Node-Based Range-Extending Recon Drone

Final Report

ECE 445 - Fall 2018

Team 27

Dhruv Diddi (diddi2)
Thomas Korenchan (kornchn2)
Michael Lally (mlally2)

TA: Hershel Rege

12 December, 2018

Abstract

Our project addresses limitations in operational space, the region in which an operator can control a drone or robotic system, with the idea that a geometrically complex wireless network can be established on-the-fly. Without a pre-established network, operators are often limited to line-of-sight communication with their vehicle. This prevents complex operations around obstacles or inside structures.

A node-based solution, in which the nodes are deployed by the main vehicle during operation, can allow an operator to maintain control of the vehicle far outside of the original line-of-sight. Nodes placed at strategic junctions can re-transmit operational commands, daisy-chaining them from operator to lead vehicle.

A two-node vehicle make up the body of our project. The lead vehicle tows the intermediate, unhooking it at desired locations. The lead is then free to operate anywhere in the intermediate's line of sight, effectively extending its operational space.

Contents

1. Introduction

1.1 Objective

The utility of drones in reconnaissance roles is a large and expanding field. Drones are used to assess the spread of both urban and forest fires. They're increasingly used in search and rescue roles in a variety of scenarios; they can fly to and deliver floatation devices to struggling swimmers, locate lost or distressed hikers, and have been tested as a means of locating survivors in building collapses and fires.

However, their utility is still largely limited to exterior action, either operating above the canopy of the forest or in the air around a burning building, as examples. Their ability to operate is limited by where their command signals can reach, making control in locations like building interiors, forests floors, sewers or tunnels, or anywhere dense with obstructions unreliable.

Our objective was to address this issue with a node-based range-extending solution. With out solution, as a drone reaches its command space limits, either due to range limits or interfering obstructions, it deploys a node. That node, still within the operator's command space, retransmits the operator's instruction. This strategy has several benefits. Nodes placed just beyond a corner or obstruction, or at a nexus of corridors, transmit commands and allow operators to maintain control of a drone beyond their line-of-sight. The drone may then continue to operate around corners through corridors, further than previously possible while relying on the original operator's signal. Nodes placed near the outer limit of the operator's signal effectively increase the operators range. These benefits are achieved on-the-fly, without the need to set up a network.

Our vehicle takes the form of two segments linked together; the lead segment is the main vehicle while the intermediate segment acts as a node. While both segments are independent vehicles capable of discrete movement, operator commands are received first by the node which then retransmits them to the lead. The node is capable of detaching itself from the lead and positioning itself at an advantageous retransmission position. It acts as a middleman, routing instructions to the lead and allowing the operator to maintain control even as the lead passes out of the operator's line-of-sight.

1.2 Background

Our project wasn't the first foray into drones as a means of reconnaissance and rescue. In the days following the September 11, 2001 attacks in New York City, researchers from the University of South Florida, funded by the National Science Foundation (NSF), were on-site at ground zero using prototype robotic vehicles to search the rubble for survivors and remains. They were limited in their ability, however, as their vehicles were tethered to their operators with a maximum range of 100 feet. Our project improves on this maximum range with vehicles that can deploy nodes at critical junctions and nexuses in their search space [1]. We waived the need for a tether in favor of wireless commands, given that wireless routing between nodes grants more freedom than a physically restrictive tether.

It's also not the first exploration into the use of drones as a mesh network to extend connection. In a recent paper from University of Waterloo, researchers discussed the utility of drones as a means of extending vehicular networks. These networks enable self-driving cars to communicate, share information, and avoid collisions. Their paper touches on "Line-of-Sight Links", the principle that "drones flying in the sky have a higher probability to connect ground nodes" [2], specifically cars, due to their ability to establish line-of-sight connections over ground-based obstacles like buildings, trees, and other vehicles. Ours leverages the same principle: by deploying our nodes at locations that provide wide coverage over previously obstructed regions, we achieve the same effectiveness as a direct line-of-sight link. Though there is no direct line-of-sight between the operator and the lead segment, a daisy-chain of retransmitting nodes provides a comparably effective connection.

1.3 High Level Functionality

- Our project's lead vehicle can reach an unobstructed range of 500 feet, using the node as an intermediary for commands from our controller. This exceeds the range that the controller alone can transmit messages.

- Our project's node vehicle, while in the line-of-sight of both the lead vehicle and the controller, can route commands from the controller to the lead vehicle. This allows the lead vehicle to operate in a location outside the controller's direct line-of-sight.

- Each segment of our solution - controller, node segment and lead segment - can maintain a network bandwidth of 64 kbps, which is the minimum data rate required to send 802.15.4 messages.

- Our solution can maintain a minimum operation time of 30 minutes. This is based on the information that industry-standard drones have an operational time of 20-30 minutes [3].

1.4 Systems Overview

Our design can be described as three major systems: the Controller, Vehicle and Software. The controller and vehicle systems are made up of Power Supply, XBee Module, and Processing subsystems, which are described in detail later in this report. The Software system is considered the “brains” of the design: it integrates the Controller and any number of vehicle systems together.

Figure 1 outlines the major blocks used in our controller design. The overall functionality of the controller is straightforward: a user controls four buttons and two switches to operate the node and lead vehicles. The buttons are used for Latch Operation, Forward, Left and Right, while the two switches are used to designate which vehicles is active (Node, Lead or both).

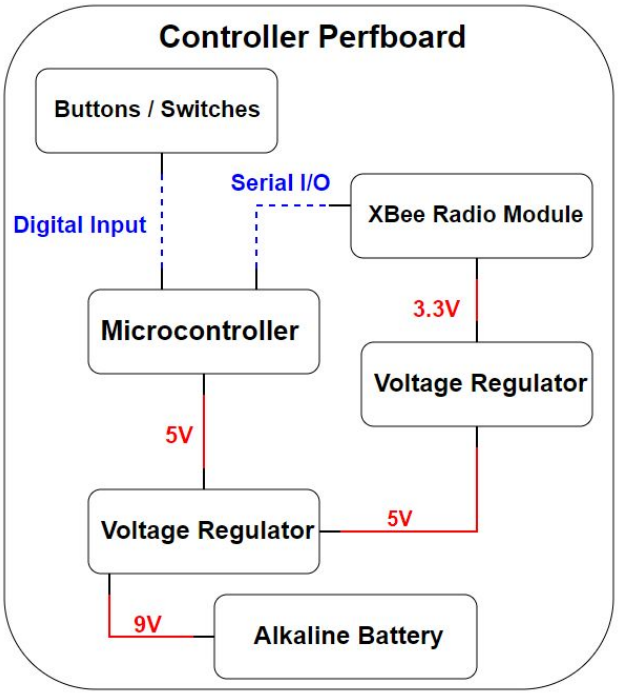

Figure 1. Controller Perfboard Schematic

Similar to our controller design, Figure 2 outlines the major blocks of our vehicle design. The overall design is almost exactly the same as the controller, with the buttons and switches replaced with the wheel and latch motors instead.

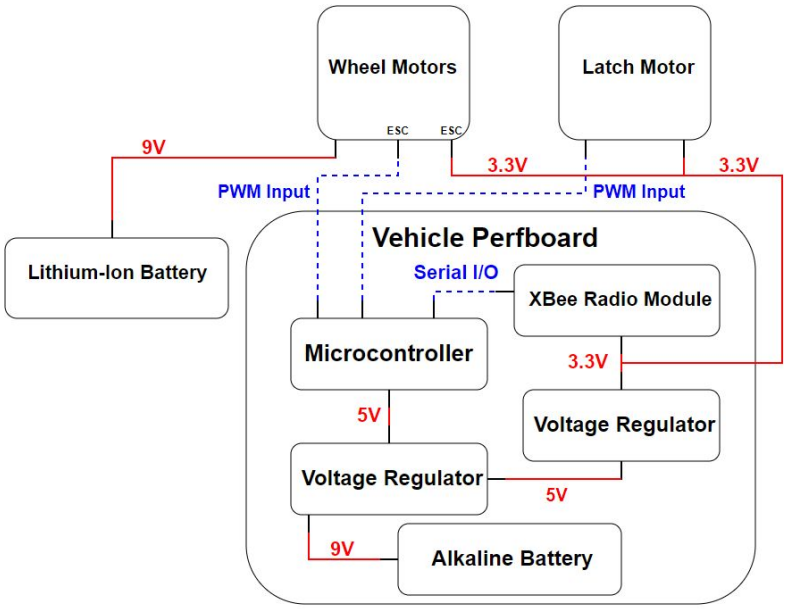

Figure 2. Vehicle Perfboard Schematic

The flowchart in Figure 10 (**Appendix A**) illustrates how our software loaded onto our microcontrollers tie all of our design's subsystems together. The Controller microcontroller takes digital input from buttons and switches and encodes a data packet of one byte (8 bits). This packet is transmitted over our XBee network to the node and lead vehicles, where the packets are interpreted and processed.

2. Design

2.1 Design Procedure

Our solution design can be described as three major subsystems: Controller, Vehicle and Software. Over the course of the design process, there were many opportunities to make decisions that would impact components of our design. Here, we will elaborate on the more crucial decisions made about these three subsystems.

A major modification made to our overall design was the utilization of perfboards, which was a decision in response to the state of our Printed Circuit Board (PCB) towards the end of our design process. After much review, our PCB design did not meet our functional requirements, specifically as it pertained to our USB-to-UART conversion chip used to load Arduino sketches onto our microcontrollers. The decision to use perfboards to integrate our subsystems together as a replacement for PCBs was not only the most crucial decision we made but also a useful one. With perfboards, we were able to replace the ATmega2560 microcontrollers we had originally intended to use on our controller and vehicles with pre-bootloaded ATmega328P's: a

simpler chip with through-hole pins, which was convenient for connecting/removing the chip from our boards when testing our final design.

Another major decision made was the design of the physical vehicle body, which we 3D-printed. In one aspect, we had originally designed the body to use treads as the method of movement, so as to obtain all-terrain maneuverability. In the end, we decided to use four wheels with the front two wheels driving each vehicle: this allowed for sharper turning and faster speeds. In the process, we ran into an obstacle with the wheels involving traction: another decision made to wrap our wheels with rubber bands allowed us to obtain better traction so our wheels could operate to our specifications.

Lastly, we decided to modify our software by changing the data packet encoding. Figure 3 shows our updated encoding, which enumerates all of our controller inputs as single bits. This method of encoding not only allowed us to reduce the packet size to a single byte, but also enables each vehicle to be distinguished from each other, so that they may process only packets intended for them.

Bit 7 = Lead Vehicle Active
Bit 6 = Node Vehicle Active
Bit 5 = X
Bit 4 = Operate Latch
Bit 3 = X
Bit 2 = Forward
Bit 1 = Left
Bit 0 = Right

Figure 3. Data Packet Encoding

While our design did not require any major equations, we did have multiple tools that were essential to the success of our project: XBee Configuration & Testing Utility (X-CTU) and the Arduino Integrated Development Environment (IDE). X-CTU was crucial to our design implementation, as its purpose was to enable configuration of the XBee radio module firmware. The firmware configuration was the foundation that allowed our modules to be on the same network, and communicate in the specific fashion that we determined through testing. Just as crucial was the Arduino IDE, which of course we used to program our Microcontroller Units (MCU) at all stages of design development, from testing XBee communication to motor functionality to full integration.

2.2 Design Details

As described in the Systems Overview, our solution is comprised of the Controller, Vehicle and Software systems. The subsystems of the Controller and Vehicle designs, as well as the Software system design, are elaborated upon in the following subsections.

2.2.1 Controller

The power supply consists of a single 9V alkaline battery and two voltage regulators, one for 3.3V and 5V respectively. Figure 4 illustrates the power supply subcircuit. The regulated 3.3V was used only to supply power to the XBee module subcircuit, whereas the regulated 5V was used to supply both the microcontroller subcircuit as well as the 3.3V regulator, since it required an input voltage no greater than 6.5V [4].

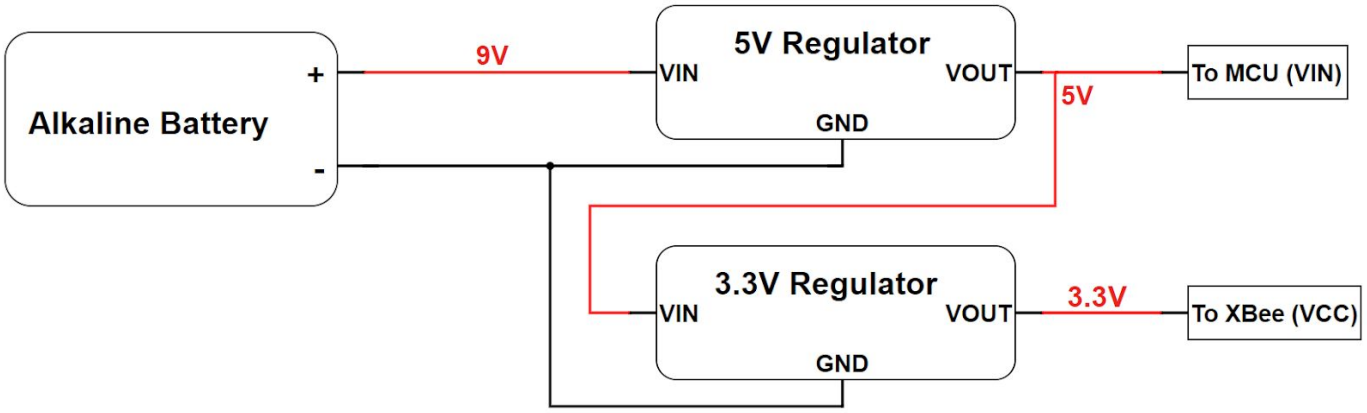

Figure 4. Controller Power Supply subcircuit

The XBee module subcircuit consists only of the module itself, as depicted in Figure 5. Interfacing with XBee modules was very simple: it requires a regulated 3.3V supply (Pin 1), a grounding connection (Pin 10), and has two pinouts which connect to our microcontroller: Data Out and Data In (Pins 2 and 3, respectively) [5]. These pins allow the XBee to communicate with the microcontroller Serial Data ports for real-time data transfer.

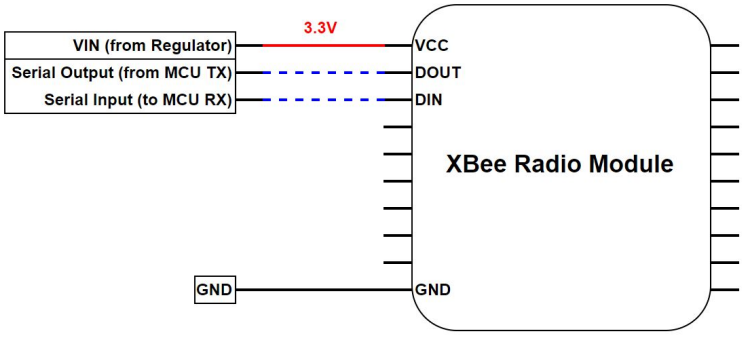

Figure 5. XBee Module subcircuit

A major design decisions that we made in the early stages of development was the configuration of our XBee modules. The models that we used are outdated, and do not have Zigbee protocol capability (Zigbee is based on the 802.15.4 protocol). As a result, the modules could not be configured as routers (two-way communication). However, through a clever configuration in the modules' firmware, we were able to achieve the network we needed, as depicted in Figure 6. The Pan ID must be the same for all XBees to talk to each other. To specify how they should talk, the Destination Low (DL) and Source Address (MY) addresses were used as depicted [6]. To test our configuration, we set up a simulation: we connected three XBee modules to three Arduino Uno boards to simulate the roles of the Controller, Node and Lead vehicles respectively. The Controller sent an encoded message intended for the Node, which would send back an acknowledgement packet. The Node would then transmit a message to the Lead, which would send back an acknowledgement after reception. The Node, upon receiving the Lead's acknowledgement, would retransmit it to the Controller. This test was successful, proving that our configuration would work for the scope of our design.

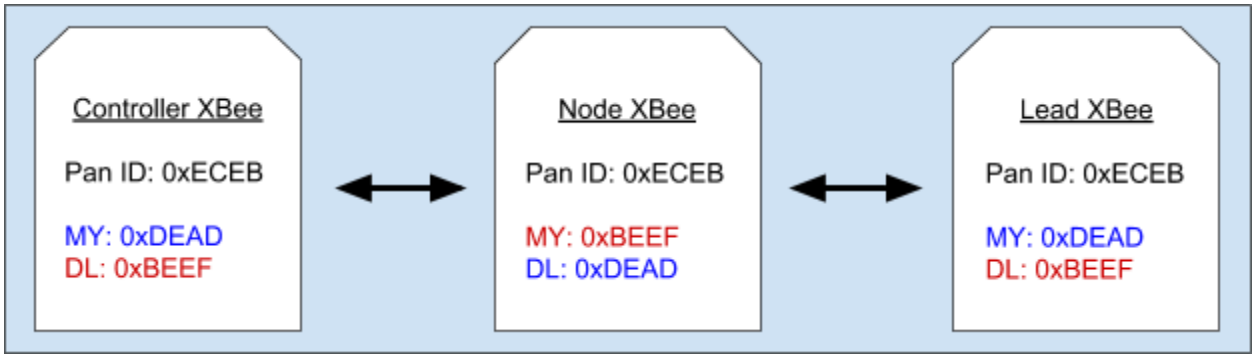

Figure 6. XBee Network Configuration

Our input processing subcircuit, as shown in Figure 7, is comprised of seven components: four digital buttons, two digital switches, and the ATmega328P chip. The chip takes a regulated 5V supply from our regulator. As the input processing and control unit, the microcontroller interfaces the button and switch inputs with the XBee module. The inputs are read through six digital Input/Output (I/O) pins, and are encoded into a single byte packet. The microcontroller then sends the packet over the Serial Data port, using its RX and TX Serial Data pins (Pins 0 and 1, respectively), to the XBee module to be transmitted through our network to the node and lead vehicles [7].

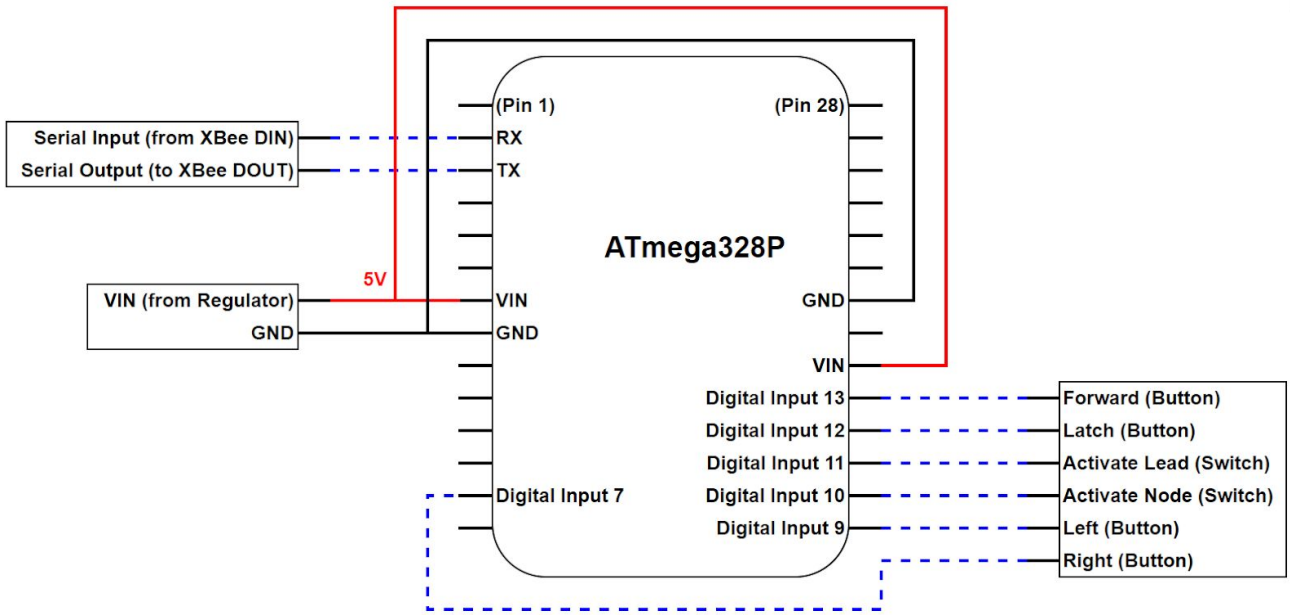

Figure 7. Input Processing subcircuit

To test our input processing, we made a decision to modify our previous data encoding. For more information, see **Section 2.2.3**.

2.2.2 Vehicle

The power supply for our vehicles, shown in Figure 8, is exactly the same as our controller: a 9V alkaline battery which interfaces with two voltage regulators at 5V and 3.3V. In addition, we include a 9V Lithium-Ion battery which exclusively powers our wheel motors. Lithium-Ion batteries have better capacity than alkaline ones, and our wheel motors require a considerably higher voltage and current relative to all of our other components.

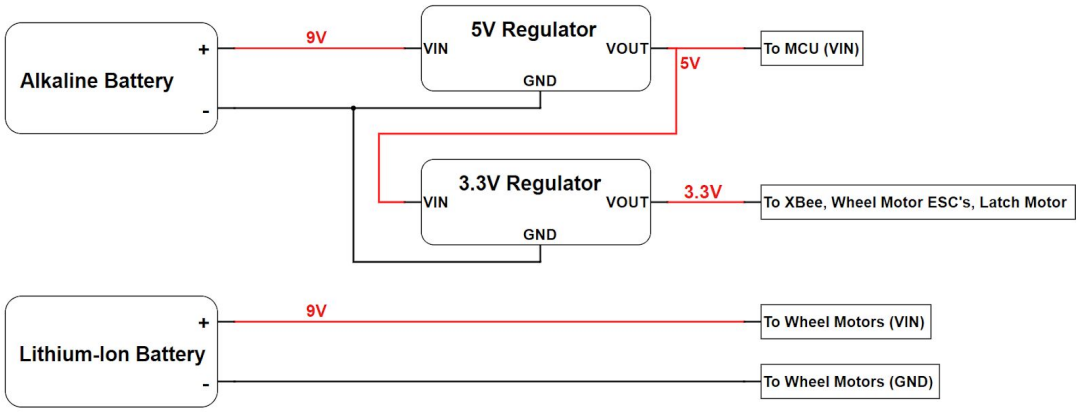

Figure 8. Vehicle Power Supply subcircuit

The XBee subcircuit and network configuration simulation is exactly the same as that of the controller. See Figure 5.

Our output processing subcircuit, depicted in Figure 9, is very similar to the input processing subcircuit on our controller. We again use an ATmega328P microcontroller chip which takes a regulated 5V input voltage from our regulator, and grounding connections. The chip's RX and TX pins (Pins 0 and 1, respectively) are again connected to the XBee to communicate with the controller and other vehicle. Instead of digital button and switch input, however, the vehicle microcontroller instead outputs a Pulse-Width Modulation (PWM) signal to the two wheel motors and latch motor. The latch motor takes the PWM input directly, whereas the two wheel motors take it indirectly through an Electronic Speed Control (ESC) module. This module converts the PWM signal into a 3-phase signal, which is used to control the wheel motor's rotation [8]. Both the wheel motor ESC's and the latch motor take regulated 3.3V input and grounding connections along with the PWM data.

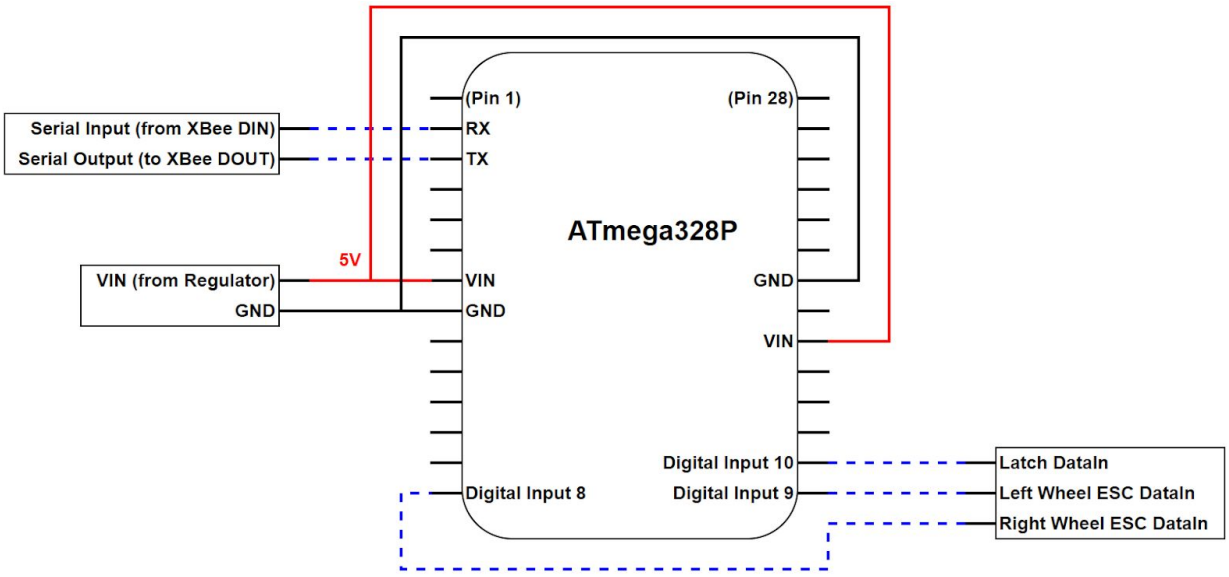

Figure 9. Output Processing subcircuit

To test our output processing, we made a decision to modify our previous data encoding. For more information, see **Section 2.2.3**.

2.2.3 Software

The software flowchart detailing packet encoding on our Controller microcontroller is shown in Figure 11 (**Appendix A**). Figure 12 (**Appendix A**) illustrates how our Vehicle microcontrollers process packets from the Controller and update the wheel and latch motors accordingly. Using these programs, we simulated both input and output processing.

Table 1 enumerates the steps and per-step results for our input processing. Table 2 depicts the same type of information for output processing.

Steps	Per-Step Results
1. Read inputs from digital I/O pins 3-8. Print to console. 2. Encode values into byte packet. Print packet to console, press Forward button (should be bit 2 HIGH). 3. Encode values into byte packet. Print packet to console, change Lead_Active switch (should be bit 7 HIGH). 4. Encode values into byte packet. Print packet to console, change Lead_Active switch and press Forward button (should be bit 7,2 HIGH).	1. All buttons/switches show visible change in digital signal. Noted that they are ACTIVE LOW. 2. Read encoded packet on console: **00000100** (Forward HIGH) 3. Read encoded packet on console: **10000000** (Lead_Active HIGH) 4. Read encoded packet on console: **10000100** (Lead_Active and Forward HIGH)

Table 1. Input encoding simulation steps

Steps	Per-Step Results
1. Hardcode encoded packet for Lead to move Forward (**10000100**). Print received messages from Vehicle to Serial Monitor. 2. Test updateMovement() functionality. Hardcode 1300 PWM input to wheel motors (lowest speed) if packet is **10000100**. 3. Test Node_Active switch HIGH, Forward HIGH from switch/button. Hardcode 1300 PWM input to wheel motors (lowest speed) if packet is **01000100**. 4. Test updateLatch() functionality, Node_Active switch HIGH, Latch HIGH from switch/button. Hardcode 2000 PWM input to latch motor (raise up) if packet is **01010000**.	1. Read Serial Monitor: **10000100** prints indefinitely. 2. Read Serial Monitor: **10000100** prints indefinitely. Wheel motor spinning. 3. Read Serial Monitor: **01000100** prints indefinitely. Wheel motor spinning, updateMovement() processes packet correctly. 4. Read Serial Monitor: **01000100** prints indefinitely. Latch motor raised indefinitely, updateLatch() processes packet correctly.

Table 2. Output processing simulation steps

3. Cost & Schedule

3.1 Cost

The following section outlines the development costs of the project and delineates the different parts required during the development of the project. We've also added additional costs of processes like 3D Printing, Labor effort and PCB printing which are key components in the development of the project. Our Bill of Materials is listed in Table 3.

Part	Cost (prototype)	Cost (bulk)
Microcontroller * 3 (Microchip, ATMEGA2560V-8AU)	$6.45	$5.37
XBee Radio chip * 3 (Digi-Key, 602-1892-ND)	$75.00	$51.00
Wheel motors * 4 (Amazon, YoungRC A2212 1000KV Brushless Motor)	$59.94	$59.94
Latch motors * 1 (Amazon, KOOKYE 1PCS Servo Motor Metal Gear 180 Degree Rotation)	$8.99	$8.99
Batteries * 3 (Energizer, 522 9 Volt Alkaline Battery)	$9.39	$9.39
Batteries * 4 (Energizer, 522 9 Volt Lithium Battery)	$20.00	$8.00
PerfBoard	$2.00	$0.50
Assorted circuit components (resistors, capacitors, ICs, etc) (Digikey; est.)	$10.00	$4.0
3D printer filament 1kg * 2 (Hatchbox, 3D PLA-1KG1.75-BLK)	$39.98	$17.99
Total	$271.73	$165.18

Table 3. Bill of Materials

Additionally, we include a cost of $2 per hour of 3D printer time over 30 hours. We incur an additional cost of $60.

Our labor costs estimate a $45/hour salary per engineer. They also estimate a 10 hour per week work schedule per engineer, over the 15 week duration of ECE 445. 3 engineers are working on our team.

$$\frac{\$45}{1\ hour} * \frac{10\ hours}{1\ week} * \frac{15\ weeks}{1} * 3 * 2.5 = \$50,625.00$$

Our estimated cost of labor is: **$50,625.00**

With the following costs:

- Parts: $271.73
- 3D Printing: $60.00
- Labor: $50,625.00

Our projects grand total is: **$50,956.73**.

3.2 Schedule

Our project was completed between the weeks of September 30th and December 12th. For a specific per-week and per-member breakdown of work, see Table 4 in **Appendix B**.

4. Requirements & Verification

Our project combined multiple electrical components which we had to individually verify requirements, before integrating the circuit together. The components has to be tested and verified in a specific order to minimise chances of damage to the circuitry and the design.

Following is a summary of requirements of verifications of important components of our project:

- Power Supply - The power circuit involved the batteries and the voltage regulators required to supply sufficient voltage and amperage to all the components. We used two kinds of batteries, alkaline and lithium, which both have different tolerance and delay ranges. We were able to ensure that the batteries were sufficient for our power needs by performing charge tests on the batteries. We were also able to confirm reliable output through the linear regulators through multimeter measurements.
- Wheel and Latch Motors - After testing out the power circuit, we connected our motors to an arduino to simulate MCU control and tested the torque and lift abilities of the motors. We were able to confirm that the motors were powerful enough for the project.
- XBee Modules - Through the X-CTU interface, we were able to configure and test the network of the XBee modules.
- MCU and Software - We were able to upload sketches and process input data packets from the XBee at the expected baud rates. We are also able to send PWM signals to control the motors through the ESC. We confirmed that we were able to send reliable signal out from the pins and the software processed the packet buffer properly.
- Circuit Peripherals - USB-to-UART was an unsuccessful attempt to upload sketches to the MCU. We overcame this failure and accomplished our original goal through the Serial Programmer on the Arduino. We were able to test the reliability of switches through a simple circuit. Light-Emitting Diodes (LED) were not implemented in the final design, but are a simple addition to the MCU output.

For further data regarding requirements and verifications, refer to the tables in **Appendix C**.

5. Conclusion

This section describes the project accomplishments as well as obstacles faced during development. The status of the project's three main components - the vehicle, the controller and the software - is discussed. Through the project's design phase, our project goals were met while making adapting to our original proposal and technology. The following sections elaborate on the different aspects of the project.

5.1 Accomplishments

The project successfully met our original requirements. In conclusion we were able to showcase the following:

- A working prototype of a daisy chained networked components which can communicate upto 500 feet in outdoor environments.
- A vehicle node being controlled by a controller node with a reliable packet transfer rate of upto 64 kbps for a scalable modules of nodes
- The ability to have controller and vehicle nodes operational more than 30 minutes at a stretch
- Comprehensive control of wheel as well as latch motors with are reactive to controller signals for navigation
- A scalable design for circuit module which could be housed in multi-functional systems like flying drones, aquatic drones etc.

Through the course of the project we were exposed to various elements of the design process such as PCB design, soldering, embedded systems, signal processing, and control systems. Together, we significantly deepened our technical skills; our soft skills needed for communication and work distribution were also further developed. Our team successfully collaborated to complete the project's requirements.

5.2 Uncertainties

Over the course of the project we overcame multiple challenges to fulfil the project's goals. Adaptations to our technology and structural design were necessary to reach on operational project.

One major cause for uncertainty in the project was PCB design. Unfamiliarity with the PCB design software meant that we had limited ability in PCB design. This led to delays in delivery of PCB design printing . The prototype PCB design traces were not consistent with the pin layout for certain components and the traces had potential misconnections. In the interest of time, we decided to solder our components onto a perf board and make good connection with a

combination of single core and threadwire. We were able to verify the viability of perf boarded components through unit and integration tests.

Another hiccup in our development was the delay in 3D printing prototyping. 3D Printing facilities around campus had unexpectedly long queues which delayed the number of design iterations we could test for the project. Alternatively, we were able to buy housing and wheel components from local vendors. This reinforced our confidence in the modularity of the node circuits.

5.3 Ethical considerations

Safe and ethical practices were a concern in the completion of this project as multiple components of the vehicle that required safety considerations. In development of the radio module, we fully adhered to Federal Communications Commission (FCC) standards for amateur radio communication, so as to avoid harmful interference with other entities. We ensured that the channels of communication for the RF module were not susceptible to attacks from proxy controllers. XBee RF series 1 modules allow encrypted transmission and the Pan ID-based network access model ensures only network-configured XBee devices are able to communicate reliably on our network [5].

Another area for consideration was the battery. We planned on using alkaline and lithium batteries to power our vehicle segments, which can be dangerous if handled incorrectly. In order to “hold paramount the safety, health, and welfare” [9] of us and those around us we strictly followed standard safe battery handling procedures.

Lastly, all circuits were unexposed and use good engineering practices to ensure circuitry is properly soldered and tested for temperature conditions with the battery loads. The motors from the circuit are also susceptible to overheating and damage from running at improper voltage inputs. All components of the project are protected by voltage regulators which ensure that the connected devices receive proper inputs to maintain a good working state. PLA filament has a glass transition temperature of 60–65°C and a melting temperature of 173–178°C. With proper mounting we ensured that the 3D printed design is not affected by the heat from the circuitry or the motors.

5.4 Future work

In the development of the project, we demonstrated the viability of a node-based system in extending a drone operator's effective command space. In future versions of the project, we would like to add the following features:

- Better wheel design for navigation in rough terrain
- Mesh packet routing to be able to address non-daisy chained nodes
- Higher communication bandwidth to support camera feed and larger network size
- The ability to house and test this circuitry in airbourne and aquatic drones
- Increase robustness of circuit housing to operate in extreme temperature and pressure conditions

Network access is an important aspect in the drones control systems of the future. With robots become viable for more jobs which are considered unsafe for humans, it's important to deploy systems like our project that extend their effective range, increasing the operational abilities of these robots.

6. References

[1] nsf.gov. At WTC Search, Graduate Students Deploy Shoebox-Sized Robots, 2001 [Online]. Available: https://www.nsf.gov/od/lpa/news/press/01/pr0178.htm [Accessed 20 Sep. 2018].

[2] W. Shi, H. Zhou, J. Li, W. Xu, N. Zhang and X. Shen, "Drone Assisted Vehicular Networks: Architecture, Challenges and Opportunities," in IEEE Network, vol. 32, no. 3, pp. 130-137, May/June 2018.

[3] Wondershare Inc., “Top 10 Drones with Longest Flight Time for 2018”, 2018 [Online]. Available: https://filmora.wondershare.com/drones/drones-with-longest-flight-time.html [Accessed 4 October 2018]

[4] ABLIC Inc., “Ultra Low Current Consumption and Low Dropout CMOS Voltage Regulator”, 3.3V Linear Regulator Datasheet, 2015. [Online]. Available: https://www.mouser.com/datasheet/2/360/S1206_E-20108.pdf

[5] Digi International Inc., “XBee/XBee-PRO RF Modules”, XBee Series 1 RF module datasheet, 2009. [Online]. Available: https://www.sparkfun.com/datasheets/Wireless/Zigbee/XBee-Datasheet.pdf [Accessed 4 October 2018].

[6] Daniel Godrick, “Maker Challenger: Wirelessly Control Lights and Motors Using XBee Communication!”, 2018 [Online]. Available: https://www.teachengineering.org/makerchallenges/view/cub-2219-xbee-lights-motors-arduino-wireless-communication [Accessed 11 November 2018].

[7] HobbyTronics, “Arduino ATmega328 Pinout”, ATmega328P pinout, 2018. [Online]. Available: http://www.hobbytronics.co.uk/arduino-atmega328-pinout [Accessed 1 December 2018].

[8] “30A BLDC ESC”, Electronic Speed Control Product Manual, 2012. [Online]. Available: https://www.optimusdigital.ro/index.php?controller=attachment&id_attachment=451 [Accessed 10 November 2018].

[9] IEEE, “IEEE Code of Ethics”, January 2018. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html [Accessed 4 October 2018].

7. Appendix

7.1 Appendix A - Diagrams

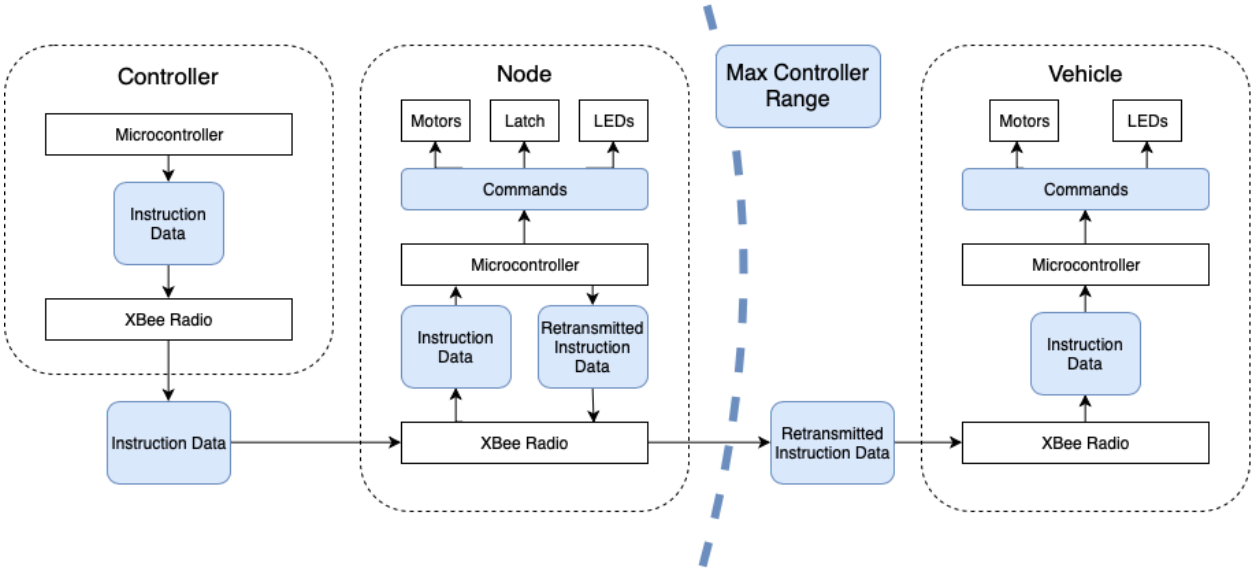

Figure 10. High-Level Software flowchart

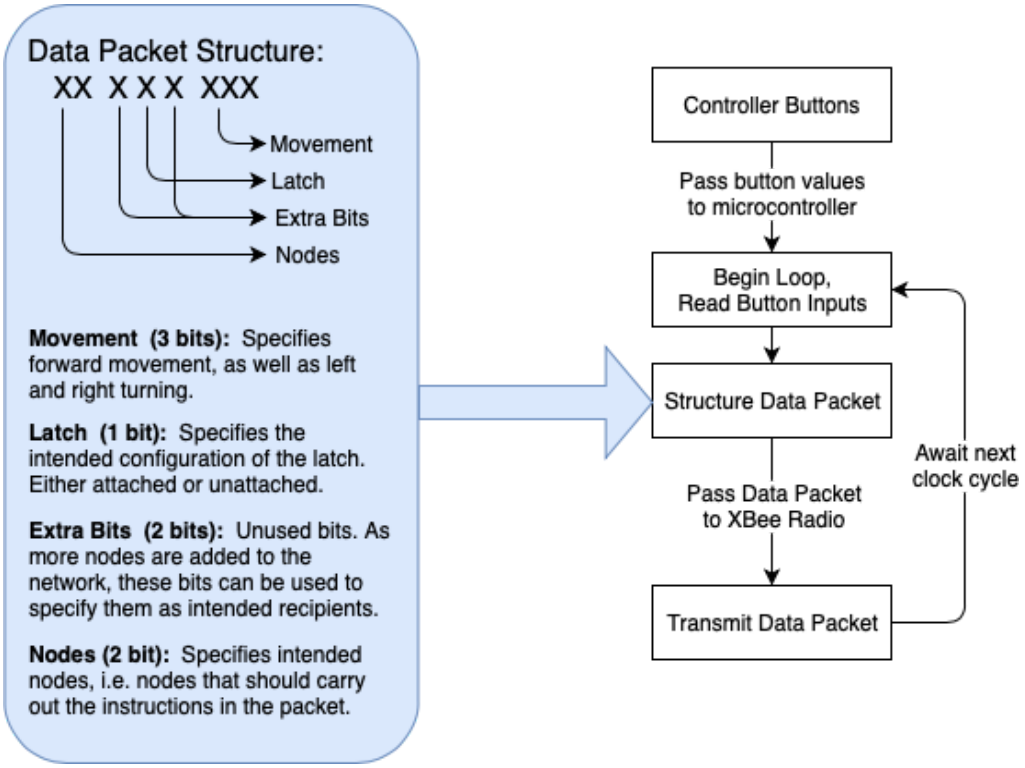

Figure 11. Input encoding flowchart

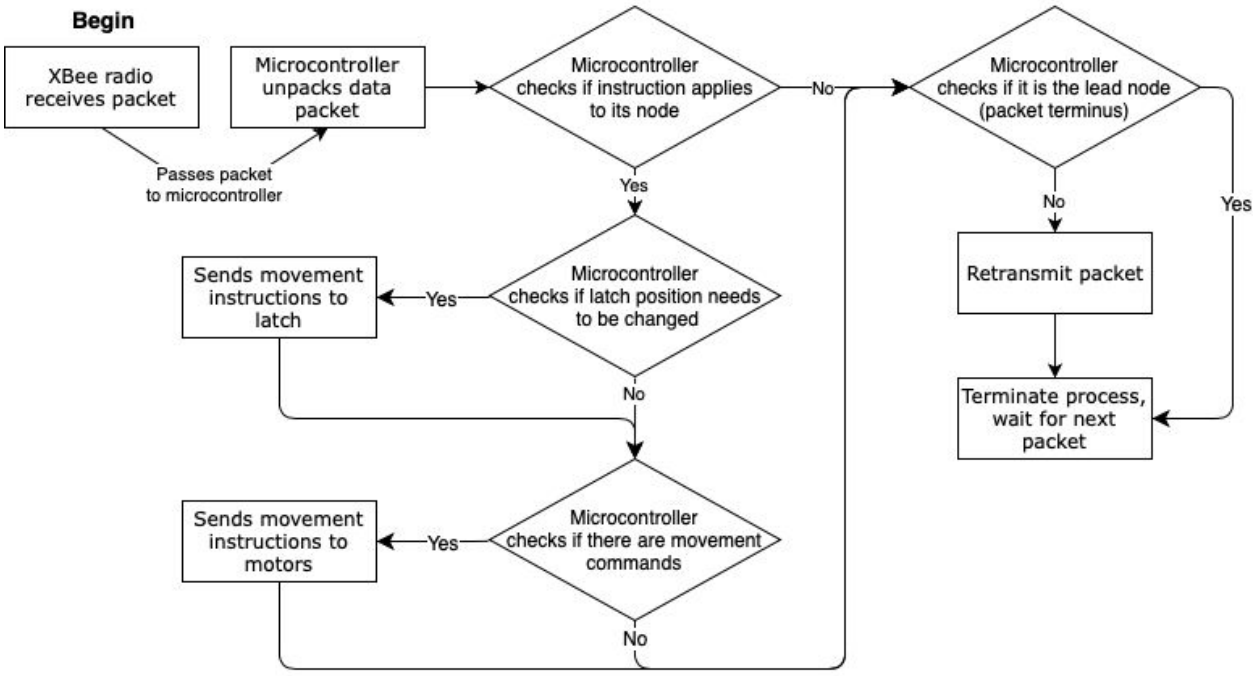

Figure 12. Data Packet Processing flowchart

7.2 Appendix B - Schedule of Work

Week	Task	Delegation
9/30	Prepare for Mock Design Review, work on Design Document	Dhruv
	Prepare for Mock Design Review, work on Design Document	Michael
	Prepare for Mock Design Review, work on Design Document	Thomas
10/7	Finalize controller PCB designs	Dhruv
	Design software flow diagrams	Michael
	Design wheel, treads, and chassis models for 3D printing	Thomas
10/14	Finalize vehicle PCB designs	Dhruv
	Finalize software flow diagrams	Michael
	Print version 1 wheels, treads, and chassis	Thomas
10/21	Begin version 2 PCBs, order components	Dhruv
	Begin coding packet transmission protocol	Michael
	Test node connections (controller: movement inputs)	Thomas
10/28	Finalize version 2 PCBs and order	Dhruv
	Configure XBee radios for group communication	Michael
	Configure XBee radios for group communication	Thomas
11/4	Test components, finalize controller design	Dhruv
	Design node connection tests	Michael
	Design node connection tests	Thomas
11/11	Integrate original microcontroller with circuit	Dhruv
	Test software logic for node connections and LED control	Michael
	Debug node connection tests and XBee firmware	Thomas

11/18	Design Test	Dhruv
	Test node connections with motor controls	Michael
	Test node connections with motor controls	Thomas
11/25	Design Test	Dhruv
	Design Test	Michael
	Design Test	Thomas
12/2	Node and vehicle design modifications and fixes	Dhruv
	Software bug fixes and improvements	Michael
	Controller and node design modifications	Thomas
12/9	Final Presentations, finish final paper	Dhruv
	Final Presentations, finish final paper	Michael
	Final Presentations, finish final paper	Thomas

Table 4. Schedule

7.3 Appendix C - Requirements & Verification Tables

Vehicle Power

Requirement	Verification
• Alkaline Battery must maintain at least 8V voltage to meet operational requirements for 30 minutes	• Charge test ○ Connect fully charged battery to a resistor circuit ○ Discharge battery using resistor circuit for 0.5 hours ○ Use voltmeter to measure final battery voltage
• Lithium Battery must maintain at least 8V voltage to meet operational requirements for 30 minutes	• Charge test ○ Connect fully charged battery to a resistor circuit ○ Discharge battery using resistor circuit for 0.5 hours ○ Use voltmeter to measure final battery voltage

Justification
Alkaline Battery - Connected fully charged 9V battery as power for servo circuit - Continuously powered the servo as it alternated between its raised and lowered configurations for 30 minutes - Final voltage: 8.9V ± 0.1V Lithium Battery - Connected fully charged 9V battery as power for one motor circuit - Continuously powered the motor as it rotated at drive speed for 30 minutes - Final voltage: 8.7V ± 0.1V

Voltage Regulators

Requirement	Verification
● Provide component-specific output voltage (see above) reliably from a 9V source	● Reliable voltage regulation test ○ Setup voltage supply of 9V from either a voltage generator or a battery input ○ Measure output voltage and current using a multimeter to be 9V and 50mAh respectively

Justification

Voltage Regulators (5V)
- Tested regulators in with 9V supply before integration into modules
- Powered the 3 project modules with 9V source
  - Controller
  - Node vehicle
  - Lead vehicle
- Measured voltage: 5V ±0.04V
 	
Voltage Regulators (3.3V)
- Tested regulators in with 9V supply before integration into modules
- Powered the 3 project modules with 9V source
  - Controller
  - Node vehicle
  - Lead vehicle
- Measured voltage: 3.3V ±0.02V

USB-to-UART Chip

Requirement	Verification
● Transmit data between UART and USB at 9.6 kbps	● USB-UART testing ○ Use a socket connection to connect a computer to the MCU ○ Use Arduino online interface to upload processing code to the MCU bootloader ○ Test connection reliability for different data transfer rates ○ Ensure data rate of at least 9.6kbps is achievable

Justification (Modified)

Modification:
- Errors in our PCB design rendered our UART/USB connection inoperable.
- Achieved data transmission to the MCU for programming in the following way:
  - Switched from ATMega2560 to ATMega328 MCUs pre-bootloaded with Arduino interface capabilities
  - Fit MCUs into Arduino board's pin headers
  - Programmed MCUs through Arduino board, then transferred MCUs to project modules
  - Ran test programs to control LEDs, servos, and motors with MCUs, confirming successful data transmission

Microcontroller Unit (MCU)

Requirement	Verification
● Can receive and transmit data from Serial interface at a minimum rate of 256 kbps	● SPI testing ○ Connect MCU to Serial interface using FT232 chip to a terminal interface ○ Start timer and send data packet of specified size ○ Echo data back from the MCU ○ Stop timer and calculate data transfer rate and ensure at least 256 kbps
● Can output PWM of duty cycle 50% signals to control motors for to navigation and latch control	● PWM testing ○ Connect MCU general output pin to breadboard ○ Make breadboard connection to oscilloscope ○ Note PWM duty cycle range to be at least functioning at 50%

Justification
MCU Data Transfer Rate - Loaded 2 MCUs with SPI test code which passed data from one MCU's transmission port to the other's receiving port and back again - Observed a >2MB/s data transfer rate MCU PWM Output - Attached a "Servo" class output to an MCU pin and swept up from servo minimum to maximum values - Observed pin output on oscilloscope - Observed PWM output over range: 10% - 90% - MCU PWMs properly control the project servo and motors

XBee Radio

Requirement	Verification
● Maintain baud rate of at least 100 kbps	● Baud rate testing ○ Connect XBee module to XBee Explorer and open X-CTU interface ○ Use X-CTU interface to configure network and send random data packets of different data sizes ○ Use another XBee Explorer to collect data from network and calculate baud rate ○ Test for different baud rates for both receiver and transmitter and ensure measurement of at least 100kbps
● Each module should maintain a transmission range of at least 60 feet in outdoor settings	● Range testing ○ Connect Xbee module to XBee Explorer and open X-CTU interface ○ Establish connection with another XBee module to receive data packets ○ Measure at distances of 100, 200, 300, 400 and 500 feet (and at least 60 feet is functional)

Justification
XBee Baud Rate - Programmed MCUs with specified 9600 baud rate (standard serial output for ATMega328) - Successfully passed packets back and forth between MCUs at >2MB/s, verifying successful communication at given baud rate Transmission Range - Programmed MCUs with code using LEDs to verify established connections with other MCUs - Established connections between MCUs via XBee modules - Walked individual modules down the full length of the north ECEB hallway (outside Rm 2070), greater than 60 feet

Vehicle Software

Requirement	Verification
● Receive and acknowledge instructions received through the radio module ● Can correctly interpret instructions ● Can facilitate the re-transmission of instructions from node to node	● Connection testing ○ LEDs on the controller will be used to give a visual indication that a node is receiving and acknowledging instructions ● Instruction testing ○ Establish an acknowledged connection between the controller and a node ○ Send instructions to change the configuration of the latch and to drive the wheels ● Daisy-chain software testing ○ LEDs on the controller will be used as a visual indication that a node is receiving and acknowledging instructions ○ One node N1 will be placed inside the range of the controller, another N2 outside the range of the controller ○ If the indicator shows that N2 is receiving and acknowledging instructions, this will show that N1's software is successfully facilitating a re-transmission

Justification
Acknowledge Messages - Programmed MCU with code receiving messages through serial port - Verified by flashing onboard LEDs with reception of message Message Interpretation - Set up MCU to receive messages in the above way - Encoded instructions in messages - Programmed MCUs to move different motors depending interpreted instructions - Verified that correct motors moved as instructed Message Re-Transmission - Set up 3 MCUs connected to each other via 3 respective XBee modules - Used flashing LEDs driven by the MCUs to verify that messages were passed from node 1 (the first MCU/XBee group) to node 2, the middle node, finally to node 3, then back again in the opposite order, repeating indefinitely

Wheel Motors

Requirement	Verification
● The torque requirements for the latch servo is at least a minimum of 1 kg/cm to propel the node segment	● Torque requirements ○ Connect motors to voltage supply ○ Attach drive gear to the end of the motor ○ Connect latch sprockets and latch physical design to the gear ○ Observe torque and latch acceleration for different input voltages/current for the motor and to ensure we have torque of at least 1 kg/cm
● Temperature of the motors does not exceed 60°C in persistent (>2 minute) duty cycle changes.	● Temperature test ○ Connect servo motors to MCU output pin ○ Create different voltage inputs using serial signal library for the MCU ○ Use IR thermometer to measure temperature of the motor casing and ensure measurement doesnt go above 60°C

Justification
Torque Range - We were able to successfully drive the node vehicle using the motors - The 3D printed wheels have a slippery outer layer which makes the vehicle gain less traction - We are able to implement turning using variable voltage output from both the motors Heat Tolerance - We ran the wheel motors, with no load, at ESC control of 1600 (our operational top speed) for 2 minutes - We measured the wheel motor temperature using an IR thermometer - Final temperature: 32.3 ±0.1°C

Latch Motor

Requirement	Verification
● The torque requirements for the latch servo is at least a minimum of 1 kg/cm to support the weight of the 3D printed latch.	● Torque requirements ○ Connect servo motors to voltage supply ○ Attach drive gear to the servo end of the motor ○ Connect latch sprockets and latch physical design to the gear ○ Observe torque and latch acceleration for different input voltages/current for the motor
● Temperature of the servo motors does not exceed 60°C in persistent (>2 minute) duty cycle changes.	● Servo temperature test ○ Connect servo motors to MCU PWM pin ○ Create different duty cycle using serial signal library for the MCU ○ Use IR thermometer to measure temperature of the motor casing

Justification
Torque Range - The latch is able to lift a hook weight - Although less than the originally perceived torque requirement, the latch required less torque Heat Tolerance - Similar to the motor heat tolerance, we performed switched the latches configuration from raised to lowered repeatedly for 2 minutes - We measured the latch motor temperature using an IR thermometer - Final temperature: 28.4 ±0.1°C

Controller Power

Requirement	Verification
● Alkaline Battery must store at least 50 mAh to meet operational time requirement of 30 minutes	● Charge test ○ Connect fully charged battery to a resistor circuit ○ Discharge battery using resistor circuit for .5 hours ○ Use voltmeter to measure and ensure battery voltage and ensure charge of at least 50mAh

Justification
Capacity Assessment - A discharging resistor circuit is used to calculate the charge capacity of the Alkaline batteries - Using the formula, R_min ~= (Cells_in_battery x 4000) / Charge_capacity, we get a resistor 80 ohm

Controller Software

Requirement	Verification
● Software must be able to transmit package instructions correctly based on button input	● Packet fidelity testing ○ Establish a connection between the controller-mounted XBee radio and a radio module inserted in a computer's serial port ○ Motor packets received and ensure bits properly align with what should be transmitted

Justification
Correct Instructions from Buttons - Programmed controller MCU with software that: - Reads button inputs - Structures instructions based on inputs - Transmits messages - Verified correct messages encoding structure through Arduino serial monitor

Input Buttons/Switches

Requirement	Verification
● Buttons must be easily pressable ● Switches must reliably switch between on and off	● Button Test ○ Press button and ensure ease in pressing down ● Switch Test ○ Connect switches to a breadboard ○ Attach voltage supply across the switches ○ Measure current using a multimeter in the on-off states of the switch

Justification
Buttons Usability - Verified user could easily press and release buttons Switch Reliability - Programmed MCU with software reading input from a given pin - Designed small circuit with switch either opening or closing a connection to HIGH - Verified correct switch functionality through input visualized in Arduino serial monitor

LEDs

Requirement	Verification
● Must be visible from 1 meter away with a suitable drive current and voltage	● LED visibility test ○ Supply 1V input voltage using voltage supply ○ Test visibility from 1m away

Justification
LED Visibility - Designed a simple circuit using an LED, resistor, and voltage source to power the LED - Verified that LEDs were sufficiently bright so as to be seen at a distance