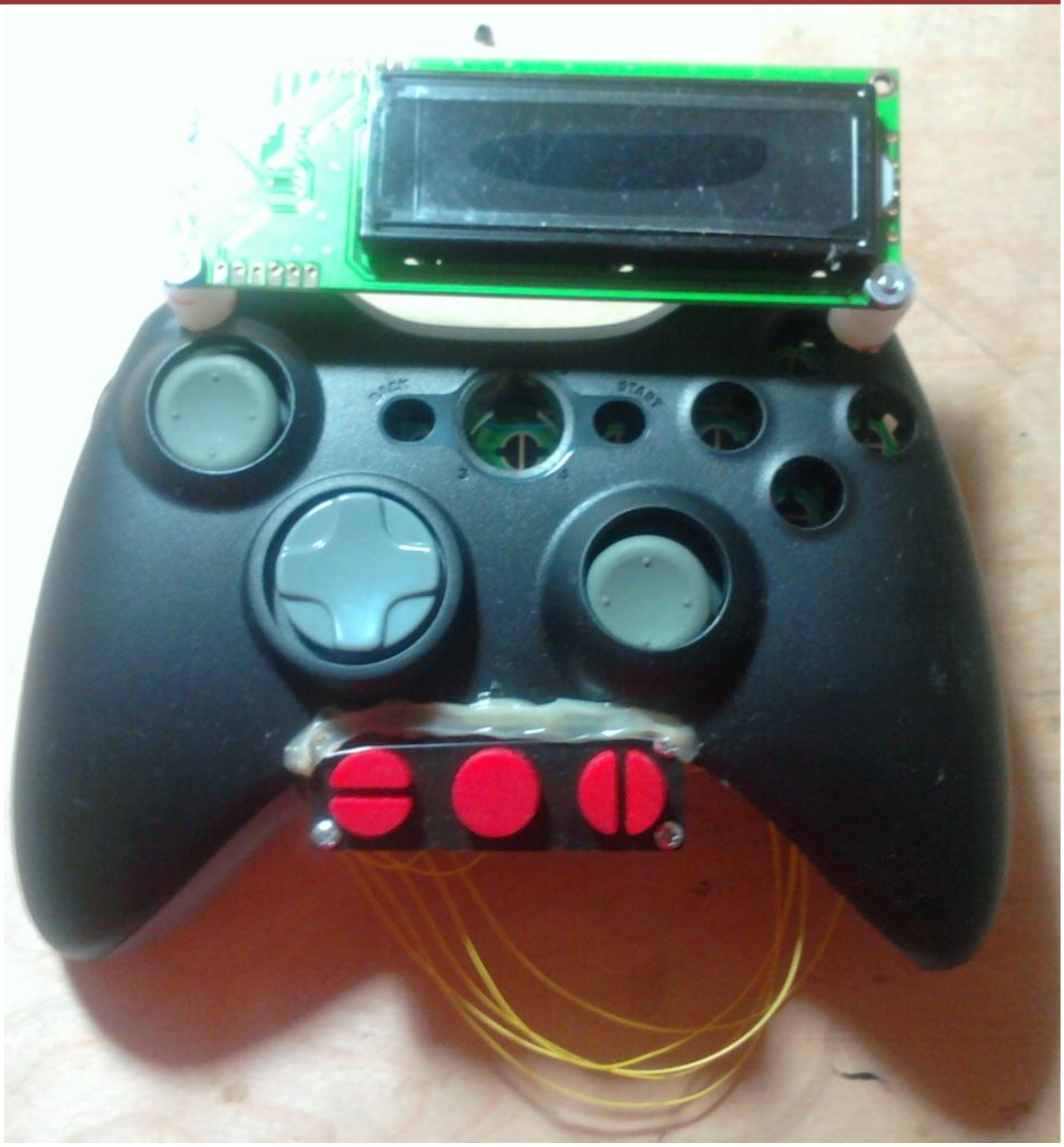
2010



David Gitz, EE
Michael Welling, EE

REMOTE CONTROL UNIT DESIGN REPORT

Contents

Executive Summary

As robotic projects are becoming more prevalent in society, the hobbyist community devoted to developing these projects is increasing at a large rate. Manual control of these systems for operational levels and debugging is necessary, but due to the large amount of different systems, architectures and programming, it is difficult to have one device that can work with every system, while being easily programmable and configurable.

The Remote Control Unit (RCU) has been developed to overcome these limitations. By providing a user with a simple, configurable and capable hand-held controller, manual and limited automatic control of a robotic vehicle is achieved.

Overview

The Remote Control Unit (RCU) is an open-source programmable device that allows a user with minimal knowledge of the programming language (SPIN) to implement remote control of their project. Due to the ubiquitous use of the Xbee radio, a common communications interface has in effect become standardized. However, the devices that interfaces between a human and a communications system is not only diverse, they present a barrier of entry into developing more advanced systems.

The RCU is built upon the Propeller Micro-Controller (uC), a unique device that contains 8 processors, called Cogs, that are each capable of being clocked at 80 MHz, and combined allows up to 160 Million Instructions Per Second (MIPS) (20 MIPS per Cog) (Parallax, 2010). Not only does the Propeller allow unprecedented processing power, it delivers that with a fairly easy to use programming language, SPIN, and a large user support system known as the Object Exchange.

The RCU is implemented off an XBox-360 Wireless Controller. Due to the existing Printed Circuit Board (PCB) constraints of the Xbox-360 controller, a custom PCB has been designed and will replace the existing PCB. This has the advantage of reducing existing “stove pipes” that inhibit future design and innovation. However, there exist disadvantages to this as well, as can be viewed in Table 1: Advantages/Disadvantages of using custom PCB. 2 Analog 2-axis joysticks, buttons, external switches and a GPS sensor are used as inputs to the RCU, and a LCD screen, 10-segment LED bar and vibrating motors are used for feedback to the user of the RCU. Also, a programming port has been added to aid in future development.

Table 1: Advantages/Disadvantages of using custom PCB

Advantage of using custom PCB	Disadvantage of using custom PCB
Minimal hardware exposed outside of Controller housing	Can not use with an Xbox 360
Not necessary to examine traces of non-custom PCB to connect to new components	Can not recharge via USB device
New buttons/indicators can be installed	Some buttons/indicators removed

As the RCU has been developed only to support the Robo-Chopper Project (see Figure 1: Robo-Chopper System Diagram) and is not dependent on that System to function. Indeed, any Vehicle is capable of being controlled by the RCU as the software and hardware are completely open-source and efforts have been made to ensure continuing viability.

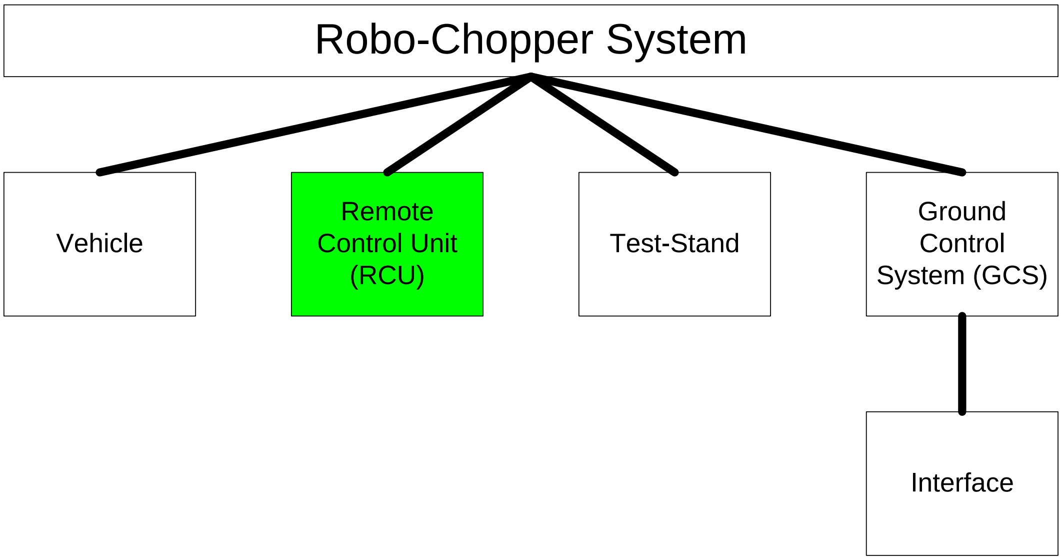


Figure 1: Robo-Chopper System Diagram

Technical Description

Functional Description

Figure 2: Functional Diagram shows the process the RCU follows during operation. Although simplistic, the RCU must be capable of reliable communications and prompting the user when any errors occur to reduce the chances of catastrophic vehicle mishaps. The RCU is an intelligent device, as can be seen in Table 2: RCU Features.

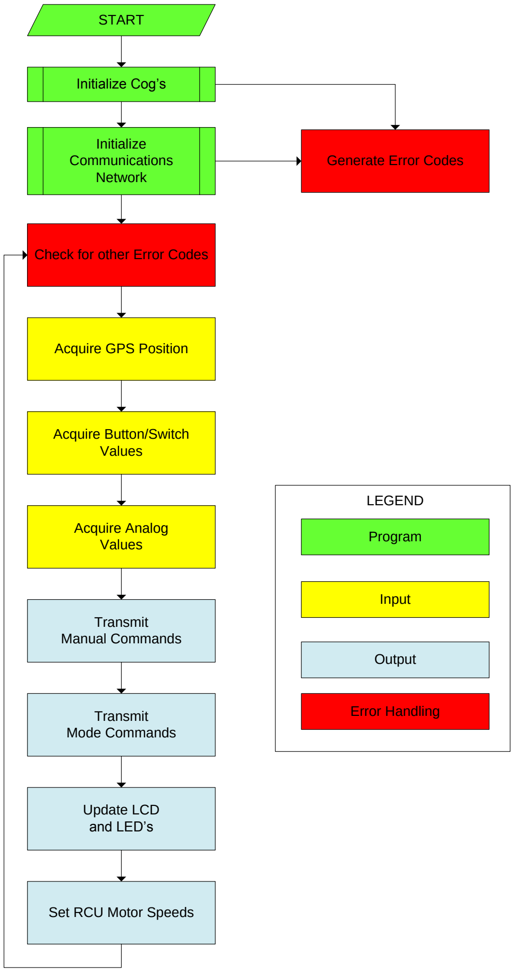


Figure 2: Functional Diagram

Table 2: RCU Features

Feature
Report Error Codes and brief explanation
Visual indicator of communications network status
"Kill Switch" turns off Vehicle motors immediately
Error detection on communications network
Feedback via vibrating motors
Long run-time (4-9 hrs)
Can use API or non-API Xbee network
GPS Sensor to enable Vehicle for autonomous flight back to RCU
Mode selection to give limited autonomous control
Programming port to allow future expansion/development

Physical Description

Figure 3: Physical Diagram illustrates the different subsystems that are involved in the RCU. Together, these subsystems are designed to provide optimum efficiency on as small of a PCB as possible, to permit mounting onto the XBox 360 Controller without diminishing the user's experience, and to allow future expansion and innovation on the RCU. See Appendix 1: Schematics and Appendix 2: Program Listing for more technical information concerning the Hardware and Software of the RCU. See Appendix 6: Datasheets for information regarding individual components.

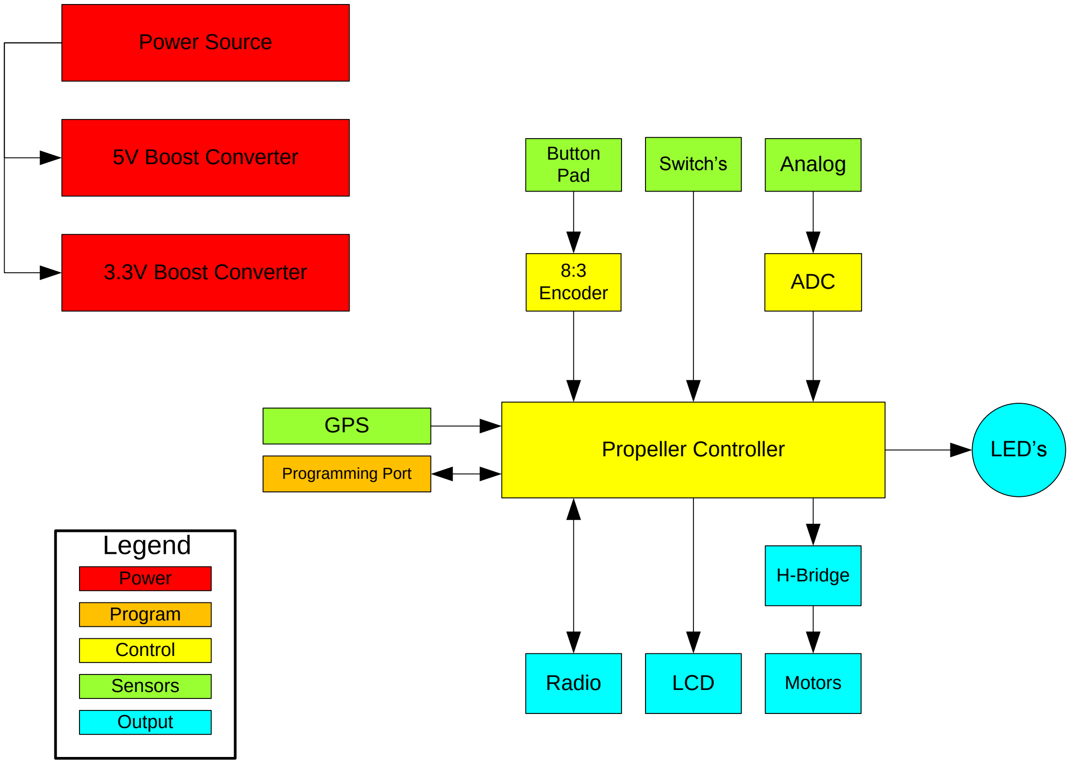


Figure 3: Physical Diagram

Inputs/Outputs:

The Analog inputs of the XBox Controller are provided via 2 2-axis thumb joysticks that are based on potentiometers. For the Digital inputs, a button pad with 8 inputs is used and these lines are connected to an 8:3 priority encoder to reduce the number of pins required on the Propeller. One “kill switch” has been incorporated that is connected directly to the Propeller. This can be accessed programmatically, and will be used in this project to remove power from the Vehicle’s motors to reduce any damage in the event of an in-air mishap.

Power

The Xbox-360 Wireless Controller's rechargeable battery pack is used as a power supply for the RCU. The battery pack is a 2.4 Volt Ni-Mh battery. On the RCU Printed Circuit Board (PCB) are two DC-DC boost converters that provide a 3.3 V and 5 V source for the various components on the PCB. Table 3: Power Requirements illustrates the power consumption of the RCU PCB components. The expected run-time of the RCU is between 4 and 9 hours of continuous use. See Appendix 3: Power Analysis for more information.

Table 3: Power Requirements

Component	***Voltage***	***Current (Worst Case) (mA)***	***Power Dissipated (Worst Case) (mW)***
Propeller	3.3	100	330
Xbee Radio	3.3	215	709.5
24LC256	3.3	3	9.9
10 Segment LED	3.3	33	108.9
GPS	3.3	41	135.3
LCD	5	60	300
MCP3208	3.3	0.4	1.32
74HC148	3.3	50	165

Sensors/Feedback

A 12-Channel GPS Sensor is an easy additional component to the RCU. It allows the extended functionality of allowing another system to view the coordinates of the RCU. In this project, the RCU is able to transmit its location to another node on the network. Also, a 16 character by 2 line LCD display is included, along with 2 LED's. Additionally, 2 motors with offset weights give the ability to vibrate as another method to give feedback to the user.

Control

The brain of the RCU is the Propeller Micro-Controller (uC). Usage of the Propeller uC allows a large level of sophistication while not impeding user development. An absence of a floating point processer, Universal Asynchronous Receiver/Transmitter (UART) and their like provides minimal internal support for extending the functionality of the Propeller uC. However, by using readily available code from the Propeller Object Exchange, the advanced usage of the Propeller uC is available. Indeed, most cores of the RCU Propeller are used and dedicated for different tasks, as can be seen in Table 4: Propeller Cog Usage.

Table 4: Propeller Cog Usage

Cog 1: Main Power	Cog 2: Xbee Radio Driver	Cog 3: Com Protocol Driver	Cog 4: 4-Port UART
Cog 5: ADC	Cog 6: PWM Driver	Cog 7: Unused	Cog 8: Unused

Communications

The XBee radio was selected due to its widespread use. The XBee has two modes of usage, Non-API and API modes. While the Non-API mode is easier to configure, it allows only 2 nodes in a network that are capable of communicating with each other, although there are exceptions to this. The API mode gives the XBee the functionality to change the destination node for each packet, similar to IP Packets, but there is an increased level of complexity. This complexity has been reduced as much as possible on the RCU, but the complexity remains on the other nodes in a system.

Reliable and high throughput are features desired on any communication network, and very much so in this project. If a waypoint is transmitted in error, or a motor speed is sent incorrectly, the Vehicle could have an in-air mishap. These errors should be reduced as much as possible but still allowing high throughput. The Transfer Connection Protocol (TCP)/Internet Protocol (IP) 3-way handshake design has been utilized in this project when initiating a communications link with a node in the system, as can be referenced in Figure 4: 3-Way Handshake.

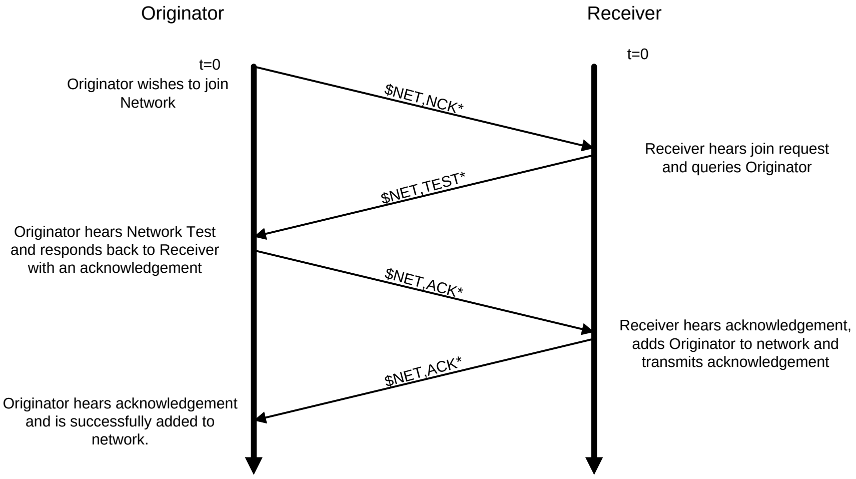

Originator
Receiver
t=0
Originator wishes to join Network
t=0
$NET,NCK*
Receiver hears join request and queries Originator
$NET,TEST*
Originator hears Network Test and responds back to Receiver with an acknowledgement
$NET,ACK*
Receiver hears acknowledgement, adds Originator to network and transmits acknowledgement
$NET,ACK*
Originator hears acknowledgement and is successfully added to network.

Figure 4: 3-Way Handshake

Additionally, error detection has been incorporated into the Project. As can be found in Appendix 4: Robo-Chopper Communications Protocol, each data packet (See Figure 5: Data Packet Detail) includes message number to add in handling missed transmissions, a checksum that is calculated using the addition of each byte value of a data packet and then inserting that value at the end of the data packet before transmission. At the receiving end, the data packet again goes through this addition process and if the calculated checksum and the transmitted checksum do not match, the data packet is treated as garbage and thrown away. No facilities for re-transmission are provided, as this would result in longer delay between receiving following data packets. However, since re-transmission is not provided important data packets (such as the previously discussed “Kill Switch”) must be transmitted continuously to ensure reception.

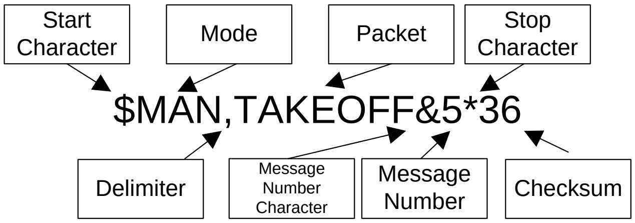


Figure 5: Data Packet Detail

Brief Overview of Operation

RCU User Controls and Indicators

Figure XX. RCU Detail – To Be Completed

Thumbsticks:

A. Left Thumbstick Horizontal - Adjust Vehicle Yaw
B. Left Thumbstick Vertical - Adjust Vehicle Throttle
C. Right Thumbstick Horizontal - Adjust Vehicle Roll
D. Right Thumbstick Vertical - Adjust Vehicle Pitch

Buttons and Switch’s

E. Takeoff
F. Hover
G. Land
H. Return Home
I. Reset Computer
J. Enable/Disable Motor Power

Indicators

K. LCD Display

L. LED Bar Graph

LED Display:
See Figure 6: LED Display for specifications on the different modes the LED Display operates under.

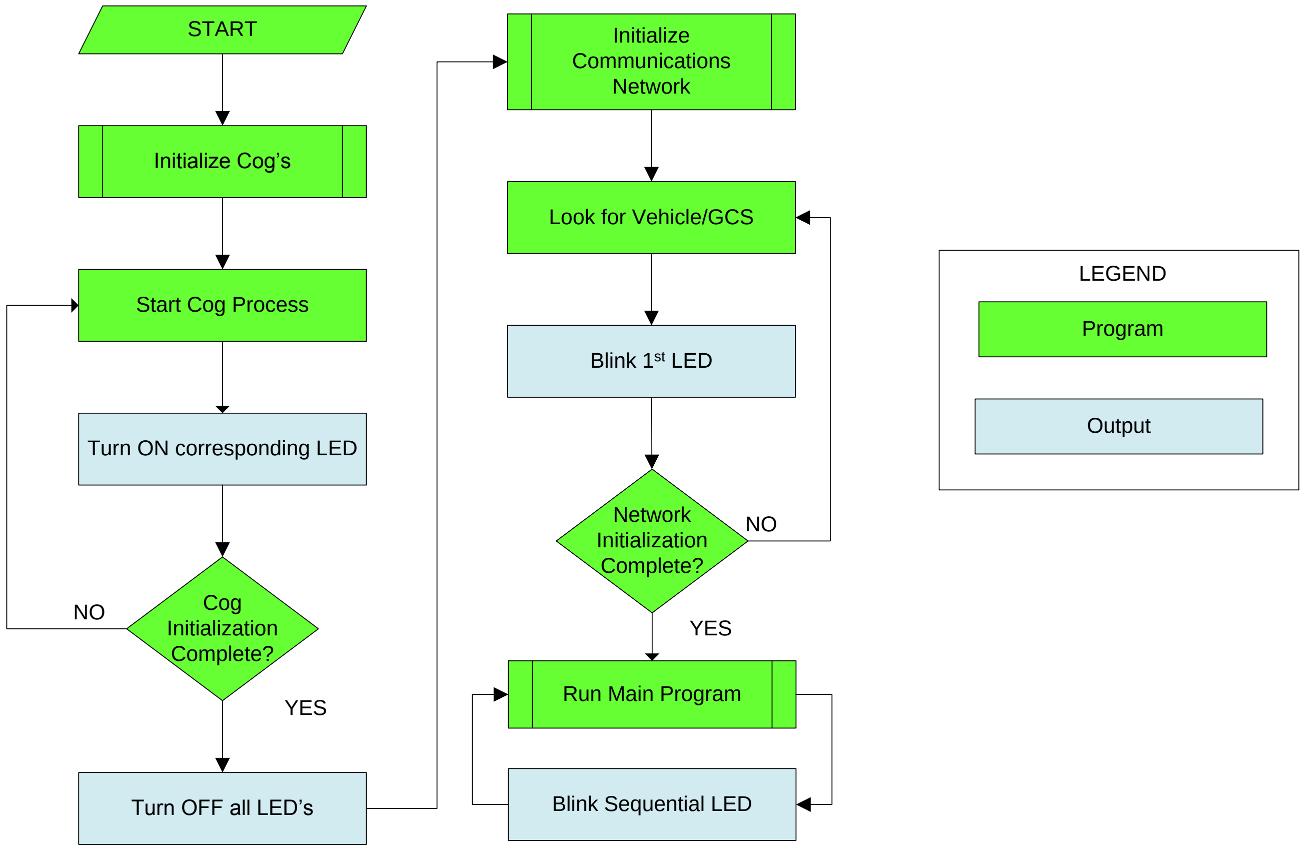


Figure 6: LED Display

LCD Display:
Line 1 - Operating Mode
Line 2 - Error Code

Vibrating Motors:
Functionality has not been implemented. Possible applications are:

1. How fast the Vehicle is rotating along the Yaw Axis
2. How fast the Vehicle is moving over the ground
3. The distance to the Ground while the Vehicle is landing.
4. If the Vehicle detects an obstacle in its flight path.

Software Versions

Continued development of the RCU software is taking place. See Table 5: RCU Software Versions for a complete listing. Unless otherwise noted, each increase in version number expands the capabilities of the RCU. The version number indicates that the capabilities listed in it have been completely designed, implemented, validated and tested, whereas a decimal version implies that the for the next version certain parts are completed, but not to the entirety to the full version number. Any Version # highlighted in red is a planned development. For a complete description of the RCU's current software version, see Appendix 5: Operator's Manual.

Table 5: RCU Software Versions

Ver #	Capability
1.0	Manual Control, Network Initialization, Error Display
2.0	Communication with GCS and Vehicle
3.0	Limited Autonomous Functions, Force Feedback
4.0	Enhanced Autonomous Functions

Version 1.0: Manual Control of the Vehicle using the Analog Joysticks, including ability to remove power from Vehicle motor controllers using Kill Switch. Network initialization using 3-way handshake. Error display on LCD Screen. RCU will only work with Vehicle, no support for the GCS is possible.

Version 2.0: Using XBee API Mode, RCU can communicate to GCS and/or Vehicle.

Version 3.0: Limited autonomous functions (non-GPS enabled) such as Land, Take-Off and Hover implemented. Force Feedback on RCU, specifics to be developed.

Version 4.0: Enhanced autonomous functions (GPS enabled) such as Return to RCU, Follow-Me, etc.

Error Codes

1 - 2000: Vehicle SOM Error
2001 - 4000: Vehicle Propellor Error
2001: Communications Processor started unsuccessfully.
2002: PWM Processor started unsuccessfully.
2003: Encoder Processor started unsuccessfully.
2004: Floating Point Processor started unsuccessfully.
2005: Kalman Filter Processor started unsuccessfully.
2008: Propellor started unsuccessfully (General Error).
4001 - 6000: Vehicle Sensor Error
4001: GPS Signal Invalid or Not Present
6001 - 8000: Vehicle Navigation Error
8001 - 10000: Vehicle General Error
8020: Network Error (General Error).

20001 - 22000: Remote Control Unit Error
20001: Radio Processor started unsuccessfully.
20002: LCD Processor started unsuccessfully.
20003: Analog-To-Digital Processor started unsuccessfully.
20004: Floating Point Processor started unsuccessfully.
20005: GPS Processor started unsuccessfully.
20008: Propeller started unsuccessfully (General Error).

20011: Interface Not Found on Network. (API)
20012: Vehicle Not Found on Network. (API)
20013: GPS Signal Invalid or Not Present
20020: Network Error (General Error).
22000: General Error.

22001 - 24000: Interface Error
22020: Network Error (General Error).

Figures

Tables

Appendix

1. Schematics
2. Program Listing
3. Power Analysis
4. Robo-Chopper Communications Protocol
5. Operator’s Manual
6. Datasheets

References

Parallax. (2010, August 26). *Propeller General Information*. Retrieved August 26, 2010, from Parallax: http://www.parallax.com/tabid/407/Default.aspx