OVERVIEW

We intend to build a quadcopter that can navigate autonomously and create and use 3D maps to accomplish tasks from a flight planner. The quadcopter will use a variety of sensors including gyroscopes, accelerometers, barometers, and magnetometers to stabilize itself and navigate. This robot will be controlled through a flight planner on an android enabled device, which allows users to track and give commands to the quadcopter. The vehicle will create 3D maps with a vision system using a customized SLAM algorithm to accomplish these tasks.

QUADCOPTER DESIGN

A quadcopter consists of four basic parts: the frame, arms, motor brackets, and skid. The design goal is to utilize the most efficient manufacturing processes and materials. Manufacturing methods (3D printing, CNC machining, water jet cutting, etc.) and materials (G-10, aluminum, ABS, etc.) will be researched and tested. Solidworks will be used to design and analyze the model. Structural analysis can be done through Solidworks for additional testing.

The major components are brushless motors, propellers, electronic speed controllers, and batteries. A general rule is to choose motors, ESCs, and propellers that provides two times more thrust than all up weight (AUW). By always having more than double the necessary thrust, the vehicle is able to accelerate more rapidly and maintain precise control. Dozens of specifications affect performance such as motor Kv, amperage, propeller pitch, diameter, etc. A variety of combinations will be researched to find one which optimizes thrust while reducing cost and mass.

The flight controller is another integral component which incorporates a microprocessor and sensors to stabilize the vehicle and provide accurate control. Many premade flight controllers are available and range from $50 to $300. To minimize cost, we will design and construct our own using the Arduino UNO, 9DOF IMU, and Barometric Pressure sensor. Eagle is a free PCB design software which we will use for this. The final design goal is to create a modularized system that incorporates all the necessary sensors and electronics and can be used on multiple platforms, such as quadcopters, underwater ROVs, and UGVs.

Another major goal is to efficiently design the electronics. A focus of this will be flight time. On average, flight times range from four minutes for nano-quadcopters to fifteen minutes for mid-sized quadcopters. With such short flight times, the applications of quadcopters are crippled. Optimizing the battery, mass, and other characteristics of the quadcopter can increase flight time. Another focus will be incorporating principles of the Taguchi method, also called the robust design method. The Taguchi method improves the fundamental function of a product by designing with noise and cost in mind. Electric noise, which is primarily caused by motor vibrations, skews sensor values.

AUTONOMOUS FLIGHT

The main function of the quadcopter is to autonomously stabilize itself and execute commands and tasks that are sent to it. These task can range from a very simple task (i.e. moving forward 6 ft.) to a very complex tasks that integrates computer vision and other features of the quadcopter. The basic autonomous system would self-stabilize and hover. Commands from the flight planner would be processed on an onboard Arduino UNO Rev3 microcontroller. This microcontroller is directly connected to the motors and onboard sensors. The Arduino UNO will be responsible for sensor integration to track the movement and location of the robot. The quadcopter will have an onboard Sparkfun 9DOF IMU breakout sensor stick, this sensor stick includes the L3DG20 gyroscope, the

LSM303DLHC accelerometer compass, and the BMP180 barometric pressure sensor. These 3 sensors will be integrated together by the Arduino microcontroller to track the location of the quadcopter. A problem associated with sensor integration is the massive amount of drift that can be caused by inaccurate gyroscope and accelerometers reading. One of the ways we plan to combat this inaccuracy is by building a mathematical model of the physics that dictate the movement of the quadcopter. The sensor values will then be cross referenced with the mathematical model of how the quadcopter moves in three dimensional space, and if these values do not match regular movement of the robot, the sensors will be recalibrated to remove their inaccuracies. Integrating these sensors on the onboard Arduino will streamline the process of stabilizing the quadcopter midflight. The microcontroller will be directly wired to the ESCs and motors to allow the least amount of delay between the movement of the robot and the response of the motors. The onboard Arduino UNO microcontroller will also receive and calculate movement commands received by the onboard Raspberry Pi's or through the wireless signal. The architecture of the code onboard the microcontroller will be structured to receive a command in the form of a string or byte of data. This command will be parsed and translated into simplified maneuvers, which will then be further broken down into thrust output for each motor. This will be handled by a PID control loop (see PID and Machine Learning) for each of the four motors controlling the quad copter. This function of translating user commands into usable maneuvers is the essential backbone of our autonomous flight system. By handling the conversion of high level command to low level robotic action on a PC/Android client off board, we can lighten the load on the Arduino performing PID stabilization. This system will allow the quadcopter to be programmed for many versatile tasks by altering the off board client, without changing the software onboard the robot. This allows the quadcopter to be tailored to many different applications without major adjustments to the vehicle.

CONTROL SYSTEM

Our quadcopter will be commanded by an android app. This app handles all of the calculations for autonomous flight, allows the user to control the quadcopter in real-time, and allows the user to see real time statistics of the quadcopter. For us to be able to connect and transfer data from the android device to the quadcopter, we will build a transponder that connects the two devices. By bridging an XBee Pro 60mW Wire Antenna – Series 1 (802.15.4) with a Bluetooth SMD Module – RN-42 receiver we can directly send data via Bluetooth and it will be transferred to the on board Arduino microcontroller within the quadcopter where it will be parsed and turned into usable data. There will be an XBee Pro 60mW Wire Antenna – Series 1 (802.15.4) connected to the onboard Arduino UNO Rev3 (A000066) microcontroller to receive the data sent by the transponder. By using a Bluetooth receiver connected to the XBee Antenna, the user can easily pair their android device to the Bluetooth receiver which makes the connection process more reliable and user friendly. The app itself will have a simplistic design, but allow the user the ability to do many powerful things. By setting up the code architecture within the program to be very flexible and user friendly, it will allow many programmers to tailor the app to their liking. We plan on releasing a fully published API to allow anyone the ability to alter our code to perform specific artificial intelligence commands and processes. This open structure and access to documentation will let the user insert their own custom intelligent agents which allows our robot to be used for a user's specific task.

COMPUTER VISION

One of our proposed navigation systems for the quadcopter robot is a stereo computer vision system. We believe that having onboard live computer vision systems will give the robot much more robust

navigation and decision making. The ability to perceive and successfully make sense of 3D spatial environments is what we believe to be an integral part of any fully autonomous aircraft. We also intend to implement processes that will begin to map in 3D space the environment in which the robot is placed. The level of detail that will be achieved in said 3D maps still remains an uncertainty. Other vision based mapping systems typically implement a simple 3D point cloud approach, which for the most part is all that is really desired. We however plan to initially investigate point clouds, but hopefully we can move to having maps which include information about specific spatial and pattern information unique to certain environment, as well objects detected within the scene. A popular method of robotic navigation is SLAM (Simultaneous Localization and Mapping) which is the process of a robot creating a map of its environment whilst also trying to localize itself in it. We intend to implement our own version of SLAM navigation using the input from the cameras as well as knowledge of the robot's own odometry and most likely some form of Kalman filtering to have our robot localize itself in its environment. We plan to implement methods of approximating the robot's flight path in 3D space to determine collisions and correct the path for successful object avoidance. We plan to use SIFT or SURF feature detection algorithms for object identification. Features can be stored and associated with certain environments. To calculate depth of objects in view we plan to calculate the stereo disparity of the images received from the two cameras, and then use this to derive a 3D point cloud of the scene. Many of the aforementioned computer vision algorithms will be computationally very expensive, so we will need to investigate how to split the calculation task between the on board Raspberry Pi's and an Android/PC client connected via XBee wireless.

PID AND MACHINE LEARNING

One of the big software obstacles in this project is minimizing latency between the onset between environmental stimuli and robot reaction. Ultimately the stabilization algorithm should be able to adapt over time with the goal of becoming more efficient. The main application of this adaptive algorithm would be for real time stabilization of the robot, as well as performing flight maneuvers. In theory the robot should with every successive iteration of the algorithm get more efficient and accurate with its adjustments. Now there are three distinct steps that need to be taken to successfully implement this control system.

First, the quad copter must know how to perform specific maneuvers to inevitably correct for errors in its desired path. To do this the robot must adjust the yaw, pitch, roll, and individual thrust output of each rotor to, in theory, translate the robot's position to any surrounding point in 3D space. To get to point where the robot would be able to do this we plan to combine physics calculations with elements of machine learning so that through trial and error the robot could "learn" what actions did what.

Second, the quad copter must be able to interpret sensor data to achieve accurate estimates of movement. This step must work in tandem with the first since the more confident the system is making the correct discretized movement, the more accurate a sensor reading correlated to said movement will be. The system should be able to take data from the accelerometer, gyroscope, compass, and potentially visual SLAM to accurately gauge 3D movement through space. We believe another machine learning process would be needed in this step, but this time it would be for pattern recognition so that the system could correlate movements to sensor data. We propose using a separate neural network for each possible desired movement (ex: up, down, left, right, etc.) so that over time with the input of sensor data and known action, they can successfully match sensor reading to movements for real time motion tracking.

Third, combining both of the previous steps of measuring and attempting movement we will need to implement a PID control system to make real time adjustments in stabilization and movement. The principal of a PID control system is that there is a desired value that the system must attempt to achieve. In the context of this project the desired value would be either maintaining upright flight or making a desired movement. Essentially what the PID does is subtract the measured value from the desired to determine error and use the error as a function as to how much the system should adjust itself. A large error would yield a large correction and a small error would yield a small correction. Now for this the measured value would be the calculation from step two and the correction would the action taken from step one. The desired value would come from a processed.

MECHANICAL OJECTIVES

- Optimize mechanical design to increase flight performance
- Research manufacturing techniques and materials to construct quadcopter
- Increase flight time
- Prepare for and earn CSWP
- Learn Eagle to design PCBs for the flight control board
- Create a modularized system which can be adapted to multiple vehicle platforms

ANDROID AND AUTONOMOUS FLIGHT OBJECTIVES

- Write and compile a working Android app that can connect to wireless Bluetooth devices
- Build a bridge between a Bluetooth and Wireless connection to transfer flight data
- Write software to guide and navigate the quadcopter through different environments
- Create a system to path find in three dimensions using the A* (Astar) method
- Release a fully published API that developers can access and program intelligent agents to work with our quadcopter

INTEGRATED CONTROL OBJECTIVES

- Design adaptive PID stabilization algorithm for attitude control
- Create spatial awareness system for indoors localization (SLAM)
- Make a custom flight control board with implementation for IMU, sonar, and camera.
- Complete Autonomous Navigation for Flying Robots course on EDX provided via Technische Universität München.
- Investigate visual odometry and compare it too sonar/IMU odometry.
- Learn 3D geometric matrix transformations.
- Design system for post-flight diagnostics and simulated memory consolidation for machine learning.
- Investigate integrated image compression