☐	☐	☐	cowl flap setting
32	☐	☐	☐	☐	magneto setting

Figure D.18: Data Input & Output - Data Set - Selected Parameters for Data Transfer

Also we use the same aforementioned parameters with '0 frame rate' to be selected to be shown on the cockpit during flight.(Figure : D.18)

The UDP transfer rate of data is selected to be 40.0 Hz.(Figure : D.19)


UDP rate 4 0.0 /sec

disk rate 1 0.0 /sec

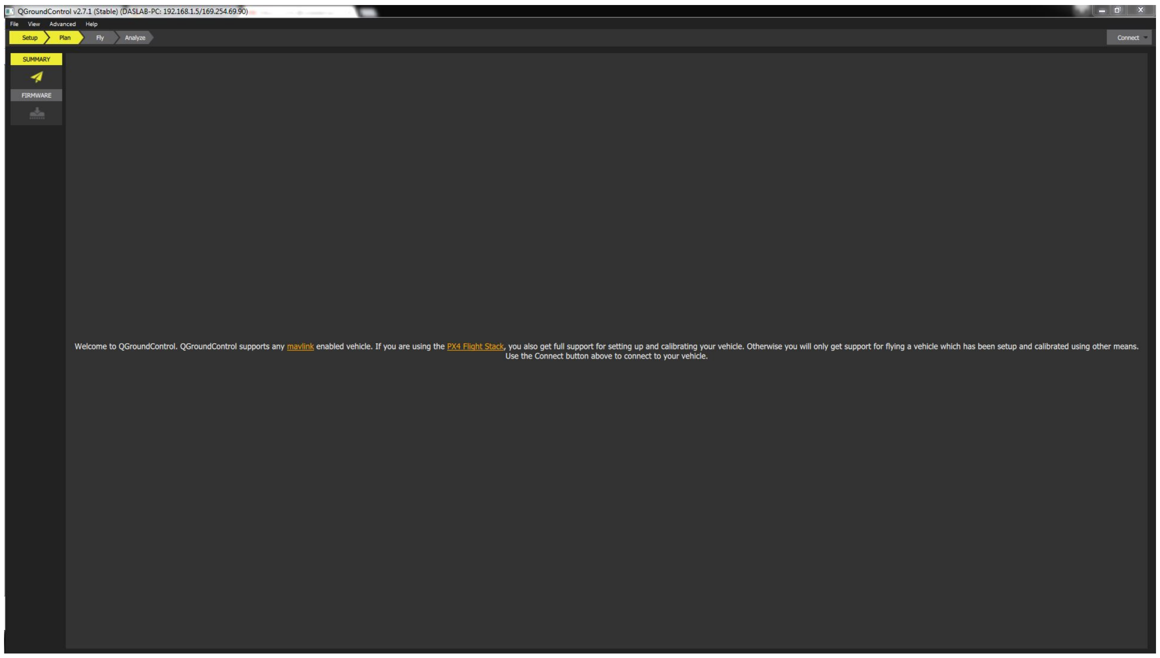
Figure D.19: Data Input & Output - Data Set - UDP Transfer Rate

D.3.6 QGROUNDCONTROL Ground Control Station

QGROUNDCONTROL is an open source Micro Air Vehicle Ground Control Station/Operation Unit.This can be downloaded at `http://qgroundcontrol.org/downloads`, where you find the installation file for QGROUNDCONTROL Stable Build v2.7.1.
The Main Features of QGROUNDCONTROL include

- In-flight manipulation of waypoints and onboard parameters
- 2/3D aerial maps(Google Earth support) with drag-and-drop waypoints
- Real-time plotting of sensors and telemetry data
- Support for UDP, serial(radio modem)and mesh networks
- Logging and plotting of sensor logs
- Support for Head-up-display and digital video transmission/display

After installing the QGROUNDCONTROL, when the program is initialized the following screen is shown.(Figure : D.20)


QGroundControl v2.7.1 (Stable) (DASLAB-PC: 192.168.1.5/169.254.69.90)
File
View
Advanced
Help
Setup
Plan
Fly
Analyze
Connect
SUMMARY
FIRMWARE

Welcome to QGroundControl. QGroundControl supports any mavlink enabled vehicle. If you are using the PX4 Flight Stack, you also get full support for setting up and calibrating your vehicle. Otherwise you will only get support for flying a vehicle which has been setup and calibrated using other means.
Use the Connect button above to connect to your vehicle.

Figure D.20: Start Up window of QGROUNDCONTROL v2.7.1

QGROUNDCONTROL can communicate with STABILIS using the **Serial** and **UDP** protocols .

Communication Link with Serial Protocol

- Goto File in the Menu Bar and Select **Manage Communication Links** shown.(Figure : D.21)

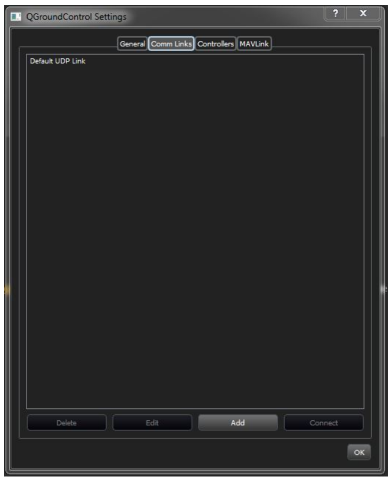
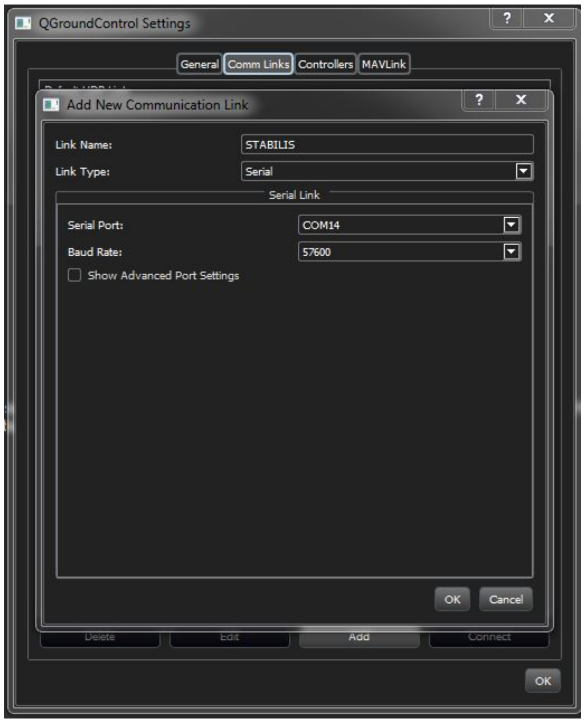


Figure D.21: Manage Communication Links Window in QGROUNDCONTROL

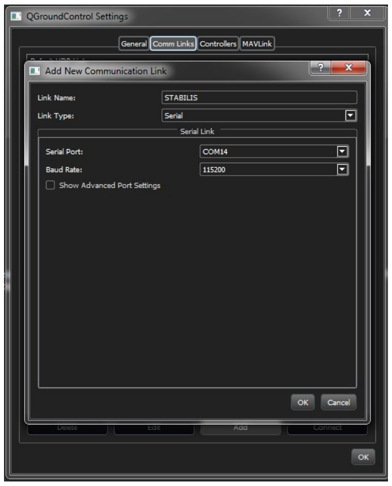
- Select the **Comms Link**
- Select the **Add** button and the following screen pops out.(Figure : D.22)



Link Name: STABILIS
Link Type: Serial
Serial Port: COM14
Baud Rate: 57600
Show Advanced Port Settings

Figure D.22: Add New Communication Link Window in QGROUNDCONTROL

- Add a link name and select **Serial** option from the Link Type drop down menu, then the following screen pops up
- Enter the Serial Port in which the Antenna is connected on the Ground Station Computer and Enter the baud rate as **115200** and click OK. (Figure : D.23)



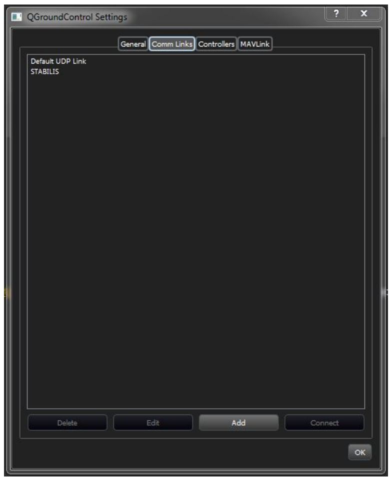
Link Name: STABILIS
Link Type: Serial
Serial Link
Serial Port: COM14
Baud Rate: 115200
☐ Show Advanced Port Settings

Figure D.23: Add New Communication Link Window in QGROUNDCONTROL

- Click OK

Communication Link with UDP Protocol

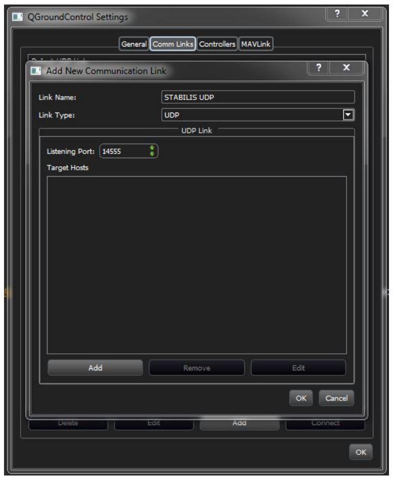
- Goto File in the Menu Bar and Select **Manage Communication Links**(Figure : D.24)



Default UDP Link
STABILIS

Figure D.24: Manage Communication Links Window in QGROUNDCONTROL

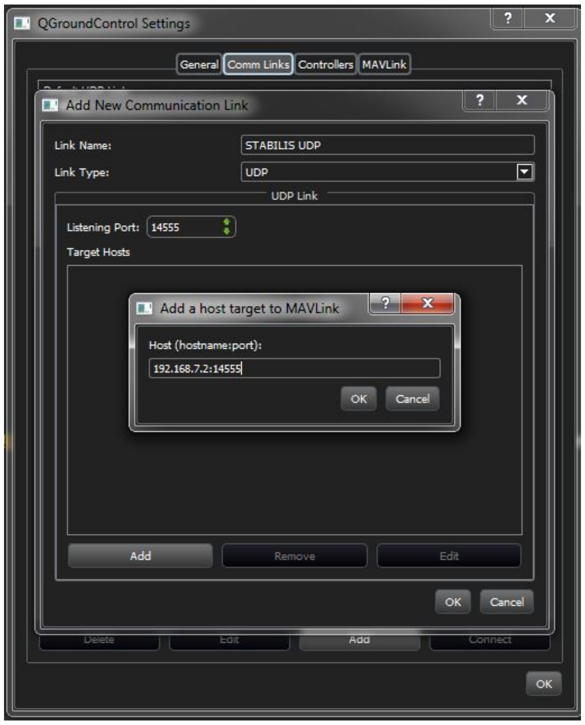
- Select the **Comms Link**
- Select the **Add** button and the following screen pops out(Figure : D.25)



Link Name: STABILIS UDP
Link Type: UDP
UDP Link
Listening Port: 14555
Target Hosts

Figure D.25: Add New Communication Link Window in QGROUNDCONTROL

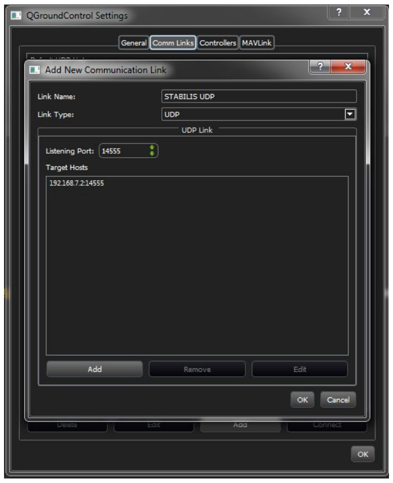
- Add a link name and select **UDP** option from the Link Type drop down menu, then the following screen pops up
- Enter **14555** as the Listening Port and click on **Add** and the following screen pops up.(Figure : D.26)



Link Name: STABILIS UDP
Link Type: UDP
UDP Link
Listening Port: 14555
Target Hosts
Host (hostname:port):
192.168.7.2:14555

Figure D.26: Add New Communication Link Window in QGROUNDCONTROL

- Enter **192.168.7.2:14555** as the Host and click OK(Figure : D.27)



Link Name: STABILIS UDP
Link Type: UDP
Listening Port: 14555
Target Hosts
192.168.7.2:14555

Figure D.27: Add New Communication Link Window in QGROUNDCONTROL

- click OK
- click OK

APPENDIX E

Hardware in the Loop Testing

After setting up all the softwares required , we proceed on to show how to perform the Hardware in the Loop Testing.

E.1 Precautions

- Disconnect the propulsion motors from power supply to avoid accidents
- Make sure the Transmitter and the Controller are paired.
- Make sure the Telemetry radios are paired.
- Switch on the Remote Controller before you run the Stabilis program

E.2 Procedure

- Plug in power on STABILIS
- Plug in the mini USB on STABILIS to the Computer running the Simulation Software(X-Plane)
- Wait until the pop-up menu to show that a Removable Drive is inserted(Figure : E.1)

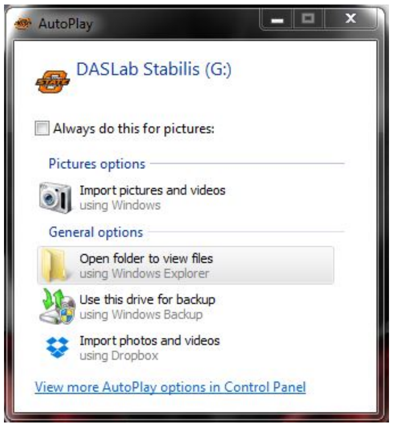
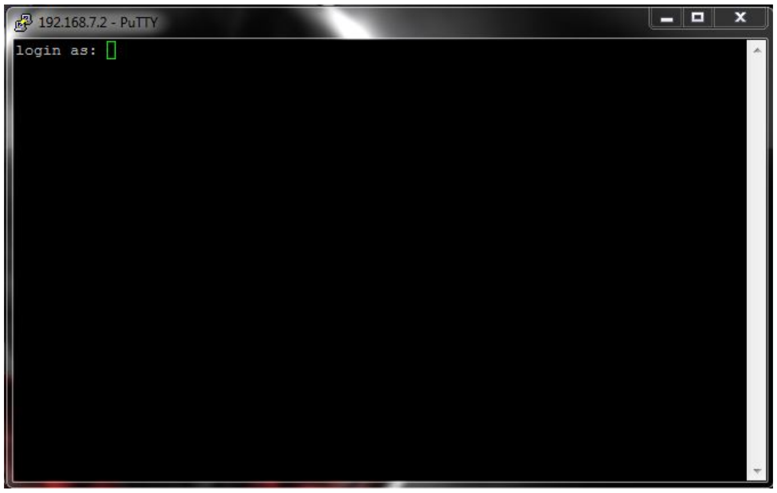


Figure E.1: Autoplay Menu for Removable Device

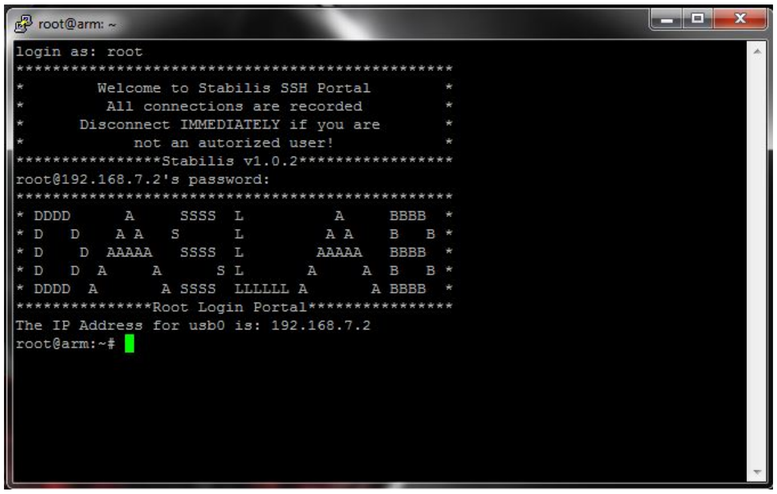
- Open `PuTTY`, ssh into STABILIS.(Figure : E.2)



```
login as:
```

Figure E.2: Terminal of STABILIS

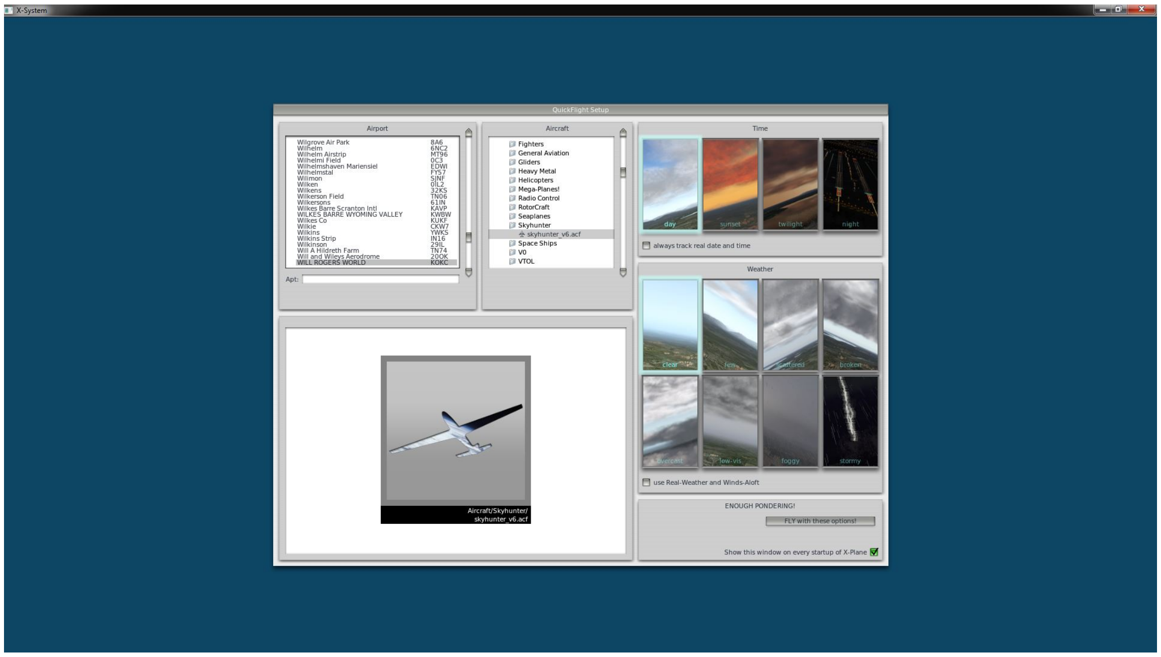
- Enter the login credentials. Login : root and Password : root(Figure : E.3)



```
login as: root
**************************************************
*        Welcome to Stabilis SSH Portal          *
*         All connections are recorded           *
*      Disconnect IMMEDIATELY if you are         *
*            not an autorized user!              *
****************Stabilis v1.0.2******************
root@192.168.7.2's password:
**************************************************
* DDDD      A     SSSS  L          A     BBBB  *
* D   D    A A   S      L         A A    B   B *
* D    D  AAAAA   SSSS  L        AAAAA   BBBB  *
* D   D  A     A      S L       A     A  B   B *
* DDDD  A       A SSSS  LLLLLL A       A BBBB  *
***************Root Login Portal*****************
The IP Address for usb0 is: 192.168.7.2
root@arm:~#
```

Figure E.3: Logging into STABILIS securely

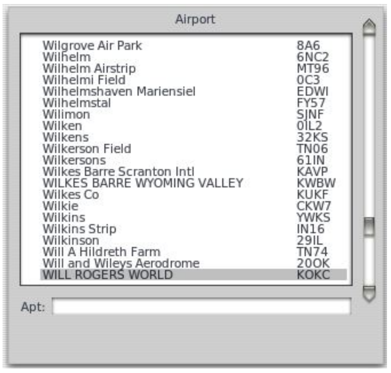
- Open X-Plane, the following shows up, upon loading(Figure : E.4)



Wilgrove Air Park	8A6
Wilhelm	6NC2
Wilhelm Airstrip	MT96
Wilhelmi Field	0C3
Wilhelmshaven Mariensiel	EDWI
Wilhelmstal	FY57
Wilimon	SJNF
Wilken	0IL2
Wilkens	32KS
Wilkerson Field	TN06
Wilkersons	61IN
Wilkes Barre Scranton Intl	KAVP
WILKES BARRE WYOMING VALLEY	KWBW
Wilkes Co	KUKF
Wilkie	CKW7
Wilkins	YWKS
Wilkins Strip	IN16
Wilkinson	29IL
Will A Hildreth Farm	TN74
Will and Wileys Aerodrome	20OK
WILL ROGERS WORLD	KOKC

Figure E.4: Start Up screen of X-Plane

- Select the airport from which you want to fly.(Figure : E.5)



Wilgrove Air Park	8A6
Wilhelm	6NC2
Wilhelm Airstrip	MT96
Wilhelmi Field	0C3
Wilhelmshaven Mariensiel	EDWI
Wilhelmstal	FY57
Wilimon	SJNF
Wilken	0IL2
Wilkens	32KS
Wilkerson Field	TN06
Wilkersons	61IN
Wilkes Barre Scranton Intl	KAVP
WILKES BARRE WYOMING VALLEY	KWBW
Wilkes Co	KUKF
Wilkie	CKW7
Wilkins	YWKS
Wilkins Strip	IN16
Wilkinson	29IL
Will A Hildreth Farm	TN74
Will and Wileys Aerodrome	20OK
WILL ROGERS WORLD	KOKC

Figure E.5: Selection of Airport

- Select the aircraft that you want to fly with.(Figure : E.6)

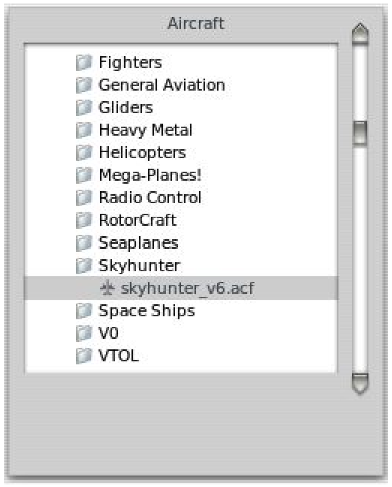


Figure E.6: Selection of Aircraft

- Then select the weather conditions you want to fly.It can be either real time data at the selected location or predefined constant weather conditions.(Figure : E.7)

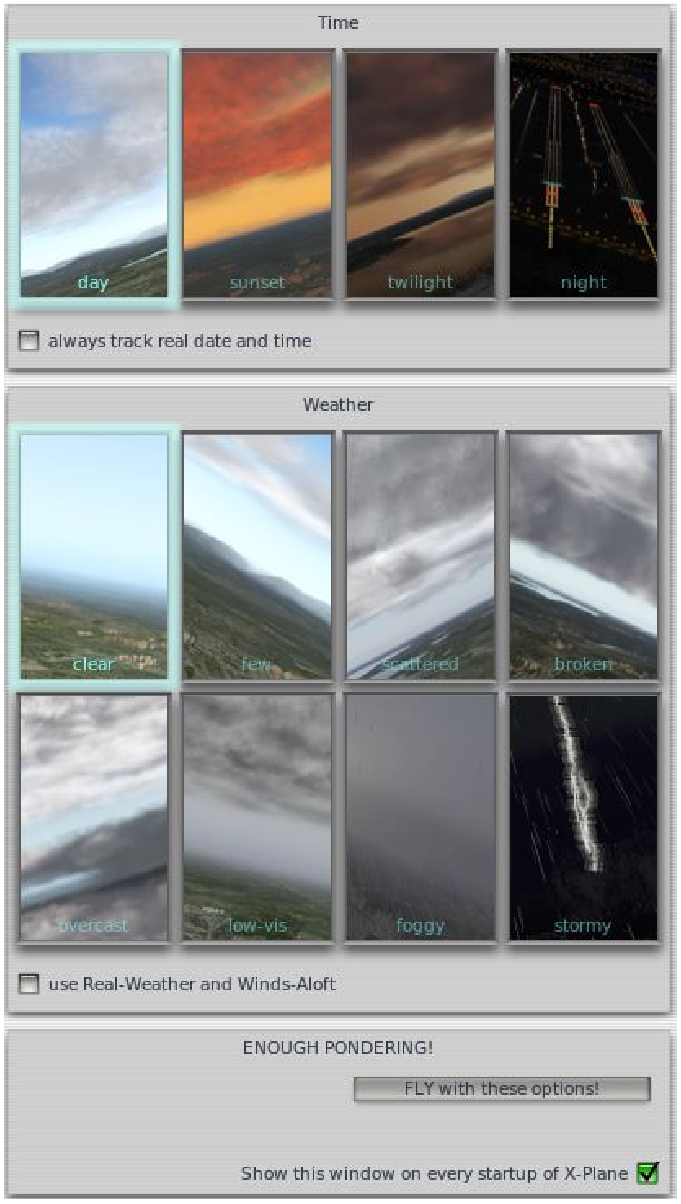


Figure E.7: Selection of the Date, Time and Weather

- Open QGROUNDCONTROL, Goto File then to Manage Communications Links. The following window pops up. Select the protocol through which you wan to connect to STABILIS(either Serial or UDP)
- Goto Advanced option on the Menu bar and select HIL Simulation. Or you can use the shortcut Ctrl + 5 (QGC v2.7.1 only).
- Switch to `PuTTY` terminal and go to Stabilis folder and Run ./Stabilis v1.0.2 executable

- Hit Connect with the select Communication Link on QGROUNDCONTROL(Serial or UDP)
- Then you can see the Aircraft at the selected Location on the Map in QGROUNDCONTROL.
- Select the Home Location at the Aircraft position by right clicking on the Map in QGC.
- There are two ways to load the Waypoints on QGC
- Hit Edit Waypoints
- The First way is to select the Waypoints on the map by double-clicking the location on the map and assigning the parameters associated with the waypoints.
- The second way is to load a pre-defined Waypoint list
- Then hit set in the lower right corner of the QGC window

Bibliography

[1] K. S. Narendra, Y.-H. Lin, and L. S. Valavani, "Stable adaptive controller design, part ii: Proof of stability," *Automatic Control, IEEE Transactions on*, vol. 25, no. 3, pp. 440–448, 1980.

[2] K. S. Narendra and A. M. Annaswamy, *Stable adaptive systems.* Courier Corporation, 2012.

[3] R. W. Beard, N. B. Knoebel, C. Cao, N. Hovakimyan, and J. S. Matthews, "An l1 adaptive pitch controller for miniature air vehicles," in *AIAA Guidance, Navigation, and Control Conference, Keystone, CO*, 2006.

[4] C. Cao, N. Hovakimyan, and E. Lavretsky, "Application of l1 adaptive controller to wing rock," in *AIAA Guidance, Navigation, and Control Conference, Keystone, CO*, 2006.

[5] C. Cao and N. Hovakimyan, "Design and analysis of a novel l1 adaptive control architecture with guaranteed transient performance," *IEEE Transactions on Automatic Control*, vol. 53, no. 2, pp. 586–591, 2008.

[6] C. Cao and N. Hovakimyan, "L1 adaptive controller for systems with unknown time-varying parameters and disturbances in the presence of non-zero initialization error," *International Journal of Control*, vol. 81, no. 7, pp. 1147–1161, 2008.

[7] I. Kaminer, A. Pascoal, E. Xargay, N. Hovakimyan, C. Cao, and V. Dobrokhodov, "Path following for small unmanned aerial vehicles using l1 adaptive augmentation of commercial autopilots," *Journal of guidance, control, and dynamics*, vol. 33, no. 2, pp. 550–564, 2010.

[8] C. Cao and N. Hovakimyan, "Vision-based aerial tracking using intelligent excitation," in *American Control Conference, 2005. Proceedings of the 2005*, pp. 5091–5096, IEEE, 2005.

[9] C. Cao, N. Hovakimyan, and J. Wang, "Intelligent excitation for adaptive control with unknown parameters in reference input," *Automatic Control, IEEE Transactions on*, vol. 52, no. 8, pp. 1525–1532, 2007.

[10] T. Yucelen and A. J. Calise, "Derivative-free model reference adaptive control of a generic transport model," in *AIAA Guidance, Navigation, and Control Conference, Toronto, ON*, 2010.

[11] T. Yucelen and A. J. Calise, "Derivative-free model reference adaptive control," *Journal of Guidance, Control, and Dynamics*, vol. 34, no. 4, pp. 933–950, 2011.

[12] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *Neural Networks, IEEE Transactions on*, vol. 1, no. 1, pp. 4–27, 1990.

[13] F. Lewis, "Nonlinear network structures for feedback control," *Asian Journal of Control*, vol. 1, no. 4, pp. 205–228, 1999.

[14] A. J. Calise and R. T. Rysdyk, "Nonlinear adaptive flight control using neural networks," *Control Systems, IEEE*, vol. 18, no. 6, pp. 14–25, 1998.

[15] B. S. Kim and A. J. Calise, "Nonlinear flight control using neural networks," *Journal of Guidance, Control, and Dynamics*, vol. 20, no. 1, pp. 26–33, 1997.

[16] A. J. Calise, N. Hovakimyan, and M. Idan, "Adaptive output feedback control of nonlinear systems using neural networks," *Automatica*, vol. 37, no. 8, pp. 1201–1211, 2001.

[17] E. N. Johnson, M. A. Turbe, A. D. Wu, S. K. Kannan, and J. C. Neidhoefer, "Flight test results of autonomous fixed-wing uav transitions to and from stationary hover," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference Exhibit, Monterey, CO*, 2006.

[18] G. Chowdhary and E. Johnson, "Adaptive neural network flight control using both current and recorded data," in *AIAA Guidance, Navigation, and Control Conference, AIAA-2007-6505. Hilton Head*, 2007.

[19] G. Chowdhary, T. Wu, M. Cutler, and J. P. How, "Rapid transfer of controllers between uavs using learning-based adaptive control," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pp. 5409–5416, IEEE, 2013.

[20] J. Park and I. W. Sandberg, "Universal approximation using radial-basis-function networks," *Neural computation*, vol. 3, no. 2, pp. 246–257, 1991.

[21] G. Chowdhary, H. Kingravi, J. P. How, and P. A. Vela, "Bayesian nonparametric adaptive control using gaussian processes," *IEEE Transactions on Neural Networks*, 2013 (submitted).

[22] T. Kanade and M. Okutomi, "A stereo matching algorithm with an adaptive window: Theory and experiment," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 16, no. 9, pp. 920–932, 1994.

[23] L. Csató and M. Opper, "Sparse on-line gaussian processes," *Neural computation*, vol. 14, no. 3, pp. 641–668, 2002.

[24] R. Grande, G. Chowdhary, and J. How, "Experimental validation of bayesian nonparametric adaptive control using gaussian processes," *Journal of Aerospace Information Systems*, 2013 (Submitted).

[25] R. W. Beard and T. W. McLain, *Small Unmanned Aircraft Theory and Practice.* Princeton: Princeton University Press, 2012.

[26] R. Nelson, *Flight stability and automatic control.* Boston, Mass: WCB/McGraw Hill, 1998.

[27] E. Lavretsky and K. Wise, *Robust and Adaptive Control: With Aerospace Applications.* Springer, 2012.

[28] G. Cai, B. M. Chen, and T. H. Lee, *Unmanned rotorcraft systems.* New York: Springer, 2011.

[29] S. Boyd and S. S. Sastry, "Necessary and sufficient conditions for parameter convergence in adaptive control," *Automatica*, vol. 22, no. 6, pp. 629–639, 1986.

[30] R. M. Sanner and J.-J. Slotine, "Gaussian networks for direct adaptive control," *Neural Networks, IEEE Transactions on*, vol. 3, no. 6, pp. 837–863, 1992.

[31] K. S. Narendra and A. M. Annaswamy, "A new adaptive law for robust adaptation without persistent excitation," *Automatic Control, IEEE Transactions on*, vol. 32, no. 2, pp. 134–145, 1987.

[32] G. Chowdhary, J. How, and H. Kingravi, "Model reference adaptive control using nonparametric adaptive elements," in *Conference on Guidance Navigation and Control, Minneapolis, MN*, 2012.

[33] C. E. Rasmussen, *Gaussian processes for machine learning.* MIT Press, 2006.

[34] J. Stockton, "Modular autopilot design and development featuring bayesian non-parametric adaptive control," Master's thesis, Oklahoma State University, 12 2014.

VITA

Sri Theja Vuppala

Candidate for the Degree of

MASTER OF SCIENCE

Thesis: IMPLEMENTATION AND VALIDATION OF GAUSSIAN PROCESS MODEL REFERENCE ADAPTIVE CONTROL FOR FIXED WING UNMANNED AERIAL SYSTEMS

Major Field: Mechanical and Aerospace Engineering

Biographical:

Education: Completed the requirements for the Master's of Science degree with a major in Mechanical and Aerospace Engineering at Oklahoma State University in April, 2016. Completed requirements for Bachelors of Technology degree with a major in Mechanical Engineering at Jawaharlal Nehru Technological University, Kakinada in May, 2012.

Experience:

Research Assistant for DasLab, at Oklahoma State University, Stillwater, OK: Spring 2014 to May 2016.

Teaching Assistant for Measurements and Instrumentation (MAE 3113), at Oklahoma State University, Stillwater, OK: Fall 2014

Teaching Assistant for Systems - I (MAE 3723), at Oklahoma State University, Stillwater, OK: Spring 2015 and Fall 2015