BitDrones: Design of a Tangible Drone Swarm as a Programmable Matter Interface

By

John Calvin Rubens

A thesis submitted to the Graduate Program in Electrical and Computer Engineering in conformity with the requirements for the Degree of Master of Applied Science

Queen's University

Kingston, Ontario, Canada

September 2019

Abstract

This thesis details the design and functionality of BitDrones, a programmable matter interface (PMI) composed of micro cuboid drones. Each self-levitating BitDrone represented a tangible voxel and possessed four degrees of freedom to position itself in three dimensions and control its orientation about the *z*-axis. A single BitDrone consisted of a custom designed quadcopter suspended inside a carbon fiber wireframe cube with onboard RGB LEDs for illumination. As a swarm, these drones formed a human-computer interface that could be physically manipulated via manual user input as a Real Reality Interface (RRI). RRIs render interactive digital experiences via the manipulation of physical matter but faced several functional limitations until now. Historically, RRI elements were not self-levitating and could only create structurally stable models, or such elements had self-levitating capabilities but were not tangible and could only create sparse three-dimensional models. The spatial independence, tangibility and self-motility of each BitDrone voxel granted more versatility than previous RRIs, and this versatility enabled BitDrones to meet the functional criteria of a PMI as defined in the Human-Computer Interaction (HCI) literature. This work presents the evolving design of the BitDrones, the key components of the computational architecture that governed the system, as well as the tangible interactions, tools and controllers designed to use the BitDrones display. Several academic explorations over the development timeline are also presented which were completed as the BitDrones system matured.

Acknowledgements

BitDrones was an ambitious undertaking and involved the contributions of several individuals. The initial inspiration for a self-levitating programmable matter interface was conceived by Dr. Roel Vertegaal, the Director of the Human Media Lab at Queen's University. The subsequent design rationale of the system was a product of the collaboration between the author, Dr. Vertegaal and Sean Braley (M.Sc.). Dr. Antonio Gomes, was the guiding figure for the publication *BitDrones: Towards Using 3D Nanocopter Displays as Interactive Self-Levitating Programmable Matter* [1] and was the primary designer of the BitDrone V3 model. Dr. Timothy Merritt, an associate professor of computing at Aalborg University in Denmark was a major collaborator in the conceptualization of the BitDrone LEGO® interactive experience.

Table of Contents

List of Figures

List of Abbreviations

Acronym *Expanded Name*

ABS	*Acrylonitrile Butadiene Styrene*
AP	*Access Point*
APF	*Acoustic-Potential Field*
CAD	*Computer Aided Design*
DOF	*Degrees of Freedom*
DPDT	*Double-Pole, Double-Throw*
DSM	*Digital Spectrum Modulation*
ESC	*Electronic Speed Controller*
FC	*Flight Controller*
FOLED	*Flexible Organic Light-Emitting Diode*
FRP	*Fiberglass Reinforced Plastic*
GUI	*Graphical User Interface*
HDI	*Human-Drone Interaction*
IC	*Integrated Circuit*
ID	*Identifier*
IEEE	*Institute of Electrical and Electronics Engineers*
IMU	*Inertial Measurement Unit*
INS	*Inertial Navigation System*
I/O	*Input/Output*
IP	*Internet Protocol*
IR	*Infrared*
LAN	*Local Area Network*
LED	*Light-Emitting Diode*
LLC	*Logic Level Converter*
MCU	*Microcontroller Unit*
MDF	*Medium-Density Fiberboard*
MIPI	*Mobile Industry Processor Interface*
MOSFET	*Metal-Oxide Semiconductor Field-Effect Transistor*
MSP	*MultiWii Serial Protocol*
MW	*MultiWii*
MWC	*MultiWii Controller*
NUC	*Next Unit of Computing*
OLED	*Organic Light-Emitting Diode*
OS	*Operating System*
OUI	*Organic User Interface*
PC	*Personal Computer*
PCB	*Printed Circuit Board*
PID	*Proportional, Integral, Derivative*
PMI	*Programmable Matter Interface*
PWM	*Pulse Width Modulation*
RAM	*Random Access Memory*
RF	*Radio Frequency*
RGB	*Red, Green, Blue*
RPM	*Revolutions Per Minute*
RRI	*Real Reality Interface*
SLS	*Selective Laser Sintering*
SoC	*System on Chip*
SP	*Setpoint*
SPI	*Serial Peripheral Interface*
SSID	*Service Set Identifier*
TUI	*Tangible User Interface*
UAV	*Unmanned Aerial Vehicle*
UDP	*User Datagram Protocol*
UI	*User Interface*
USB	*Universal Serial Bus*
UWB	*Ultra-Wideband*
VR	*Virtual Reality*
WLAN	*Wireless Area Network*
XOR	*Exclusive OR*

List of Equations

Chapter 1. Introduction

BitDrones is a drone-based programmable matter interface (PMI) composed of tangible micro quadcopters. The inspiration for BitDrones was based upon the theoretical concept of programmable matter: artificial matter that can alter its physical configuration and properties. It is also a real reality interface (RRI) that leverages the nature of real matter (the drones) to display physical structures and accommodate tangible user interactions. The major contributions of this thesis are the introduction and design of tangible drones as constituents of programmable matter, the application of both familiar and novel interaction techniques to manipulate programmable matter, and finally to provide inspiration and a detailed reference for future drone-based interfaces. As of the completion of this thesis, BitDrones represents the most capable 3D prototype of a PMI due to the drones' superior motility and self-levitating qualities over previous 2D prototypes.

First, prior research in the areas of programmable matter, tangible user interfaces, organic user interfaces, self-levitating real reality interfaces, swarm displays and human-drone interaction are examined. Next, the design of the system is rationalized based on definitive programmable matter literature. An overview of BitDrones follows, introducing the essential hardware, firmware, software, interaction devices and how they function together. BitDrones represents the product of several years of development and comprises multiple explorations, each subsequently improving on certain aspects of the system and/or introducing new features. These explorations are introduced chronologically and focus on interaction techniques for manipulation and modelling of the BitDrone swarm, technological improvements, as well as overall refinement of the user experience. Lastly, recommendations are provided for future work on drone-based interfaces and programmable matter systems.

1.1 Collaboration

Interaction techniques were designed collaboratively, where the author designed the mathematical and algorithmic processes of the techniques which were then implemented and further refined by Sean Braley (M.Sc.) in BitDrones OS. Sean Braley was also the creator of the system's network architecture as well as of the designer of BitDrones OS; the software which governed the BitDrone display. The control systems were developed, tuned and refined collaboratively between the author and Sean Braley. The author solely contributed the catomic and programmable matter design rationale, BitDrone designs (save for Bitdrone V3), input device designs, drone and device firmware, distributed routing techniques, *DroneZ* and *DroneForm*.

Chapter 2. Related Work

The development of BitDrones was closely related to prior work in several other areas of research. These areas are Tangible User Interfaces (TUIs), Organic User Interfaces (OUIs), self-levitating interfaces, swarm displays and Human-Drone Interaction (HDI).

2.1 Programmable Matter

In 1965, Ivan Sutherland first conceptualized a display in which a physical computing interface would fully embody digital information in *The Ultimate Display*, which could "control the existence of matter" [2]. He imagined this display as a window into the digital construct of a computer's memory, and as such should translate this information to be comprehendible to the user. As we understand the world around us using our senses, Sutherland proposed that the Ultimate Display should also "serve as many senses as possible", going so far as to suggest a display capable of generating taste and smell. More realistically, he described a full-body kinesthetic display with haptic feedback and gaze detection to augment more traditional audio/visual computing experiences. In his words "The ultimate display would, of course, be a room within which the computer can control the existence of matter. A chair displayed in such a room would be good enough to sit in. Handcuffs displayed in such a room would be confining, and a bullet displayed in such a room would be fatal" [2].

Toffoli and Margolus coined the term programmable matter to describe the conceptual architecture of a scalable cellular automata machine, "CAM-8" [3]. In such an interface, programmable matter could assemble into clusters of arbitrary size with arbitrary material properties that would not necessarily be constrained by the traditional laws of physics. This work was the first definitive outline for the functional requirements of a programmable matter interface. Envisioned as a "uniformly textured, fine-grained computing medium", it would have the following properties:

1. It can be assembled into arrangements of arbitrary size, governed not by technological limitations, but by economic ones.
2. It can be dynamically reconfigured into any uniform parallel computing network.
3. It can be interactively driven via internal or external events.
4. It can be observed, analyzed and modified in real-time.

The definition of programmable matter was further refined through Goldstein and Mowry's *Claytronics*, in which the functional criteria of programmable matter elements were defined [4]. Claytronics was envisioned as a subtype of programmable matter that would mimic shape and surface to match the visual appearance of an object utilizing claytonic atoms or "catoms". Four functional requirements were also defined for such catoms.

1. Each catom should be self-sufficient in computation, communication, sensing, actuation, locomotion and adhesion.
2. To avoid excessive heat and power requirements, no static power should be required for adhesion after attachment.
3. Coordination of catoms should be performed via local control, i.e. with no external computation.
4. In the interest of reliability and economic viability, catoms should contain no moving parts.

This initial foray into programmable matter led to the development of prototype catoms linked with electromagnets that could form interconnected 2D networks. Further works by Goldstein et al. focused on the improvement of subsequent catom prototypes in adherence to the ensemble principle, where "a robot module should include only enough functionality to contribute to the ensemble's desired functionality" [5].

More recently, Ishii et al. introduced *Radical Atoms* [6]. TUIs simply provide a tangible interface with which to manipulate data, while *Radical Atoms* described the evolution of TUIs into hypothetical dynamic materials which would serve as physical manifestations of digital information. A storyboard exploration of *Radical Atoms* resulted in the conceptualization of *Perfect Red*, a clay-like substance that could be programmed to possess features of modern-day computer-aided design (CAD) software, such as interpolating three dimensional shapes and snapping to geometric points of interest.

Kinetic Blocks was a functional take on programmable matter that utilized an actuated tabletop display to create structures from simple passive building blocks [7]. The tabletop consisted of a two-dimensional array of pin elements that could be actuated in the *z*-dimension. By exerting vertical forces through the pins, the display could roll, lift and even throw the passive construction cubes. Structures could be created by appropriately positioning the cubes, which would link together via self-orienting spherical magnets positioned on each face.

M-Blocks presented the design of modular self-assembling robotic cubes capable of independent locomotion [8]. Their design incorporated a flywheel which could be quickly spun and stopped, taking advantage of the flywheel's inertia to generate torque about one of the cube's principal axes. This flywheel could be dynamically reoriented inside each M-Block allowing for torque generation about different axes, granting the cubes the ability to roll and even jump. M-Blocks also incorporated passive magnets on each face for physical connection, enabling them to form lattice structures. Their self-motility and magnetic adhesion meant that they could climb on and move across structures made from other M-Blocks. One significant drawback of this prototype was that the resulting architecture had to be structurally stable and it could not accommodate for any physical disturbances that jeopardized this stability.

True self-contained catomic elements [4] are exceptionally difficult to create with present-day technology, so some programmable matter implementations have focused on dynamic 3D

printing. A recent example of this trend is *DynaBlock*, which could reconstruct true three-dimensional models utilizing a 16 pin-actuated display as an assembly mechanism [9]. Display elements consisted of small plastic cubes with one disk magnet on the top and bottom faces, and four spherical self-orienting magnets in the side faces. *DynaBlock* is the first dynamic 3D display capable of autonomously assembling and disassembling full three-dimensional tangible structures.

2.2 Tangible User Interfaces (TUIs)

TUIs are an even more recent concept than that of programmable matter. First introduced as Graspable User Interfaces in *Bricks* by Fitzmaurice et al. [10], they allow for the manipulation of electronic or virtual objects through physical control handles. This preliminary graspable UI possessed control handles represented by two small bricks that were used to control a drawing application projected onto a tabletop. With two control points, *Bricks* could be manipulated bimanually to increase the degree of control during specific drawing tasks, e.g. simultaneously controlling position, orientation and scaling of a curve. The term “Tangible User Interface” was definitively coined in *Tangible Bits*, which expanded on the concepts introduced in *Bricks.* This subsequent work sought a variety of ways to turn physical matter into “interfaces between people and digital information” [11].

Early TUI explorations were limited to utilizing just one or two tangible interaction objects at a time. Underkoffler and Ishii [12] created an urban planning system, *URP*, that aided in the visualization of issues such as shadows, reflections and wind in the design process. “Luminous-tangible interactions” were defined as when an interface provides information and functionality to physical objects via projected visuals. *URP* enabled luminous-tangible interactions on a projected surface coupled to physical building models which could be dynamically positioned within the interaction space. Reflections, wind and other environmental phenomenon were projected onto the physical scene in order to visualize their effects at scale.

As more complex sensing and imaging technology became available, it became practical to not only use tangibles as input devices to a system, but for the system to provide bidirectional feedback through the tangibles. In the case of *Illuminating Clay*, surface geometries of a physical landscape model could be recorded with a laser or infrared scanner, analyzed, and then projected upon to visually convey the outcome of the computed analysis [13]. In this design, sand or clay was used as the tangible sculpting medium for the landscape model, while scanner data was used to compute a matrix of surface elevations. The analysis interpolated surface elevations from the measured elevation points, which were then colour mapped and projected back upon the sculpting medium to more easily visualize topological qualities.

TUI elements later gained user-alterable parameters to increase and even change their functionality. *ReacTable* consisted of tangible objects on a circular rear-projected interaction surface and was a research effort to unify musical instruments and tabletop interfaces [14]. These tangible interaction objects represented different functional components of electronic instruments such as signal generators, mixers, filters, etc. Parameters of these functional components could be altered via a projected touch interface that radiated outwards from the components.

Patten and Ishii [15] combined the popular projected tabletop interface with kinetic tangible elements to create *Pico*. The research aimed to explore optimization tasks through tactile feedback and physical constraints when paired with the abstraction available via modern computing. This unique TUI was among the first to provide visual as well as physical feedback. It was designed with an electromagnet array underneath the tabletop which could move *Pico* objects through the coordinated activation of the electromagnetic elements. Users could manually move *Pico* objects but they could also constrain the attempted movement of *Pico* objects by the system, manually or with other physical objects like flexible barriers. Further

visual information was linked to each tangible object and displayed via projection onto the tabletop around and between the objects.

The use of purpose-made tangible interaction objects to embody non-tangible information remains a limitation of current TUIs. *MirageTable* allowed for interactions with real physical objects from the user's environment by scanning and replicating them in real-time alongside virtual objects [16]. The visual and physical qualities of scanned objects were virtually duplicated through a stereoscopic 3D display positioned in front of the user. This system afforded natural interaction modalities due to the user's manipulation of the real objects as input.

2.3 Organic User Interfaces (OUIs)

TUI interaction generally occurs in two dimensions or on a surface, while an OUI is defined by Holman and Vertegaal in *Organic User Interfaces* as a computer interface with three distinct properties [17]*:*

1. Employs a non-planar display for output and input.
2. Possesses the ability to "become" the data through deformation via manipulation or actuation.
3. Graphics are shaped through multi-touch and bimanual gestures.

Project FEELEX was an early OUI that deployed a device to add surface haptics to projected graphics [18]. The haptic display consisted of a two-dimension array of actuators that could move up and down linearly along the *z*-axis. A flexible screen stretched over the actuator array allowed simultaneous display from a projector suspended overhead, as well as for deformation via the actuators to convey haptic information that corresponded to the visual display.

Raffle et al. [19] first implemented bidirectional tactile sensing and kinetic feedback in an OUI through *Topobo*, a 3D constructive system composed of passive and active components. Models created using a combination of these passive components and motorized networkable

active components could retain kinetic memory of their movement. Essentially these models could be programmed by example, where the kinetic information could then be replayed to animate the model.

Lumen was an interactive tangible display similar to *Project FEELEX* that was capable of bi-directional shape-change as well as displaying visual images. The display was composed of a two-dimensional array of twenty-five illuminated cylinders. These cylinders could be actuated in the *z*-dimension by the system to create sparse three-dimensional surfaces. The illumination of the cylinders afforded extra dimensionality to the system, enabling further visualizations on the rendered surface.

2.5D displays are displays that are only capable of forming 3D structures with no cantilevered features. This means that 3D structures displayed on 2.5D displays must be structurally stable because the individual elements are not self-supporting. Blackshaw et al. [20] introduced *Relief* as a 2.5D tabletop display similar to *Lumen* but with a sizeable increase in display resolution, utilizing 120 motorized aluminum pins instead of 25 illuminated cylinders. The goal of *Relief* was to create a platform suitable for a wide variety of applications and interactions through this functional resolution increase, as well as through the application of low-cost, low-complexity components. Due to the physical size of the supporting hardware in the tabletop, the display was relatively sparse, with the pin elements spaced about 1.5 inches apart. “Relief” is a modelling technique in which architectural features share a common anchoring plane and as such, many subsequent 2.5D displays after the introduction of *Relief* are referred to as relief displays.

Blackshaw et al. created *Recompose*, which built upon *Relief* through the addition of a projector to display visual information on the surface of the haptic display and a depth camera to detect gestural actions from a user [21]. To form a projection surface, a square tile was affixed atop each of the 120 actuated pins. This also had the effect of decreasing the physical sparseness of

the display elements while maintaining information density. *Recompose* retained the same tangible input and output abilities of *Relief* but was designed primarily to explore the combination of tangible and gestural interactions. Blackshaw et al. posed that it was difficult to directly manipulate large areas of an OUI due to physical constraints of the body. Ultimately through this work, they showed that additional gestural interactions allowed for functional manipulation of larger areas of a display.

Likely the most famous OUI created to date has been the *inFORM* Dynamic Shape Display, which continued the trend of actuated pin driven interfaces and set a performance standard for future OUIs [22]. The goals of *inFORM* were threefold:

1. To facilitate and explore interactions through “Dynamic Physical Affordances”, such as dynamic physical buttons and handles generated by the display for the user.
2. To introduce dynamic physical constraints to guide interaction and limit possibilities within the *inFORM* ecosystem.
3. To manipulate and augment passive objects such as balls or bricks via actuation through the shape display.

The display was augmented with projected graphics from a suspended projector overhead, while a Microsoft Kinect® depth camera tracked objects and user interaction. The complex demands outlined by the defined goals required near real-time actuation of the pins and a much higher functional resolution of 900 pins in a 30x30 two-dimensional grid. *InFORM*’s successful implementation resulted in a more practical and useable OUI finally capable of serving as a general-purpose shape-changing interface.

The most prominent issue with *inFORM* was the limited size of its operational environment, but a recent research project called *shapeShift* has freed *inFORM* from its stationary operating limitations by combining several technologies [23]. It is a pin-actuated shape-changing display

like many OUIs before it, but simultaneously uses a smaller number of pins (288 vs. 900) and increases both the visual and tangible operating area. To increase the visual operation area, *shapeShift* uses a virtual reality headset instead of an immobile projector to visually immerse the user in a virtual three-dimensional environment. To broaden the tangible operating space, the shape-changing displays are mounted on robotic platforms with omni-directional wheels, which allow for movement along any two-dimensional vector in any orientation.

A motion capture system unified the visually rendered scene in virtual reality with the haptically rendered scene in real reality, while the hands were also tracked to determine the necessary positions of the mobile haptic displays. This means that only one haptic display is needed for unimanual interaction with *shapeShift*, or two displays for bimanual interaction. It also was possible to exert lateral forces for the first time with a vertically actuated pin display through the omnidirectional drive system. The convergence of virtual reality with tangible robotic augmentation made for the possibility of a room-scale OUI vs. table-limited interactions.

Even more novel OUIs such as *Tangible Drops* have been developed; a two-dimensional visio-tactile display that actuated liquid-metal droplets [24]. As a new display type, it aimed to develop display primitives such as droplet movement, oscillation and more novel aspects such as droplet splitting and merging. Creation of new OUI interaction devices that utilize real matter as display elements bring us closer to the development of true programmable matter systems.

2.4 Self-Levitating Real Reality Interfaces

2.5 dimensionality is a problem even in the most advanced OUIs due to the fact that 3D structures rendered on 2.5D displays must be solid, cannot possesses cantilevered features, and must be anchored to the display base. These restrictions exist because the individual display elements are not self-supporting. Self-Levitating Interfaces combat this issue by utilizing independent levitating display elements.

Karagozler et al. [25] first implemented modular, self-levitating robots to enable remote construction of distant structures in space and on extraterrestrial bodies. These robotic constructive elements were built on the basis of catomic elements described in Goldstein and Mowry's *Claytronics* [4]. They satisfied the catomic criteria for self-sufficiency in terms of their own actuation, adhesion, control and power requirements, enabling them to function according to the ensemble principle [5].

Each catomic element was built around a helium filled cube structure to mimic the low gravity that would be encountered in orbit and on other planetary bodies. These cube-shaped robots each possessed 24 flaps (4 per cube face) actuated by shape memory alloy to provide mobility. Electrostatic forces generated on the surface of the flaps enabled them to adhere to flaps on other robotic cubes. Through actuation of these flaps, they could be made to "snap" to other flaps with the electrostatic forces generated. After adhesion, the cubes could actuate themselves relative to their anchoring cube while linked via the charged flaps.

Lee et al. [26] created *ZeroN,* a contactless mid-air tangible display via computationally controlled magnetic levitation of a spherical permanent magnet. The levitated magnetic display element served as a tangible landmark to a virtual three-dimensional feature and was capable of both input and output. The display consisted of an electromagnet mounted on a linear actuator perpendicular to the *z*-axis which was positioned in 2D via a linear gantry. The electromagnet could control three-dimensional motion and levitation of the display element, however it had a limited functional volume. Through the combination of linear actuation, this smaller workspace was extended over the desired operating volume of the whole *ZeroN* interaction space using linear actuators. A set of IR stereo cameras triangulated the display element's position for feedback control, while a Microsoft Kinect depth camera was used to detect users' interactions. A projector was also utilized to visually augment the scene created within the interaction space, which consisted not only of the tangible dynamic display element,

but also passive interaction objects. *ZeroN* was the first levitating display whose elements were physically untethered.

Other approaches to levitation relied on acoustic waves as was the case in *Pixie Dust*, a system that was capable of manipulating particles of varying materials to form a tangible levitating matter interface [27]. The motivation for the creation of the system was to utilize real matter as graphical components. In prior acoustic levitation systems, particles and small objects could be suspended in the standing waves of acoustic signals. *Pixie Dust* created acoustic-potential fields (APFs) that allowed acoustically suspended objects to be repositioned through the manipulation of these waves. APFs granted static control that allowed real matter to be used as physical pixels, as well as dynamic control that allowed real matter to be used to display vector graphics. *Pixie Dust* was the first interface capable of rendering multiple levitating display elements composed of real matter.

2.5 Swarm Displays

Swarm displays consist of independently actuated robotic elements that perform in accordance to the ensemble principle [5] to physically render two or three-dimensional models. In 2011, Alonso-Mora et al. devised a system for pattern formation that included collision-free trajectory planning and accounted for varying kinematics in robots that employed wheeled differential drive systems [28]. Essentially, this was the first functionally detailed swarm robotics graphics engine. One year later in 2012, this engine was refined and applied to varying swarms of up to fifty independent illuminated robotic elements to form sparse two-dimensional structures [29].

Many prior efforts towards swarm displays focused on swarm algorithms and decentralized control. Due to the cost and complexity of managing large numbers of robots, research would usually be validated through simulation. *Kilobot* enabled researchers to deploy physical swarms using diminutive, low-cost and easy-to-assemble robotic elements called “Kilobots” [30].

Individual Kilobots were capable of independent locomotion using two vibrational motors, ranging and communicating between each other, and had onboard power and computational capabilities. Algorithmically, they were able to determine their location relative to a universal "seed" Kilobot through shortest path optimization of distances between other robots. Initially only a swarm of 25 was demonstrated, but the system was soon scaled to 1024 Kilobots to reflect their namesake [31]. This subsequent iteration utilized the same technologies and techniques introduced in *Kilobot,* and successfully demonstrated a massively parallel swarm robotic display capable of rendering much denser two-dimensional structures than previously possible.

While their creation was an incredible technological achievement, Kilobots were not capable of any sort of user interaction. Though they were designed for ease-of-use by swarm robotics researchers, they were not designed with "users" in mind in terms of an interface. *Zooids* introduced the first tangible swarm user interface [32], composed of small autonomous robots similar in size and form to Kilobots and designed with the vision of *The Ultimate Display* [2] and *Radical Atoms* [6]. The robots themselves differed in that they used two wheels for locomotion, received coordination commands from a centralized control node and localized themselves through a structured light pattern from an overhead projector.

The system enabled physical rendering, tangible bidirectional feedback and even had the ability to interact with surrounding objects. Shape interpolation enabled user creation of simple two-dimensional shapes from just two tangible control points similar to Fitzmaurice et al. [10], as well as spatial operations such as translation and scaling. *Zooids* also introduced interactive swarm rendering of time-series data, with Zooid elements rendering the two-dimensional data and two additional Zooids serving as control widgets to specify the width of the time-series window.

Another aspect of swarm UIs that has only recently been addressed is that of programming such an interface through the interface itself. *Reactile* introduced a high-level, tangible programming framework in place of traditional programming done on computers [33]. The

interface consisted of a projected tabletop with an embedded two-dimensional array of electromagnetic coils that actuated passive tangible markers. Similar to the tabletop system *Pico* [15], the markers could be moved around the space as output by the system, or as input from the user. Their positions were tracked via computer vision from a camera suspended overhead, from which the system could interpolate shapes and visualize data similarily to *Zooids* [32]. *Reactile* created a programming workflow for a swarm interface that allowed for the creation of shapes as basic swarm UI elements, abstraction of object attributes (e.g. size, shape, position, etc.), specification of object behavior, and propagation of user changes.

Swarm UI research has become practical due to increasing technological capabilities along with decreasing component costs, but it is mostly limited to two-dimensional development. This is purely due to the fact that levitating physical matter is much more complicated. Inspiration for a three-dimensional swarm display appeared in 2010 with the debut of *Flyfire*, a computer-generated glimpse into the functionality of a self-levitating, three-dimensional swarm display [34]. Developed through a collaboration between MIT's SENSEable City Lab and ARES Lab, the interface consisted of illuminated coaxial rotor drones, where each drone represented a "smart" pixel in the display area.

In 2016 drone swarm displays truly became a reality through Intel's *Drone 100* project [35]. As the name suggests, this system was composed of 100 independent, self-illuminating *Shooting Star* quadcopter drones. Within the same year, this quickly became *Drone 500* in a record-breaking attempt to fly the most unmanned aerial vehicles (UAVs) simultaneously [36]. This record was broken a third time in 2018 in celebration of Intel's 50th anniversary with a total of 2018 airborne drones [37]. Intel's system was deployed at a number of events since *Drone 100*'s introduction [38, 39, 40], however it was only suitable for outdoor use in exceptionally large operating volumes. The physical size of the drones and the thrust they produced also limited the vertical proximity attainable between each other.

Since the introduction of large-scale outdoor drone-based swarm displays, there have been a number of efforts tailored towards indoor use. Notably, *Crazyswarm* is an architecture for controlling a large swarm of drones indoors in relatively dense formations compared to Intel's *Shooting Star* drone displays [41]. The system is capable of flying 49 drones simultaneously and is based on *Crazyflie 2.0* miniature quadcopters and a motion capture system for positioning. It was not designed specifically as a swarm display, but rather to experimentally validate new control systems and techniques for swarm behavior. However, *Crazyswarm* demonstrated the construction and rotation of a three-dimensional pyramid with illuminated drones spaced 50 cm apart, proving that it was capable of rendering physical models.

Verity Studios is a recent company that creates ambient indoor swarm displays utilizing miniature quadcopter drones similar in form to *Crazyswarm* [42]. The feature that sets *Verity*'s system apart from others is it's positioning system, which uses a network of Ultra-Wideband (UWB) radios for multilateration. Utilizing UWB positioning hardware means that the system is much more portable than those requiring a vision-based motion capture system but is also much less accurate. For this reason, *Verity*'s offering should be considered an ambient drone display as there is not enough precision to form complex, stable three-dimensional structures.

All of the drone-based swarm displays presented until this point have only been able to fly in relatively sparse configurations due to turbulence, visual occlusion of tracked elements (obscuring them from a visual positioning system), or three-dimensional positioning inaccuracy of the display elements. *ModQuad* eliminated this sparsity by magnetically linking drones to form rigid, flying two-dimensional structures [43]. *ModQuad* was an engineering effort designed to assemble arbitrary structures in midair to solve collective tasks. This was achieved by altering the control system of each composite drone in a structure to accommodate for the dynamics of the newly constructed flying object, essentially treating them as a single drone.

2.6 Human-Drone Interaction (HDI)

Human-drone interaction is an emerging research area that has only been possible through recent technological developments which in turn have increased the accessibility to necessary hardware. Many areas of HDI involve interactions with a single drone to grant more intuitive control, or the integration of visual display technology into the drone itself to serve as a mobile display platform. Only within the past several years have research endeavors began to experiment with control of robot swarms through gestures and physical interactions.

Schneegass et al. [44] explored the utility of flying displays in their 2014 publication: *MidAir Displays*. It described the potential of free-floating public displays with the ability to provide personalized information to individuals or groups as necessitated in variable situations. A prototype was constructed consisting of an octocopter drone with an iPad® mounted underneath. It was envisioned that in the future, such technology could be deployed at sporting events, during emergency situations for crowd control and as a mobile information display when content needed to be shared collaboratively. *iSphere* [45] presented an implementation of a spherical drone display with better visibility than would be possible with the suspended 2D displays introduced in *MidAir Displays*. The prototype supported a spinning persistence of vision display ring around the drone. Its main benefits were high visibility due to the bright pixels used as well as the spherical shape of the display which gave it a viewing angle of 360 degrees around the drone.

Cauchard et al. [46] investigated gestural human-drone interactions for the command of a personal drone in *Drone and Me*. Personal drones still mainly utilize discrete traditional remote controllers, and their increasing pervasiveness necessitates the use of more accessible and ubiquitous methods for control. *Drone and Me* highlighted preferred gestural interactions for specific drone behaviour through a preliminary user study with several participants. *Drone Near Me* expanded on *Drone and Me* by creating a prototype “safe-to-touch” drone and performing a

similar study to evaluate preference between touch-based, gesture-based and sound-based interactions. The results of the study showed that twice the number of people felt comfortable interacting in close proximity with a caged drone than a regular drone.

Tactile Drones introduced drones as haptic devices to augment traditional VR experiences [47]. Many haptic devices and wearables for use in VR scenarios provide only localized stimulation, so drones were envisioned as a possible method to create more immersive tactile feedback. A prototype was created that placed the user in a virtual scene while an OptiTrack motion capture system tracked the user and a drone with a surrounding protective cage within a physical interaction space. A Microsoft Kinect was used to discern the location of the user's various body parts in the OptiTrack's frame of reference. The drone could then be guided to make physical contact with specific parts of the user's body according to visual cues from the virtual experience.

HoverBall was a more novel take on tactile drones and aimed to augmented sports with a flying ball [48]. The drone's frame mimicked a ball and was fully enclosed for physical interaction. With a robotic flying ball, it would be possible to modify the ball's physical responses and even add new laws to defy traditional physics. It was suggested that in future sports, handicaps could be applied or special abilities granted to players with respect to ball manipulation. *HoverBall* even proposed that new sports could one day be created that revolve around artificially imposed physical behavior of sport objects.

Augugliaro and D'Andrea first introduced a method to alter apparent inertia, damping and stiffness of tangible drones through *Admittance Control for Human-Quadrocopter Interaction* in 2015 [49]. Impedance control is a common method of robotic control for interaction with the physical environment, in which a robot's physical impedance is regulated via force feedback. Augugliaro and D'Andrea developed an admittance control layer within their position and attitude control system that governed drone movement in order to alter the way the drones

respond to physical forces applied by a user. Through this admittance control method, a control scheme that granted intuitive physical control of drones was demonstrated.

The first example of gesture-based swarm control was presented by Alonso-Mora et al. [50] with a 2D robotic system in a tracked interaction space. This prototype setup was similar to that described in previous works [28, 29] and consisted of differentially driven two-wheeled robots tracked by a motion capture system and a Microsoft Kinect to track a user's body. It showcased free-form manipulation for direct control of individual robots and sub-groupings, as well as shape-constrained manipulation methods where degrees of freedom of group were limited. A virtual ray cast from the user's index finger could be used to select one or more robots by altering the selection radius of the ray. Groupings could be scaled, rotated and repositioned gesturally within the interaction area. Even the colour of a single drone or group could be altered by mapping colours to users' hand height. Furthermore, shapes such as emoting faces and basic human skeletal models could be morphed through tracking the relative difference in locations on the user's body during gestures.

Chapter 3. Design Rationale

BitDrones was designed in the visionary image of *The Ultimate Display* [2] but more importantly it was designed with the functional capabilities necessary to explore elementary interactions with, and applications of, programmable matter. As discussed in *2.1 Programmable Matter*, four operative features were defined for programmable matter as well as four for its catomic constituents [5, 4]. In relation to this research, the Bitdrones system as a whole served as a primitive programmable matter interface, while an individual BitDrone functioned as a catomic element of programmable matter.

3.1 Catom Design

According to Goldstein and Mowry [5], catoms should contain no moving parts, require no static power for adhesion between catoms, be entirely self-sufficient in all senses and be coordinated via local control. These requirements were largely based on the concept of mass-produced catomic elements, those that would be robust and inexpensive enough to operate in programmable matter swarms composed of potentially millions or billions of catoms. If programmable matter scalability on that magnitude is not a priority, then the functional requirements of catoms change. Several key elements of programmable matter as an interface from a user standpoint are also missing from these preliminary requisites, which led to a new set of requirements for catoms.

3.1.1 Kinetic Components

Moving parts on/in a catom were specified against due to potential unreliability and added cost. This would be a hugely important design aspect for catomic elements in gross numbers, but it does not contribute in any way to the immediate functionality of a catom. Adhering to this design specification would have been prohibitively difficult with present technology and was not deemed worthwhile with respect to the end goals of the project. In order to satisfy other

necessary functionality, each BitDrone is equipped with four rotary motors with propellers for propulsion.

3.1.2 Passive Physical Connection

In order to reduce excess power requirements and heat buildup, catoms should not need static power for adhesion between each other. To increase endurance, power conservation would be essential, however this would not be very important for short-term operation. Heat buildup could also pose an endurance problem but may not if there is adequate cooling or if operation times are sufficiently brief. For these reasons, this requirement does not contribute to the immediate functionality of a catom and was rather considered a design recommendation. Ultimately to reduce complexity, the drones were designed to attach to each other via a passive array of self-orienting, spherical magnets which consume no static power.

3.1.3 Self-Sufficiency

In terms of individual capability, each BitDrone was equipped with WiFi for wireless communication, an inertial measurement unity (IMU) for orientation, a battery for power, as well as four motors and propellers for controllable flight. To obtain accurate real-time drone position information, a centralized vision-based motion capture system was used. To minimize latency as well as streamline the system for active development, it logically followed to centralize their coordination as well. Consequently, BitDrone catomic elements were not totally self-sufficient as they relied on external systems for localization, computation and coordination.

3.1.4 Local Coordination

Local coordination falls under the umbrella of self-sufficiency in terms of computation, so this was not considered a separate requirement and was handled centrally. Each drone had the capability to roughly navigate using its onboard IMU for a couple of seconds, however due to

sensor drift this is not a stand-alone solution. This limited navigational capability is only useful when fused with another positioning technique.

3.1.5 Sensory Stimulation

No prior theoretical definitions of catoms reflected the visceral nature of traditional matter. A key characteristic of matter in relation to a human-computer interface is that because it is *real* it is capable of stimulating all senses. Individual BitDrones were designed to appeal to users' haptic and visual perception, both of which are exploited most widely by traditional interfaces. A carbon fiber wireframe cage surrounding each BitDrone served as a graspable exterior for user interaction and as a visually defining boundary for a representative 3D shape. To facilitate manipulation, the drones had to be small enough to easily grasp with one hand. Coloured red, green, blue (RGB) light-emitting diodes (LEDs) provided illumination to convey further visual information. BitDrones' small size was also important in order to perceive them as individual volumetric pixels instead of individual drones.

3.1.6 Unconstrained Behaviour

Programmable matter structures should not be subject to environmental limitations unless otherwise specified, so catoms need the individual ability to overcome them and even be able to adhere to artificially imposed physical laws like in *HoverBall* [48]. It is unrealistic to neutralize every environmental factor affecting such a system, but the most restrictive and pervasive is gravity. Programmable matter unconstrained by the force of gravity would not require structural support and its catomic elements would have superior mobility over ground-based elements. Therefore, it was deemed the highest priority for the programmable matter interface to overcome gravity via self-levitating catomic elements. Due to small size requirements in combination with current technological limitations, rotorcraft were chosen as the scaffold for those elements.

3.1.7 New Catomic Criteria

Over the course of the Bitdrones project, hundreds of people interacted with the system informally at conferences as well as in a lab environment. Due to the immersivity of the interface and the unobtrusiveness of the supporting hardware, a common misconception from users was that the BitDrones themselves were an entirely self-sufficient and self-governing system. This indicated that it was not possible to tell without prior knowledge if any of the functional criteria of catoms were unmet. It was inferred that the functional requirements of catoms in a programmable matter interface should be defined mainly by users' perception. From the previously discussed areas of importance, the design requirements for catoms were revised. The new requirements were that:

1. Each catom should behave as a self-sufficient entity [3].
2. Catoms should strive to "serve as many senses as possible" [1].
3. Catomic elements should have the means to embody the properties of the programmable matter that it constitutes, regardless of environmental factors.

3.2 Programmable Matter Design

In review, programmable matter should have the ability to be assembled into arrangements of arbitrary size, be dynamically reconfigured into any parallel computing network, interactively driven internally or externally, as well as observed, analyzed and modified in real-time [2].

3.2.1 Arbitrary Arrangements

It was stipulated that programmable matter arrangements of arbitrary size would "not be governed by technological limits, but by economic ones" [2]. BitDrones can be assembled into arrangements of arbitrary size, which are restricted on a small scale by the catomic resolution of individual drones, and on a large scale by the operational volume created by their supporting infrastructure. At some point in size, BitDrones would fail due to technological limitations

whether it be for lack of centralized computational power, visual obstruction of the drones from the positioning system, or turbulence from surrounding drones. However, with up to 30 drones in use at once in the most recent iteration of the BitDrone system, there was no indication of any functional technological limitations imposed on the size of their structures.

3.2.2 Dynamic Reconfigurability

BitDrones could be dynamically reconfigured in space, but due to their centralized control architecture the elements could not form a dynamic computing network. From an interface standpoint this was of little concern to a user so long as the catoms' behavior reflected the properties of the programmable matter that it constituted. In the control architecture, drones were treated as software objects and external user activity propagate changes through programmable matter "material" composed of recruited drone objects. The behavior of each physical BitDrone reflected the behavior specified in its drone software object.

3.2.3 Internal and External Drive

Most importantly, BitDrones was a programmable system composed of actual matter (the drones). It could be controlled internally through events triggered in its centralized system and was designed to respond to external user input. This external input was applied in the form of direct manual interaction, gestural interaction, and facilitated interactions through various purpose-made hardware and software tools.

3.2.4 Real-Time

As a result of its physical nature, BitDrones could be directly observed visually and haptically. It could also be modified in real-time through several modes of manual, gestural and facilitated interactions. The control architecture allowed for analysis through real-time logging, as all information was centralized.

Chapter 4. System

4.1 Overview

4.1.1 Drones

Individual BitDrones were the main components of the programmable matter system and served as the physical manifestation of digital information. An external carbon fiber wireframe structure provided a graspable shape to facilitate tangible user interaction, and integrated RGB LEDs illuminated the drone for increased dimensionality. Several different generations of drone were designed over the course of this research effort to improve upon aspects of previous drones or to address limitations discovered in prior designs**.**

4.1.2 VICON Motion Capture

As BitDrones lacked the means to independently position themselves, an external vision-based motion-capture system was used to find the cartesian coordinates of the drones in relation to an arbitrarily specified origin. The VICON motion capture system triangulated IR reflective marker patterns from the network of calibrated cameras with sub-millimeter accuracy, where each distinct pattern corresponded to a separate drone. The functionality of BitDrones was restricted to the operational volume of the VICON system, which was usually limited by the size of the room available to host BitDrones, the available camera mounting positions, or the maximum visual range of the cameras (which is further dependent on the size of the markers, ambient reflection, user-tuned focus and zoom)**.**

4.1.3 BitDrones OS (Operating System)

BitDrones OS was created by Sean Braley (M.Sc.) and coordinated each BitDrone in the swarm to behave in accordance with the material qualities of the programmable matter that they constituted. At a high level, this equated to moving the drones to specific 3D coordinates at

varying speeds and changing the colour of their onboard LEDs. This multi-drone system was unique in its low-level centralization because it sent each drone directional and thrust commands to control position and velocity in real-time via a standard WiFi connection. This is as opposed to sending positional updates to each drone through point-to-point connections. BitDrones OS also ran software for different interaction scenarios that provided varying functionality to the drone swarm, which took advantage of several different input devices.

4.1.4 Input

There were several input modalities to interact with individual BitDrones, as well as to interact with groupings of BitDrones. Tangible manipulation and interaction were prioritized, as that is the most natural way that humans interact with physical matter. Gestural interactions were introduced in order to accommodate for shortcomings found with purely tangible interactions as development progressed. External physical and app-based tools were created to more easily sculpt and manipulate large numbers of drones, as well as to alter non-physical parameters within BitDrones OS. Lastly, embodied interactions were explored through specially designed controllers that allowed for intuitive control of the BitDrone swarm.

4.1.5 Network

The drones, VICON, BitDrones OS and input devices were networked together with a special focus on low-latency data transfer. The VICON system was networked with BitDrones OS via a local area network (LAN), while the drones and input devices were connected via a wireless LAN (WLAN). Due to the centralized control system, special efforts were taken to minimize latency in the network through the use of unicast user datagram protocol (UDP) broadcast packets and shifting the bulk of the routing to individual BitDrones.

4.2 Hardware

4.2.1 BitDrones

As the BitDrones project progressed, ten different drones were successively developed in order to address emerging challenges and introduce new functionality. In the beginning the goals were simply to become familiar with drone design and to implement basic wireless communication and computer control. By the end of the project, the goals had matured to reflect larger challenges associated with programmable matter such as forming cohesive physical structures from the drones.

4.2.1.1 Version 0

Figure 1: BitDrone V0

BitDrone V0 was the first quadrotor drone that was designed for the BitDrones project and was capable of self-stabilization in flight and manual wireless control via software running on a remote computer. It was based on an off-the-shelf Micro MWC hobbyist flight controller (FC) board, a 2.4 GHz XBee S2 (Series 2) radio frequency (RF) module for point-to-point communication and four 6 mm diameter coreless motors paired with 45 mm diameter propellers for propulsion, all powered by a 300 milliamp hour (mAh) lithium polymer (LiPo) battery. The Micro MWC consisted of an ATmega328P-AU microcontroller unit (MCU), an Invensense MPU-

6050 IMU, a 2.4 GHz DSM2 (Digital Spectrum Modulation standard 2) RF module, four PWM (pulse-width modulation) driven MOSFETS for coreless motor control, and 3.3V and 5V power regulation. The frame was designed in SolidWorks 2014 and 3D printed with acrylonitrile butadiene styrene (ABS) plastic on a Makerbot Replicator 2X.

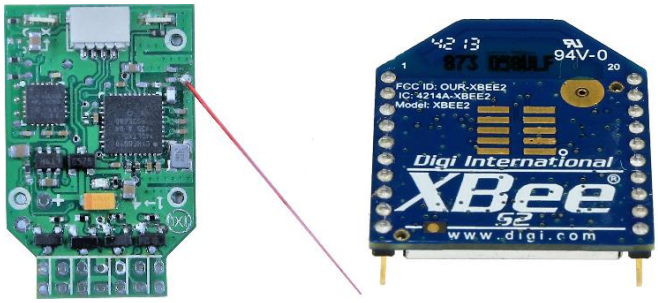


Figure 2: Micro MWC FC (left), XBee S2 w/trace antenna (right)

The main goal of V0 was to replace the DSM2 module on the Micro MWC FC with an XBee to allow for wireless serial communication. Normally the DSM2 module would connect to a standard hobbyist radio controller to allow for manual control, but this was not feasible to use for the autonomous coordination of many drones simultaneously. XBees are industrial RF modules meant for use in mesh networks and are highly reliable, communicating through several user-specifiable protocols.

At the time of its design, there were only two viable open-source autopilot projects; ArduPilot and MultiWii (MW). MultiWii was adapted to a much wider variety of hobbyist autopilots and allowed for a lower level of firmware customization over ArduPilot, which made it a logical choice for the basis of BitDrones. The MultiWii Serial Protocol (MSP) is a packet-based serial communication protocol that allows for control of MultiWii enabled FC via a serial interface. MSP was also designed to be more efficient than the Micro Air Vehicle Link (MAVLink) protocol utilized by ArduPilot, which would reduce latency when controlling a large number of drone elements.

A computer running Processing 2.2.1 sent serial commands to BitDrone V0 from a USB 2.0 connected XBee. To determine an acceptable command update rate that the XBees needed to provide, MultiWii 2.3.0 was modified to only accept commands from the DSM2 module every *x* seconds as specified by a user through a custom Processing application and through modification of the MSP. Changing the update rate from the DSM2 module allowed the user to qualitatively determine the minimum update frequency necessary to fly the drone from a standard hobbyist radio controller. It was determined experimentally that a minimum command update rate of 20Hz was required for stable and responsive control.

Another custom Processing application accepted input from a keyboard's "w, a, s, d" and arrow keys to control the drone, constructed MSP packets, then wrapped those MSP packets as the data payload within XBee packets for transmission. "w, s" increased and decreased thrust, "a, d" controlled yaw and the arrow keys controlled pitch and roll. Holding a keyboard key would additively apply increasingly larger values to its associated parameter, while releasing a key would reset that parameter. Updates were sent at significantly more than 20 Hz, resulting in stable flight from a standard computer. However, no autonomy was granted to V0 as it could not accommodate VICON markers for tracking.

4.2.1.2 Version 1

Figure 3: BitDrone V1 (PixelDrone)

BitDrone V1, or the “PixelDrone”, was the first BitDrone designed for autonomous flight and user interaction in formations of up to four units. It was built on the design of V0 and utilized the same electronic components. Notably, V1 added a ring structure above the drone to accommodate an IR marker patter for VICON tracking, as well as an RBG LED. The frame was 3D printed from ABS on a Stratasys uPrint SE utilizing dissolvable support material. At this point, BitDrones were not fully tangible due to the exposed propeller blades, but merely touchable from the ring structure. The LED also granted this drone the ability to act as a flying pixel, but because it lacked a tangible volumetric exterior to illuminate it, it was not considered a voxel. Like V0, it was powered by a 300 mAh LiPo battery which provided approximately five minutes of dynamic flight time for interaction.

V1 BitDrones were centrally controlled from a Bootcamped 2012 iMac Pro with 16 GB of RAM and an Intel® Core i7 running Windows 7, a custom C# application to control the BitDrones, and VICON Nexus 1.8.5 to obtain drones’ location and orientation. At this initial stage, it was decided to centralize the drones’ positional control system because sending VICON position and orientation data would have meant sending larger packets than sending packets containing direct control commands alone. This choice translated to faster update rates and ultimately to better flight control.

V1 utilized the same onboard XBee RF modules as V0 to serially interface with the flight controller. The onboard XBee also controlled the BitDrone’s LED directly via its own I/O from packets generated by the custom C# application. XBee networks are highly scalable and are able to adjust their networking mode on the fly, so it was desired to configure a single XBee on the host computer to address all airborne XBees. Unfortunately, configuring the XBees as a mesh network resulted in severe packet throughput issues, and efforts to enact sequential point-to-point communication in a star network configuration were frustrated by high latency. In order

to maintain update frequencies and to minimize latency, each V1 drone required its own point-to-point XBee connection to the host computer.

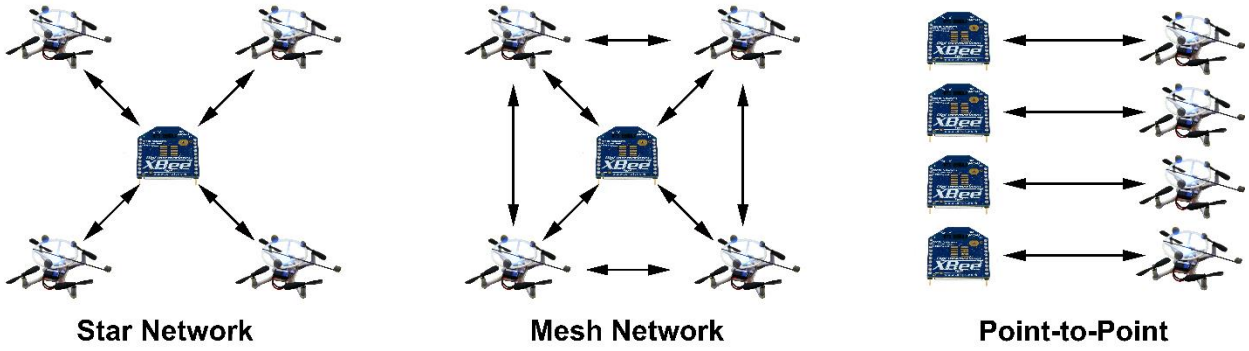


Figure 4: Various XBee networking configurations

The VICON marker pattern for each drone was composed of four adhesive IR reflective markers that allowed the system to recognize the drone's location and orientation. Later versions would use a minimum of five markers for redundancy. Another issue with BitDrone V1 was that it was extremely delicate and would often be damaged in crashes during testing. For ease of development future drones were designed to be more robust to avoid investing time on repairs.

4.2.1.3 Version 2.1

Figure 5: BitDrone V2.1 (PixelDrones)

BitDrone V2.1 was the next iteration of the PixelDrone and was based on the design of V1. The most pronounced improvements were the creation of a more robust ABS frame so that the

drones could endure the rigors of testing and interaction, as well as a movable stalk on which to mount an additional fifth IR marker to further differentiate drone marker patterns. This resulted in a heavier frame which necessitated the use of larger 55 mm diameter propellers and 7 mm diameter coreless motors.

All V2 drones were controlled similarly to V1 using the same computer running VICON Nexus and custom C# coordination software. This software was dubbed “BitDrones OS” and had undergone radical changes beyond flying the drones in order to accommodate more complex interaction scenarios. In order to scale the system to support a larger number of BitDrone elements, the XBee S2s were swapped for WiFi capable XBee S6Bs so that a standard wireless-N network could be utilized instead of the ZigBee protocol. This meant that multiple point-to-point radios for the coordinator PC were no longer necessary. To accommodate for the increased power consumption of the WiFi Xbees, each V2.1 drone was equipped with a 3.3V switching regulator. Even with WiFi XBees significant latency prevented more than eight drones from being flown simultaneously.

4.2.1.4 Version 2.2

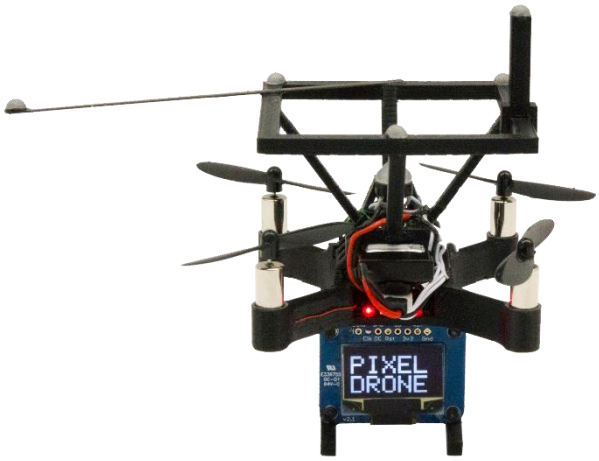


Figure 6: V2.2 with integrated OLED screen (front)

V2.2 was identical to V2.1 except for the addition of a black and white organic LED (OLED) display mounted beneath the drone. To present BitDrones’ as practical catoms, thousands or

even millions of microscopic drones would need to be able to fly in very tight formations to display complex graphics. As research progressed, it became apparent that there were particular scenarios where greater information density was required than a single RGB LED could convey.

To increase the display density, drones with different sub-voxel imaging techniques were designed. In the case of V2.2, an Adafruit 128x64 display was able to readily convey additional complex virtual information to a user via text. Due to the lack of available input/output (I/O) on V2.2's MCU, an Arduino Micro with an ATmega328P-AU mounted on the rear was used to drive the OLED screen.

4.2.1.5 Version 2.3

Figure 7: BitDrone V2.3 (DisplayDrone, front)

BitDrone V2.3, or the DisplayDrone, took on a radically different form than previous drones. Its primary feature was the inclusion of a 1280x720 LG flexible organic LED (FOLED) touch sensitive screen, which was inspired by the need for increased display density over V2.2. An Android™ board with a Qualcomm Snapdragon™ 800 paired with 2 GB of RAM, 16 GB of storage and WiFi running Android 5.1 drove the display, meaning that it could run standard Android applications. The screen was curved 90 degrees around the drone, held in place by a 3D printed ABS frame which was affixed to a custom 2 mm thick laser-cut G10 fiberglass

quadcopter frame. The screen retained its touch capabilities for user interaction, which allowed not only for control of the DisplayDrone but additionally to alter parameters of the BitDrone system as a whole, serving as a sort of terminal depending on the functionality of the Android application it was running. It also carried a 2.1 MP MIPI camera at the bottom corner of the screen which made it possible to capture real-time video and host video calls.

Though it was based around the same Micro MWC FC and XBee SB6 as previous drone versions, it required a more powerful propulsion system to carry the weight of the screen's supporting frame and electronics. V2.3 used four powerful, higher efficiency brushless motors spinning 76 mm fiberglass reinforced plastic (FRP) propellers each driven by a discrete 6A three-phase electronic speed controller (ESC). A 7.4V, 1000 mAh LiPo battery powered both the drone's main systems as well as the onboard display. V2.3's relatively sizeable battery had to be mounted in a location so that it would counterbalance the weight of the screen, supporting frame and driving electronics. This positioned the battery directly in the thrust stream of the rear two propellers which would have decreased lift, so the battery was mounted such that its longest dimension was perpendicular to the propeller's plane of rotation in order to minimize obstruction of the generated thrust. The 3D printed ABS frame that housed the battery served as a rear landing foot to stabilize the drone, as well as afforded additional rigidity to the rear two quadcopter arms.

Being more massive, possessing a much more powerful propulsion system as well as having exposed propellers, the DisplayDrone was operated only by experienced users. In order to reduce weight propeller guard were not featured in this prototype. V2.3 was much more a proof-of-concept than any other drones within the BitDrones ecosystem.

4.2.1.6 Version 2.4

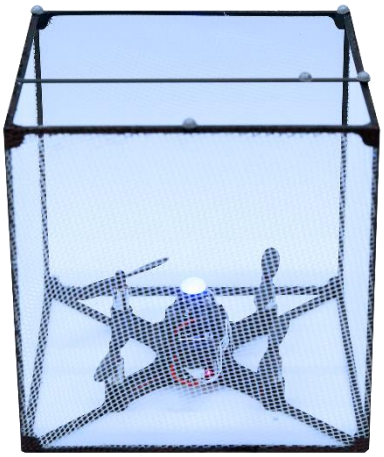


Figure 8: Illuminated BitDrone V2.4 (ShapeDrone, front)

The namesake of the BitDrones project was V2.4 or the “ShapeDrone”. The core of V2.4 was similar to V2.1 and V2.2, but it also supported a carbon fiber and ABS frame over which a highly air permeable mesh was wrapped. The main purposes of this mesh were to act as a diffuser for the internal LED and give the drone the appearance of a solid shape, as well as to provide a tangible exterior. Denser diffusers provided a much more convincing solid, coloured appearance but ultimately could not be used as they restricted airflow to each BitDrone’s propellers, grounding the drones.

ShapeDrones were developed and flown in a variety of shapes to facilitate construction as architectural components. Generally, the size of BitDrones themselves limited display density when flying in formations, so it was necessary for the drones to individually afford some level of detail as required. The cube form was chosen as the ultimate shape of future BitDrone iterations for several reasons.

Figure 9: Prototype ShapeDrones showing a rough sphere (left), cylinder (middle), rhombicuboctahedron (right)

A cube was the simplest shape to contain a drone that resulted in an easily constructed frame with the fewest parts and sufficient horizontal space to place VICON markers. Cubes also afford a more easily graspable form, as opposed to shapes such as the rhombicuboctahedral drone portrayed in *Figure 9* above which is difficult for the fingers to gain purchase on due to the obtuse angles between faces. Additionally, a BitDrone is representative of a volumetric pixel in 3D space, so it should also physically appear as the simplest form of a voxel (a cube).

As BitDrones was first and foremost a human-computer interface, its appearance was a heavily weighted focus for the design of the drones. Vintage aesthetic of low-resolution graphics and pixelated 8-bit art are based on square pixels with limited color depth, mimicking those rendered within the constraints of decades old hardware and software. BitDrones is a display with visible resolution and density limitations, so the blockish shape was adopted in part to resemble familiar retro graphics.

Finally, in many construction-based toys the simplest base shapes are rectangular prisms, such as in sets of wooden building blocks or LEGO®. Significant efforts were made in order to present BitDrones as a safe, robust, visually appealing and user-friendly system, which meant designing it with many of the same considerations as a toy. As a toy-like programmable matter system of which its primary purpose was construction, it was natural that individual display

elements would mimic simple coloured blocks to instill a sense of familiarity to its users. Conclusively, the cubed shape of the BitDrone was decided upon due to simplicity and familiarity.

Though it was possible to fly twelve drones simultaneously, the Version 2 BitDrone system was prone to latency spikes and critically low update frequencies that sometimes resulted in complete system failure. This occurrence was made more likely as the number of airborne drones increased, and as a result only about five were operated simultaneously in performance critical scenarios. This was a special concern when the DisplayDrone (V2.3) was in flight due to the potential safety hazards that it posed.

4.2.1.7 Version 3

Figure 10: BitDrone V3 (front)

BitDrone V3 was the successor to the ShapeDrone, carrying a familiar cube-shaped frame. It was designed very differently however, with an increased focus on ease-of-use for research. A significant issue with previous drones was their delicate construction and non-modular design, which meant rebuilding new drones if a single critical part could not be fixed or easily replaced. V3 focused on durability and modularity and the new design drastically increased the lifetime of each drone. Simpler maintenance and longer lifetime meant that a much larger fleet could be readily maintained for use.

V3 consisted of a flexible, durable carbon fiber frame constructed from sixteen 2x0.5 mm carbon fiber bars connected via eight 3D printed ABS plastic corners. The drone itself was a modular "core" with four thin carbon fiber arms that was secured inside the outer frame and held in place via pressure alone. This allowed the core to be secured and removed as needed. A flat ABS frame held the Micro MWC FC, while the motors press-fit into a socket at the end of each arm of the frame. This allowed the motors to be easily replaced when worn out or damaged. A clip-on 3D printed ABS "backpack" contained an RGB LED, a WiFi module and supporting circuitry. This backpack clipped onto the BitDrone core and could be swapped out for repair or replacement. A mesh diffuser was excluded from this version as they were easily ripped in

crashes and it was not worth the effort to maintain them through active development with experienced users.

The BitDrones OS was ported from C# to C++ in order to run on a near real-time kernel of the Ubuntu Server OS installed on a Dell Optiplex 745 with 12Gb of RAM and an Intel Xeon processor. This upgrade was necessary because operation in a Windows environment introduced severe delays in BitDrones OS due to system interrupts beyond the researcher's control. This modification drastically increased the reliability of BitDrones OS by eliminating latency spikes that historically resulted in system failure.

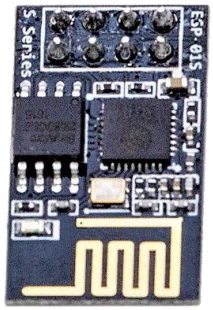


Figure 11: The ESP-01 WiFi module

Until this point, BitDrones had used XBee modules for drone-side communications. They were exceptionally reliable and easy to use, however could not deliver the throughput or near real-time link necessary to fly any significant number of drones. An ESP-01 module as seen in *Figure 11*, designed around an ESP8266 WiFi system-on-chip (SoC) replaced the XBees and was able to provide the near real-time performance required by BitDrones. This performance was obtained through WLAN optimization and routing done at the drones' firmware level which is discussed in *4.6.3 Routing*.

A small handmade daughterboard was soldered directly to each ESP-01 module and consisted of 3.3V linear power regulation and 3.3V/5V logic level converter (LLC) circuitry. Linear regulation required only three components, resulted in a much smaller footprint than the off-the-shelf step-up/step-down switching regulators and reduced weight. Initially it was believed that

the ESP-8266 SoC was 5V signal tolerant but after displaying unpredictable behavior and the death of several ESP-01's, the error was realized. The LLC was introduced as an interface between the ATmega328P-AU (5V logic) and the ESP-8266 (3.3V logic) and improved communications reliability.

In order to reduce the size of each drone and maximize the propeller area inside the footprint of the external frame, the propellers were fitted to a custom jig and laser cut from 55 mm to 45 mm diameter. The end result was that 5 mm was trimmed from each end of the propeller in order to convert them to bullnose style. As their name suggests, bullnose style propellers have blunt ends. They are also less aerodynamic and less efficient. However, they do provide a larger surface area to diameter ratio which in turn increases thrust for a given propeller diameter.

A smaller propeller meant that to achieve similar thrust characteristics, higher speed coreless motors were necessary. Utilizing smaller 45 mm bullnose propellers with the 11,000 K_v (revolutions per minute per volt) motors of previous generations resulted in overheating motors that would burn to the touch due to lower electrical resistance at higher revolutions per minute (RPM). The heat significantly affected motor lifetime (dropping from the estimated 5-6 hrs to 2-3 hrs) as well as drone flight time which dropped to five minutes. Using higher speed motors of 14,000 K_v resulted in lower temperature that granted a longer operating lifetime as well as longer flight times of approximately seven minutes.

4.2.1.8 Version 4

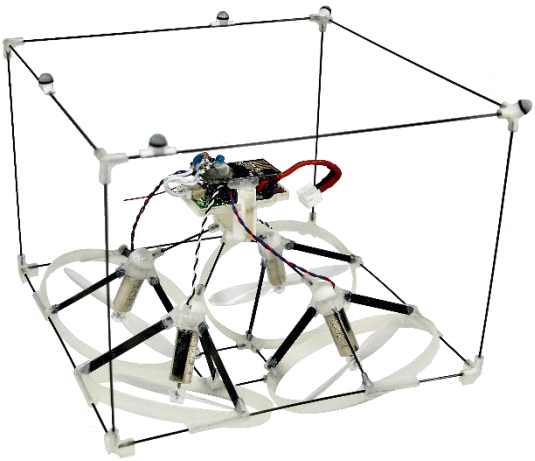


Figure 12: BitDrone V4 or the XY Drone (side)

A significant limitation of traditional drones is that they are not able to fly over top of each other in close proximity due to the downwards thrust of overhead drones. The *XY Drone* was designed with motors and propellers mounted 27 degrees outwards from the quadcopter's yaw axis in order to disperse and redirect thrust. This was done to create a less turbulent area directly below, in which another drone could occupy the same *xy*-coordinates in flight.

Apart from an absent RGB LED, V4 retained identical electronic components to V3 but was built around a much more complicated and lightweight airframe. This frame consisted of 52 individual carbon fiber and 3D printed photopolymer parts bonded with cyanoacrylate (Super Glue®). The structure was designed to be highly flexible when assembled, but this came at the expense of modularity because the components had to be bonded together. As a result, V4 had low repairability potential. Unmodified 55 mm plastic propellers were surrounded by thin profile ducts, the aim of which was to reduce propeller tip vortices in order to further decrease turbulence. The *XY Drone* did not retain RGB illumination as it was designed only to assess the effectiveness of the tilted rotor design.

4.2.1.9 Version 5.1

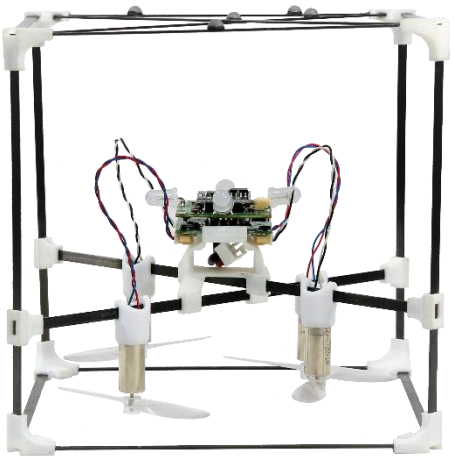


Figure 13: BitDrone V5.1 (front)

Version 5.1 was the last major BitDrone redesign and improved upon previous drones in several ways. The overall design remained very similar to that of V3, with a quadrotor core contained within an exterior frame constructed from 3x.5 mm carbon fiber bars and nylon 3D printed parts. The nylon components were fabricated by the LEGO Group using selective laser sintering (SLS) which resulted in better dimensional accuracy and much stronger parts, granting this drone terrific physical resilience.

The most pronounced feature of V5.1 was the inverted motors and propellers used in a pusher configuration, meaning the propeller was placed behind the motor. This greatly increased maneuverability of the drones due to a higher center of mass, as well as increased overall flight time by approximately one minute (17%) over V3. It is theorized that this increase in flight time was because the frame and motor itself were not in the propellers' thrust stream, thereby reducing drag and increasing efficiency. Unmodified 55 mm propellers were utilized within the same volume as V3, and to achieve this they were offset in the *z*-dimension so that the propeller paths overlapped in the *xy*-plane but would not collide.

A custom flight controller called the *WiFly V1* was designed in Autodesk Eagle 8.5.2 to consolidate components, streamline construction, improve on previous functionality and add

new features. Like the Micro MWC FC, it incorporated an ATmega328P-MU MCU, an Invensense MPU-6050 IMU, and motor-driving MOSFETs. Additionally, it included LLCs, an ESP-8266 WiFi SoC, four RGB LEDs, integrated LiPo battery charging and plug connections for modularity and interfacing. The first design is featured in *Figure 14* and *Figure 15* was a dual layer printed circuit board (PCB) with components top and bottom.

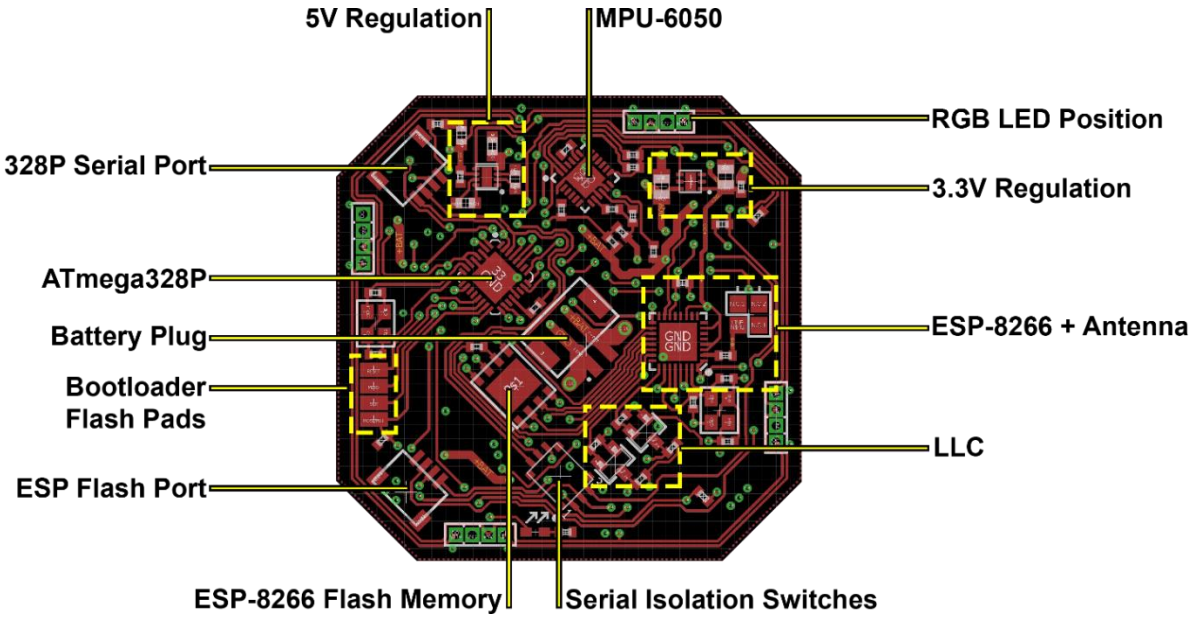


Figure 14: WiFly PCB CAD (top)

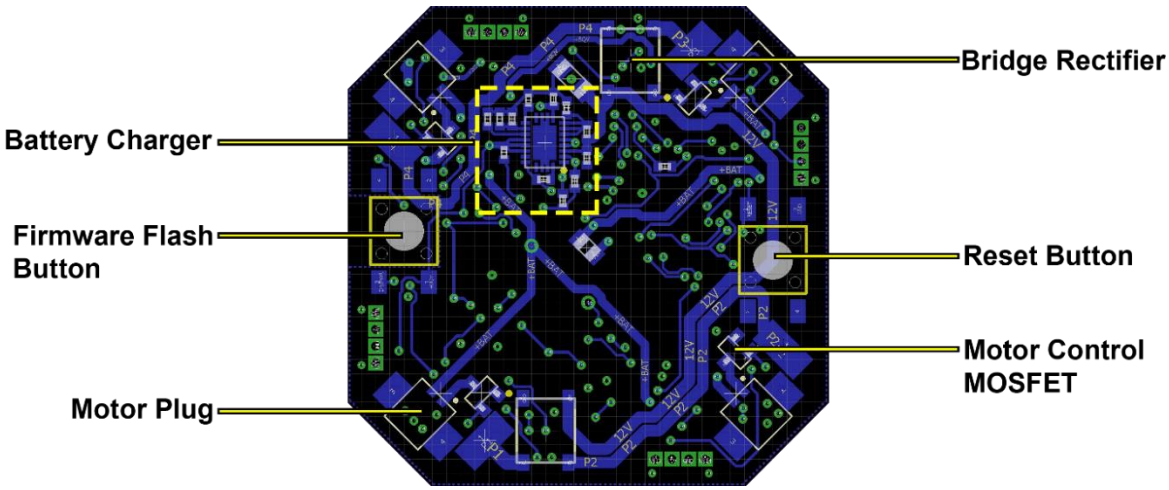


Figure 15: WiFly PCB CAD (bottom)

Due to a limited budget and timeline, only one manufacturing run was possible and only the most cost-effective production options were considered. The relatively tight tolerances required to produce the first design of the *WiFly* could have potentially resulted in a high number of manufacturing defects. For this reason, the *WiFly* was redesigned to be manufactured with looser tolerances and the first design was never produced.

Larger PCB traces and vias meant less precise tolerances were required for manufacturing but this also meant a larger physical footprint for the PCB. The decrease the footprint the design was split into three stackable boards linked together through vertically soldered signal and power pins. It was also decided that one of these three boards would be the previously used ESP-01, as they performed well and were were less expensive than sourcing the components individually for a custom WiFi board.

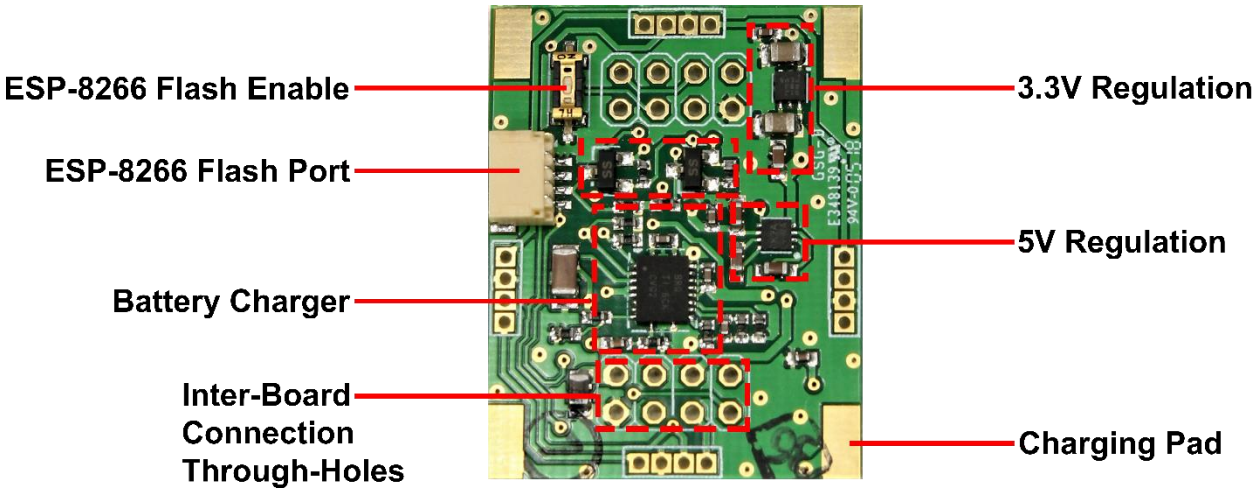


Figure 16:Completed WiFly V2 Junior PCB (top)

The middle board was nicknamed *Junior* and was positioned between the ESP-01 on the top and a second board below nicknamed *Senior*. The topside of *Junior* is shown in *Figure 16* and it contained 3.3V and 5V regulation circuitry, LLCs to interface with the ESP-01, and a Texas Instruments BQ24070 LiPo charging IC with supporting components. Four copper pads, one in each corner of the board, were connected by thin wires to conductive strips inset in the bottom

four corners of the BitDrone's frame. These strips connected to external 12V charging rails that supplied the BQ24070, which could simultaneously charge the onboard battery and power the drone via a passthrough feature. A 4-pin micro JST® plug was also included in order to interface with the ESP-01, along with a switch to force the ESP-8266 SoC to enter flash mode for firmware uploading. Lastly, four sets of through-hole solder points were also added in order to attach I2C networkable RGB LEDs. On the bottom side of Ju*nior*, shown in *Figure 17*, two bridge rectifiers controlled the polarity of the 12V power supplied to the BQ24070 regardless of changes in input polarity.

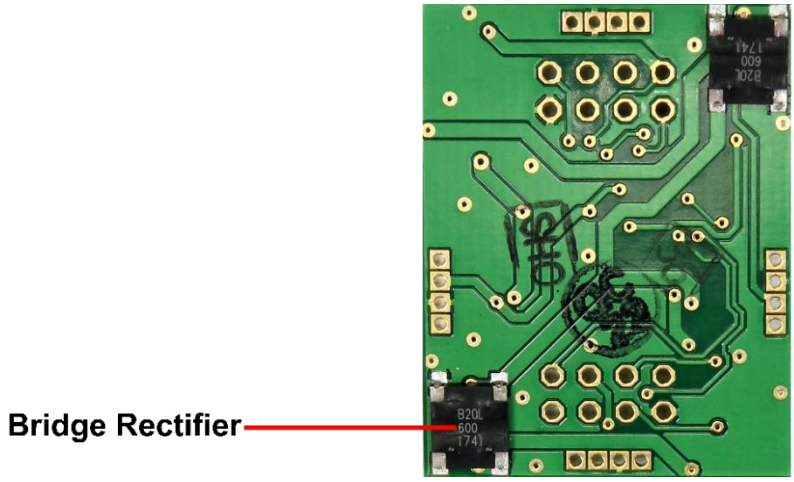


Figure 17: Completed WiFly V2 Junior PCB (bottom)

The bottommost board was nicknamed *Senior*, the topside of which is shown in *Figure 18*. It carried the ATmega328P-MU and supporting circuitry, the Invensense MPU-6050 IMU, four 4.2A MOSFETs for brushed motor control as well as voltage monitoring circuitry. Large power distribution traces were used in order to satisfy the peak power demands of the four coreless motors.

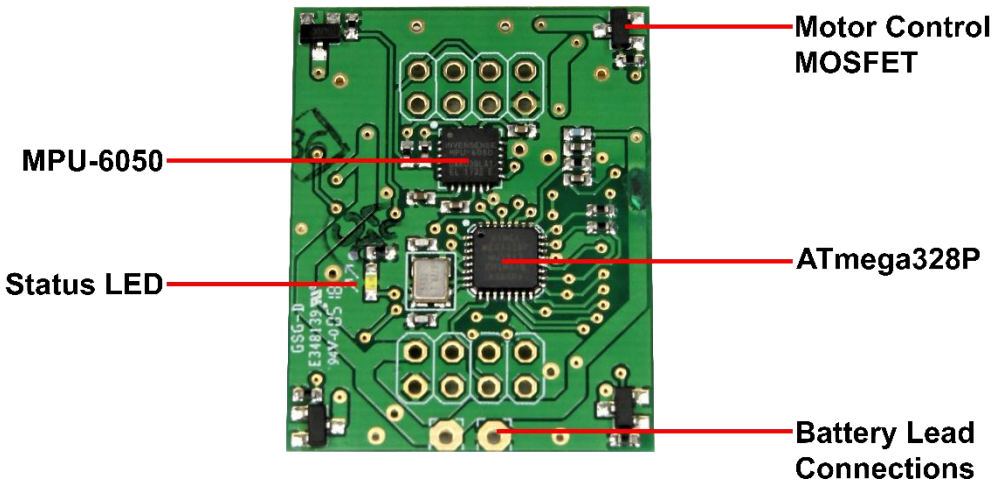


Figure 18: Completed WiFly V2 Senior PCB (top)

Figure 19 displays the bottom of the *Senior* board, which incorporated mostly interface connectors. Each of the four coreless motors attached to the four 2-pin JST plugs at the corners of the board, while there was also a 4-pin micro JST connection for interfacing with the ATmega MCU. The double pole double throw (DPDT) switch on the righthand side connected the serial receive (rx) and transmit (tx) lines between the ATmega and ESP-8266 and allowed them to be electrically isolated for separate external interfacing. The use of software programmable MCU I/O for serial communication between the ATmega MCU and ESP-8266 was tested during prototyping. This would have eliminated the need for electrical isolation in order to interface with them separately via hardware serial. However, it was found that software serial implimentations were slower than hardware serial connections and sometimes resulted in unpredicatable behavior due to buffer overflows and dropped data. A hardware serial connection was prioritized in order to avoid serial buffer overflows on the MCU. Lastly, a membrane button was added to the left side of the *Senior* board to serve as a hard reset switch and four solder pads below it were included for SPI bootloader flashing of the MCU. The positive and negative battery leads were soldered at the bottom of the board to two large through-hole pads.

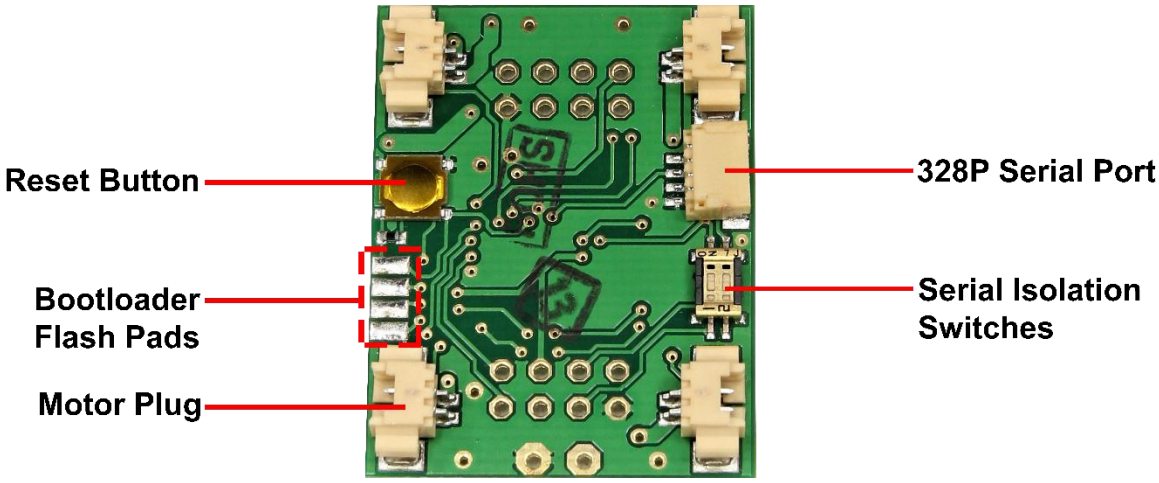


Figure 19 Completed WiFly V2 Senior PCB (bottom)

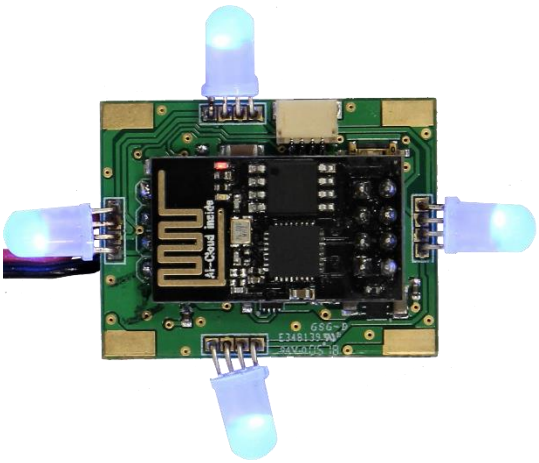


Figure 20: Assembled and powered WiFly V2 (top)

The assembled *WiFly V2* is pictured in *Figure 20* after the connection of the three separate PCBs by rigid vertical pins, and the addition of four RGB LEDs. It was secured within the BitDrone frame using Super Glue® and a clip mechanism built into the frame itself. Four copper strips were embedded into the bottom corners of the BitDrone frame as shown in *Figure 21* to contact charging pads and transfer power to the *WiFly V2*.

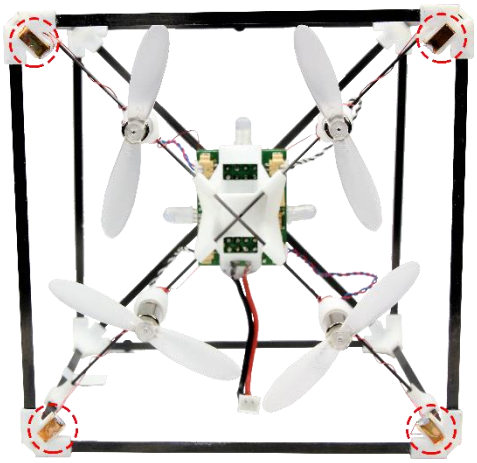


Figure 21: Bottom of V5.1 showing copper charging contacts circled in red

The charging mats were constructed of black foam core board with two strips of conductive copper tape that acted as power delivery rails shown in *Figure 22*. An IR marker pattern at either end enabled tracking of the charging mats in the VICON area in relation to the BitDrones. The copper tape also had conductive adhesive, meaning that the charging mats could be easily linked with a piece of tape bridging each rail on the underside to share the same power source. This also allowed lengthy mats to be folded for easy transportation and setup. The mats were connected to a 12V power supply via small alligator clips attached directly to the tape, with positive voltage delivered to one strip and the other connected to ground.

Figure 22: A charging mat section (left) and two charging mats connected together (right)

Due to the large surface area of the strips and the on-board rectification of the BitDrones, the drones could land in any position on a charging mat to receive power as depicted in *Figure* 23. The only physical conditions were that at least one bottom corner was required to contact the

positive strip and at least one other corner was required to contact the ground strip. The BitDrones' carbon fiber frames were additionally elevated above the charging mats by the nylon frame corners so as not to cause a short circuit across the power delivery rails.

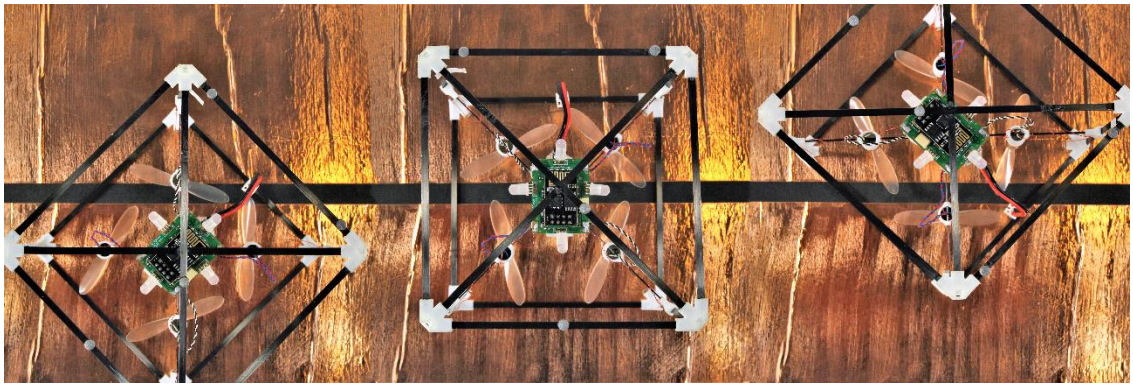


Figure 23: Three acceptable BitDrone charging positions (top)

4.2.1.10 Version 5.2

Figure 24: BitDrone V5.2 corner with embedded spherical magnet circled in red

Version 5.2 or "DroneForm" was nearly identical to Version 5.1 save for newly designed frame corners, each of which incorporated a spherical neodymium magnet. Sockets in each corner loosely held the magnets, allowing them to self-orient in the presence of other magnetic fields. For the magnets to be inserted into their sockets, one face of each corner had a slightly larger socket entry point. After introducing the magnet, a small piece of carbon fiber was attached to cover this entry point and retain the magnet inside.

These magnets enabled the drones to connect to each other at corners, along edges and to faces. BitDrones had previously struggled to form compelling 3D structures when flying in tight formations (< 10 cm proximity) for two reasons. The most prominent issue was that of acute turbulence from surrounding drones, which the BitDrones OS's control system was not advanced enough to handle. The second issue was that the motion of the drones in relation to each other was more apparent in close proximity versus flying in a large area, resulting in visually unstable 3D models. Physically connecting the drones eliminated these issues but presented alternate challenges discussed in *Chapter 5. Exploration*.

4.2.2 Input Devices

Many input modalities were developed to interact with the BitDrone system, all of which necessitated the design and use of specialized devices. Initial interaction techniques heavily focused on utilizing the physical nature of the drones as tangible manipulation as this was considered the most natural way that humans interact with traditional matter. Complex physical manipulation and modification of non-physical system parameters proved cumbersome with manual and gestural interaction alone, so various tools were devised to grant users intuitive control of BitDrones.

4.2.2.1 Hand Straps

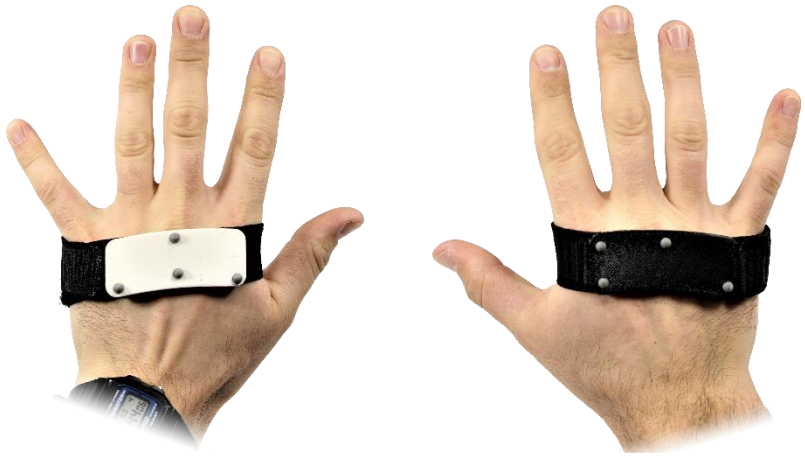


Figure 25: A user wearing left and right hand tracking straps

Hand straps were created to monitor the location of users' hands with the VICON system in the BitDrones operational area. A variety of touch detection techniques including networks of strain sensors and capacitive sensors were explored for implementation onboard the drones themselves. But as the option for gestural input was also desired it was determined that the most simple and accurate way to detect both hand proximity and gestures was to accurately track the absolute position and orientation of users' hands utilizing the existing resources of the VICON. Marker patterns were adhered directly to users' skin in the early stages of BitDrones. The VICON system was able to track a similar marker patterns between user sessions provided

that the relative locations of markers did not significantly deviate beyond the initially defined pattern. Reapplying the marker pattern to users was effortful and often resulted in inconsistent behavior due to slight variations in the applied patterns. It logically followed to create wearable permanent patterns that would produce consistent behavior.

A single design was created in SolidWorks 2016 and the model was mirrored for each hand. It consisted of a rigid ABS shape attached to an elastic fabric strap with its ends joined by detachable Velcro® for easy fitting and removal around various sized hands. The right hand was a solid black while the left was a solid white, each of which carried a differentiable IR marker pattern.

4.2.2.2 Mobile Android Device

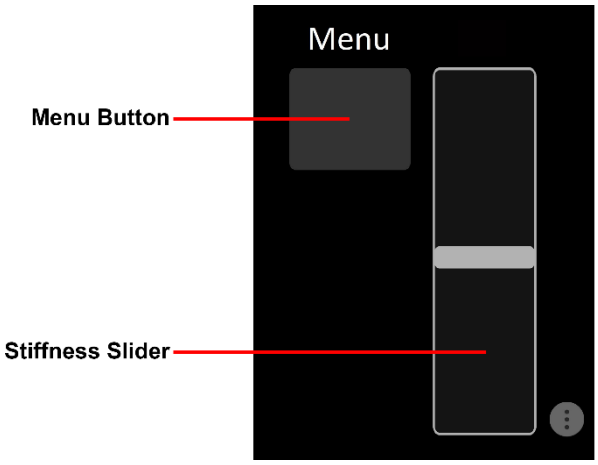


Figure 26: A simple Android app used to define non-physical parameters of BitDrones

In conjunction with physical and gestural interaction, mobile Android devices were used to queue specific behavior and to specify virtual parameters of BitDrones that would otherwise be difficult to adjust through the drone interface itself. Custom Android applications were created using TouchOSC® and the TouchOSC editor and could be run from Android devices connected to the BitDrones WiFi network. *Figure 26* shows a screenshot of one such applications with a virtual push button to summon a physical menu out of the BitDrone display and a virtual slider to specify a movement relationship between groupings of drones.

4.2.2.3 Wand

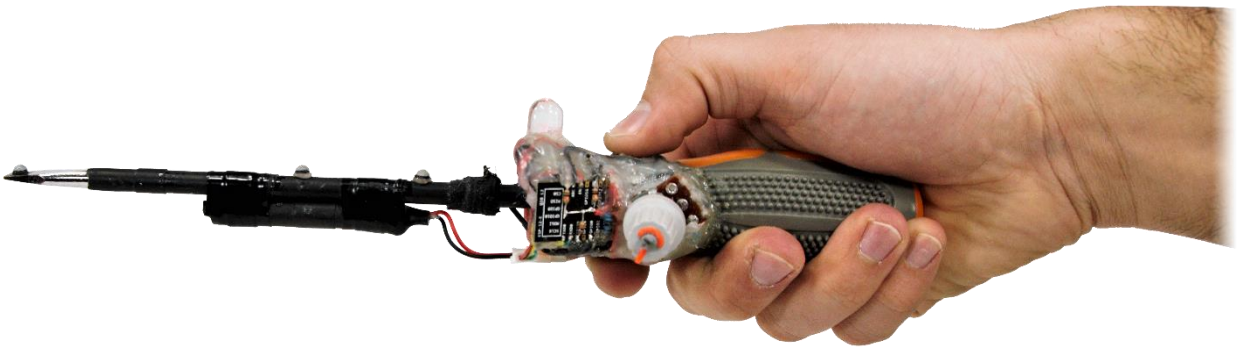


Figure 27: A user holding the first iteration of the wand

Complex selection and manipulation of BitDrone structures was onerous to perform using tangible interaction techniques alone. To facilitate such interactions, a wand-like pointing device was developed and prototyped around a conventional flat-head screwdriver as pictured in *Figure 27*. The Wand was inspired during testing from the repeated physical exertion of reaching for drones far away from each other or far from the user.

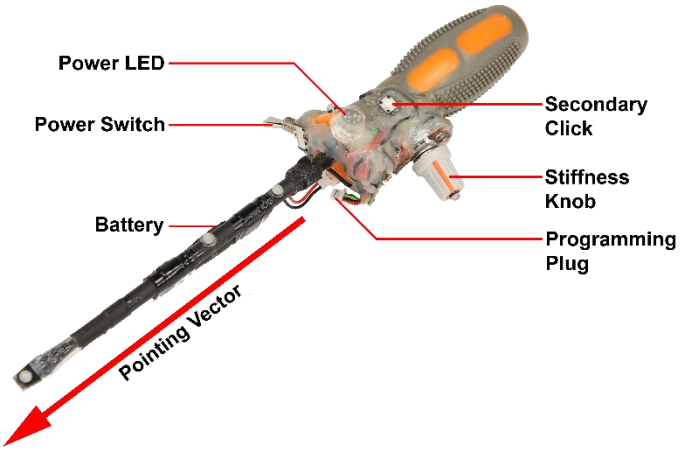


Figure 28: Components of the first wand (primary click on underside)

The most important aspect of the Wand was the pointing vector along its main axis created by three IR markers shown in *Figure 28*. This allowed virtual ray casting to select one or more drones, or to create virtual axes of rotation for 3D manipulation. It was connected via WiFi to the BitDrones network utilizing an ESP-12-E WiFi module based on an ESP8266 which also

processed user input from two buttons and a rotary control knob. These two buttons, one on the top reachable with a user's thumb and one on the bottom reachable with a user's index finger functioned similarly to the right and left click of a conventional mouse. The control knob enabled the linear modification of various BitDrone system variables and a single 200 mAh battery powered the wand.

Figure 29: User holding the second iteration of the wand

The first iteration of the Wand only had a single rotary knob with which to alter non-physical parameters of BitDrones. This one-dimensional interface was not sufficient when compared to the functionality of the Android application, so a colour touchscreen was integrated into the wand to consolidate functionality into one device. Pictured being held by a user in *Figure 29*, the second wand was designed in SolidWorks 2016 and 3D printed in ABS to match and surpass the functionality of the first wand prototype, with the touchscreen within easy reach of a user's thumb.

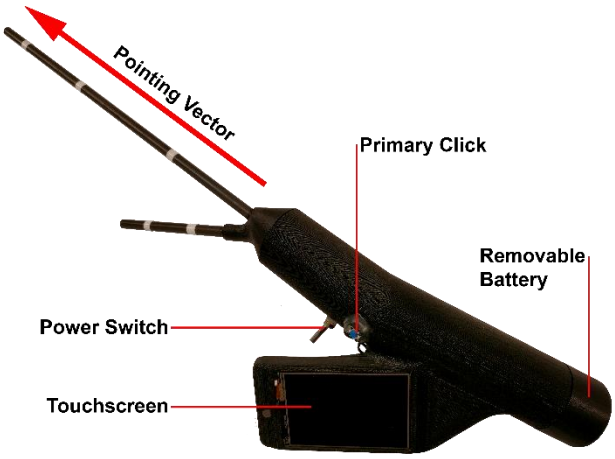


Figure 30: Components of the second wand (primary click on underside)

The second wand contained a 600 mAh 7.4V LiPo battery, an ESP-12-E, two tactile push buttons, a gen4-uLCD-24PT resistive touchscreen with integrated driver board and an indicator LED above the screen shown in *Figure 30*. Simple interfaces that consisting mostly of buttons and sliders were written in 4D Systems's proprietary 4DGL® language. The touchscreen module sent serial packets to the ESP-12-E when changes occurred in the interface, which were then sent as UDP packets to BitDrones OS.

4.2.2.4 Embodied Controller

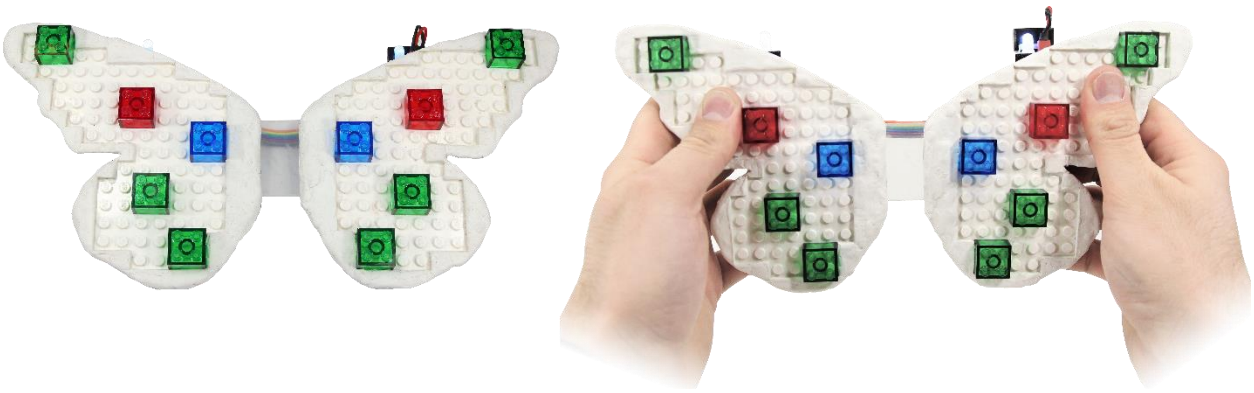


Figure 31: The handheld embodied LEGO® butterfly controller

In order to grant more intuitive control of complex BitDrone formations, a handheld controller shown in *Figure 31* was designed to construct and animate BitDrone models. It consisted of two laser cut LEGO baseplates with molded silicone edges that were joined together by flexible

translucent plastic. Coloured 2x2 LEGO bricks could be placed anywhere in the wing area and were representative of corresponding drones of the same colour in the same relative positions in the BitDrone model. A WiFi connected sensor package shown in *Figure 32* was affixed with rigid Velcro® Brand plastic hooks to the rear of each wing to measure and report the wings' orientation quaternions to BitDrones OS. Physical movement of the wings would then be reflected in the BitDrones themselves in real-time, which enabled users to fly and flap the BitDrone "butterfly" they had constructed.

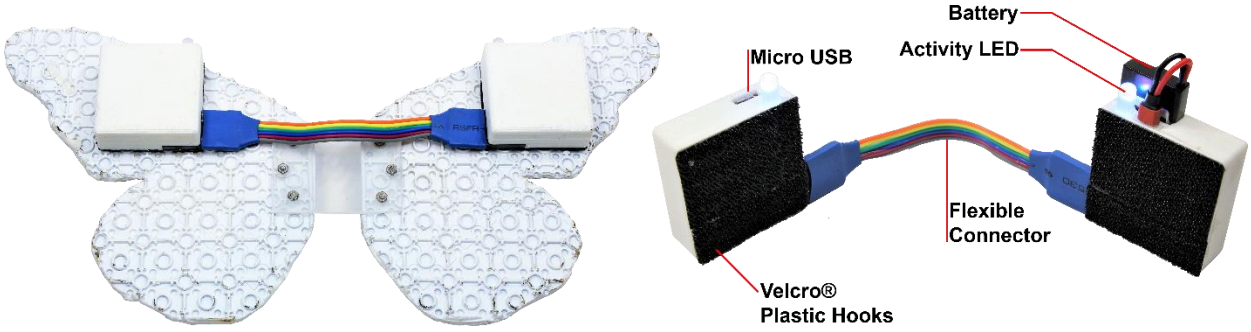


Figure 32: The sensor package affixed to the rear of the embodied controller (left) and the sensor package disconnected (right)

The sensor packages were connected together with a flexible wire connector and functioned as a single unit with distributed components. Each module contained an MPU-6050 IMU, a synchronized status LED and a strong neodymium magnet. The right package also held a 300 mAh, 3.7V LiPo battery while the left package additionally contained an Arduino Pro Micro, an ESP-01 as well as 3.3V and 5V regulation circuitry. The package housings were 3D printed in ABS and press-fitted together with the components inside. The sensor packages were designed with durability and modularity in mind, so the left and right packages were made to be hot swappable with new modules in the event of failure. The flexible pin connectors that joined them were designed to easily detach from the modules to prevent damage in the event of extreme mechanical stress.

4.2.2.5 Creation Scanner

Figure 33: The embodied controller anchored to the creation scanner

The creation scanner in *Figure 33* ascertained the colour of individual 2x2 LEGO bricks and their relative positions to each other. This colour and positional information was then used to autonomously create a BitDrone model to mimic users' LEGO patterns. The hollow body of the scanner was designed in SolidWorks 2018 and constructed of conjoined sections of laser cut medium-density fiberboard (MDF). It housed two neodymium magnets, a 1080p USB 2.0 camera, an RGB LED ring, an arcade-style pushbutton, an Arduino Leonardo and an Intel NUC with a Core i3-6100U and 8 GB of RAM.

Figure 34: The downward-pointing USB camera surrounded by the LED illumination ring

To ensure consistent imaging results, the neodymium magnets in the sensor package casings were aligned to embedded neodymium magnets on the underside of the scanning surface. This “snapped” the handheld controller into position whenever it was placed on the creation scanner. The USB camera was mounted overhead in the center of the LED ring in *Figure 34* which was required to illuminate the controller with white light for the scanning procedure. Since the actual imaging process was so brief, a more compelling visual experience was created to clearly indicate scanning activity. The scanning surface was illuminated with white light to capture the image, after which the illumination ring sequenced through white, red, green and blue colours as depicted in Figure 35.

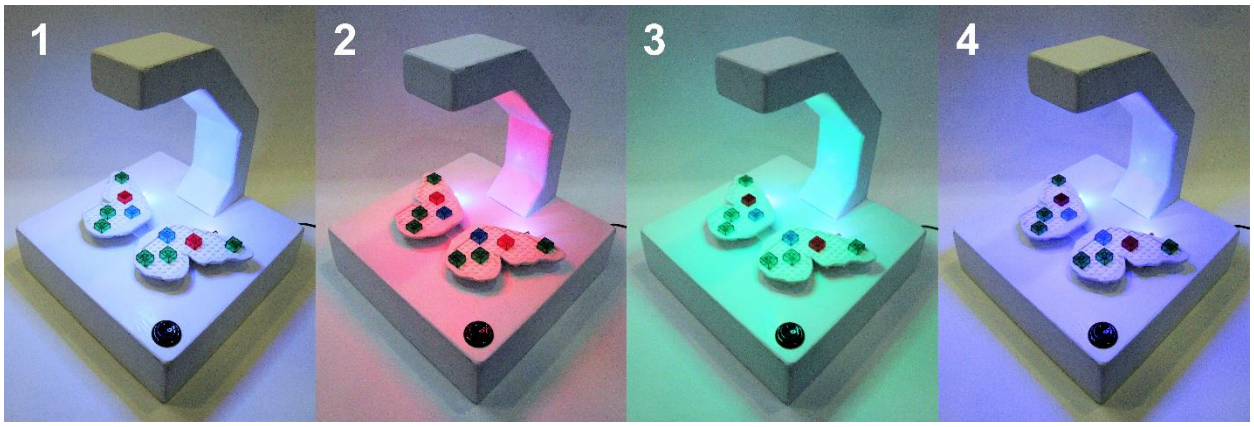


Figure 35: The lighting sequence of the scanning procedure

The webcam was connected to an Intel NUC mini PC shown in *Figure 36*, which ran a custom C++ application based on OpenCV 3.4. It communicated visual information to BitDrones OS via UDP packets through the BitDrones WiFi network. A large mechanical pushbutton connected to the Arduino Leonardo, also pictured in *Figure 36*, initiated the scanning procedure. This would synchronize the Leonardo controlled RGB lighting with the C++ application to obtain an image of the embodied controller. The Leonardo would then cycle the RGB LED ring through the colour sequence.

Figure 36: An Intel NUC (left) and Arduino Leonardo (right) that powered the creation scanner

4.3 Motion Capture

4.3.1 VICON System

Figure 37: A powered VICON MX T40 camera suspended over the interaction volume

A VICON motion capture system composed of eight T40 cameras (pictured in *Figure 37*) was suspended above the interaction area and tracked individual BitDrones and interaction devices via IR reflective markers. They were networked with a VICON MX Giganet switch and connected to a 2010 iMac Pro with a Quad-Core Intel Xeon W3565 processor and 8 GB of RAM running VICON Tracker software. Tracker software the camera data and matched arrangements of markers to geometric patterns that were predefined in software. These distinct patterns were used to differentiate between trackable objects. Individual 3D marker positions of each marker pattern were then sent in VICON frame updates to BitDrones OS. The VICON reported marker positions based on a right-handed cartesian coordinate system as *xyz*-coordinates.

To avoid visual occlusion of the markers and increase accuracy, markers were always mounted at the highest plane of the drone frame to maximize visibility from as many cameras as possible. Occlusion was never an issue from surrounding drones due to the sparsity of their construction, but occlusion from users was a pervasive concern. Surprisingly, a user's clothing could drastically affect the system's operation. Some garments that appeared unremarkable in the visible spectrum were highly reflective in IR, most notably synthetic activewear.

4.3.2 Marker Patterns

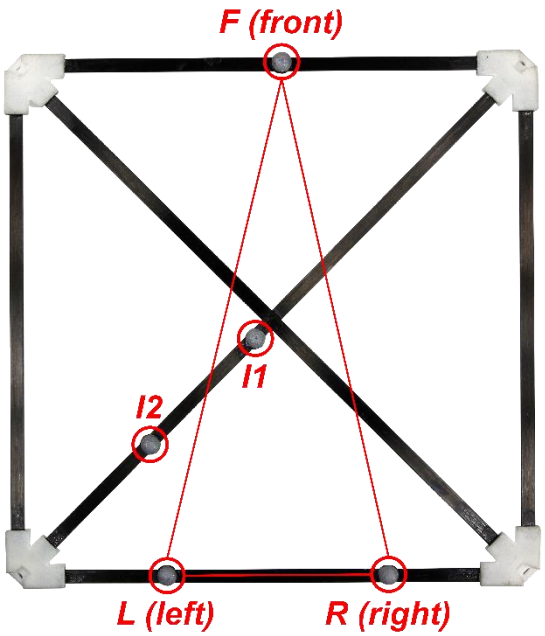


Figure 38: Drone top view showing its IR marker pattern with base isosceles triangular configuration drawn in red

The marker patterns of drones and interaction objects did not just provide their position but also their orientation. Each pattern was based on an isosceles triangle shown in *Figure 38* from which a forward pointing vector was obtained from discrete marker coordinates via *(1)* and *(2)* below, where $\boldsymbol{L}$ is the 3D coordinate of the left marker, $\boldsymbol{R}$ is the 3D coordinate of the right marker and $\boldsymbol{F}$ is the 3D coordinate of the front marker.

$$\boldsymbol{c_{rear}} = \frac{\boldsymbol{L} + \boldsymbol{R}}{2} \qquad (1)$$

$$\boldsymbol{c_{forward} = F - c_{rear}} \quad (2)$$

A symmetrical marker pattern on a plane would occasionally flip 180 degrees about the axis of symmetry because it was impossible for the VICON Tracker software to distinguish between symmetrically placed *Left* and *Right* markers. To prevent this, a fourth identification marker ***I1*** was added and a fifth marker ***I2*** was added for redundancy and to further differentiate between similar marker patterns. An orientation vector *($\boldsymbol{c_{orientation}}$)* perpendicular to the $\boldsymbol{c_{forward}}$ could then be found from *(3)*.

$$\boldsymbol{c_{orientation} = L - R} \quad (3)$$

The orientation of the object was stored in axis-angle form by computing the rotation matrices for the forward vector and orientation vector from the BitDrone's reference frame to the VICON's reference frame, which is discussed in detail in *4.5.2.2 Reference Correction*.

4.4 Network

As a real-time centralized system, it was imperative that the network operate with low latency. All devices save for interaction tools and the drones were connected via ethernet while the drones and interaction tools were connected to the network through an unsecured 2.4 Ghz WiFi network. Early versions of BitDrones were controlled through point-to-point RF connections which were soon abandoned in favor of IEEE 802.11 b/g/n WiFi. *Figure 39* visually describes the network structure based on a server running BitDrones OS.

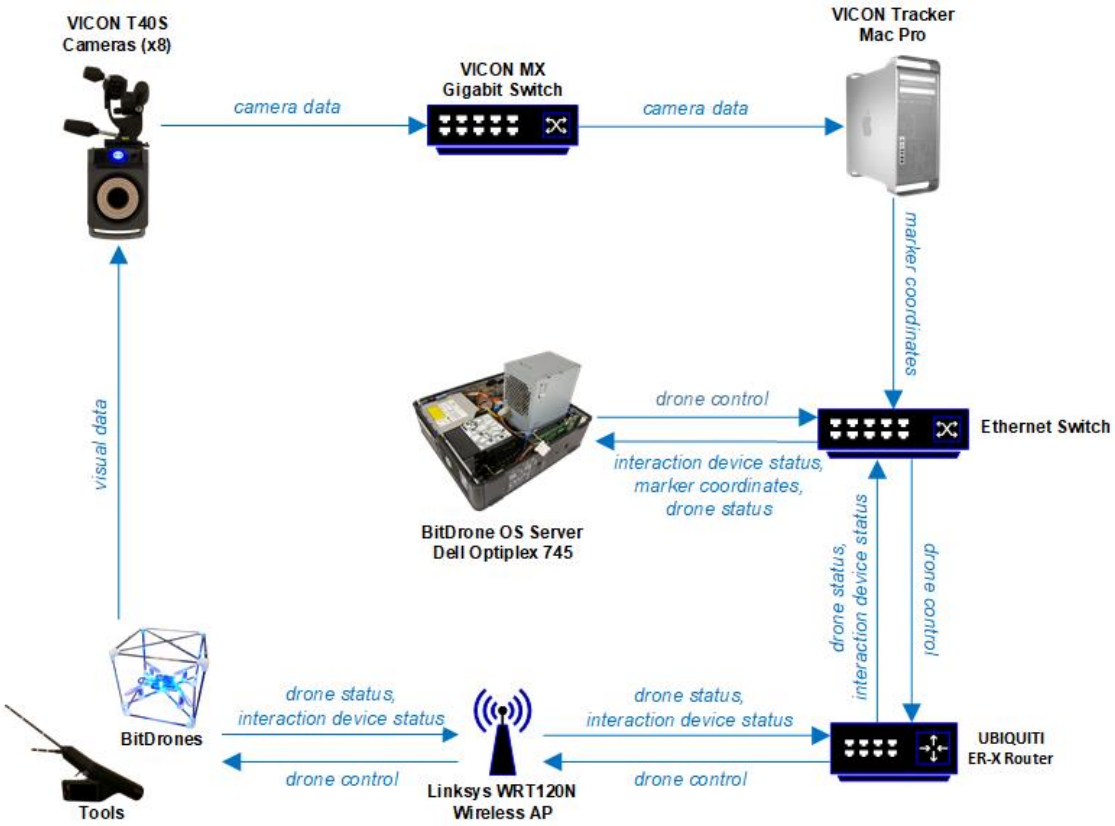


Figure 39: Flow chart showing the BitDrones network and data transfer

UDP was the preferred communications protocol that was employed in the interest of speed through unicast (transmission to a single recipient) and broadcast (transmission to all possible recipients) functions. Static IP addresses were used for all devices but were most important for drones and interaction tools because their marker patterns were paired with a corresponding IP address. This association was essential for BitDrones OS, in order to control the drones and to recognize tool input correctly. Static IP addresses also provided consistency which greatly aided in debugging.

Though UDP does not guarantee packet delivery, data would be quickly obsolete regardless and so this was considered an acceptable tradeoff in the event of occasional dropped packets. Despite the use of UDP, it was still necessary to run an unsecured network to increase communications speed. This is because it was found that if the network was password secured then authentication time between the AP and connected clients still produced noticeable delays

despite the use of UDP. This issue greatly hindered early development as it rendered the system physically unstable. To ensure some form of security of the network, the network name or service set identifier (SSID) was hidden.

Wireless network signal strength scans were performed on visible SSIDs to determine the channels in the 2.4Ghz 802.11 b/g/n band that would experience the least RF interference. Operation on a severely congested band would often cause minor in-flight disturbances but would occasionally cause catastrophic system failure. For this reason the least congested 802.11 b/g/n bands were favored.

4.5 Software

4.5.1 BitDrones OS

BitDrones OS was written in C++ and was served on a Dell Optiplex 745 running Ubuntu Server 18.04.1. It centrally controlled all aspects of the system which included drone flight, executing user interactions and serving various user applications. Multi-threading of various components of the OS was imperative due to the near real-time performance requirements of flight control. The high-level architecture and interaction between various system components of BitDrones OS can been seen in *Figure 40*.

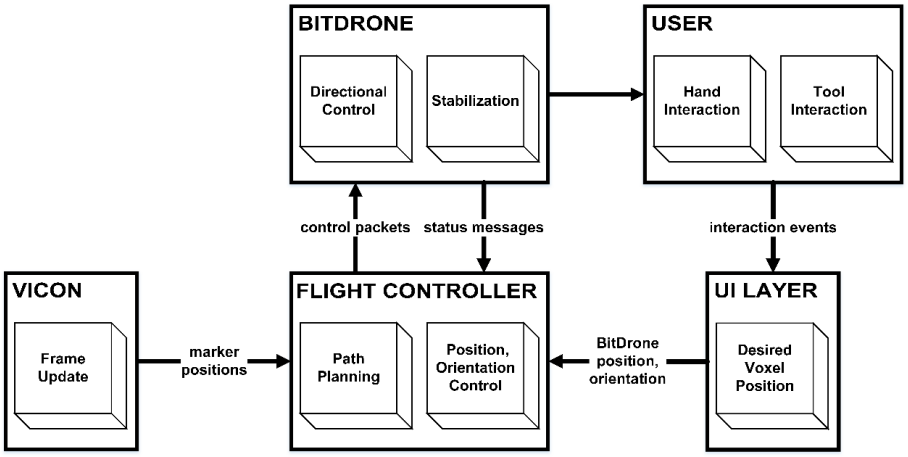


Figure 40: A diagram depicting a high-level interpretation of BitDrones OS

In BitDrones OS, each individual BitDrone was attached to a drone software object which governed and monitored the drones' physical behavior. A *UI Layer* was designed to run interaction scenarios based on the drone software objects and was implemented to make integration and testing of new scenarios faster and more efficient. The *UI Layer* determined desired drone positions based on the current running application and user input. These positions would then be passed to the *Flight Controller*, which governed the control of the drones, their physical positions and orientations in 3D, and performed rudimentary path planning based on system restrictions.

4.5.2 BitDrone Control

All drones were controlled directly from the BitDrones OS Flight Controller with multilayered PID (Proportional, Integral, Derivative) feedback loops that governed position, velocity, and orientation. Control data was delivered in the form of packets containing pitch, roll, yaw and thrust signals for individual drones.

As previously mentioned, the coordinated flight of all BitDrones was centrally controlled for two reasons. First, the drones did not have onboard positioning capabilities so positional updates would have to be sent to them in addition to desired positions, velocities and other behaviours. It would have been redundant to distribute control to the drones themselves as the drones would still require exceptionally low-latency information in the form of coordinates instead of direct control commands to position themselves. Second, the control system was developed from the ground up for this research initiative and required frequent tuning, feature changes and logging of over 50 parameters per drone for debugging purposes. These actions would have been made exceptionally more difficult and more time consuming through the implementation of a decentralized control network and would have resulted in a significant increase in wireless network traffic.

4.5.2.1 Multilayer PID Control

The control system was designed to fly the drones utilizing pitch, yaw and roll commands to rotate the drone about its principal axes, as well as thrust commands to control the magnitude of vectored thrust along the drones' *z*-axis. A dual-layer PID controller was utilized for pitch, roll and thrust, while a single-layer PID controller regulated yaw angle. The system generated digital values that mimicked those from a standard hobbyist radio controller for integration into the MultiWii firmware. PID control was chosen due to its computational simplicity, robustness and ease of implementation.

The *Yaw Controller* shown in *Figure 41* was a single-layer PID that minimized error between the current drone yaw and desired yaw setpoint. The proportional, integral and derivative gains (K_{PP}, K_{PI} and K_{PD} respectively) were tuned empirically based on logged system response data. Upper and lower limits were applied to the output of the *Yaw PID* before being packetized and sent to the corresponding BitDrone. The VICON frame update frequency of 100 Hz resulted in a system delay of about 10 ms, at which point the system received updated BitDrone position and orientation information.

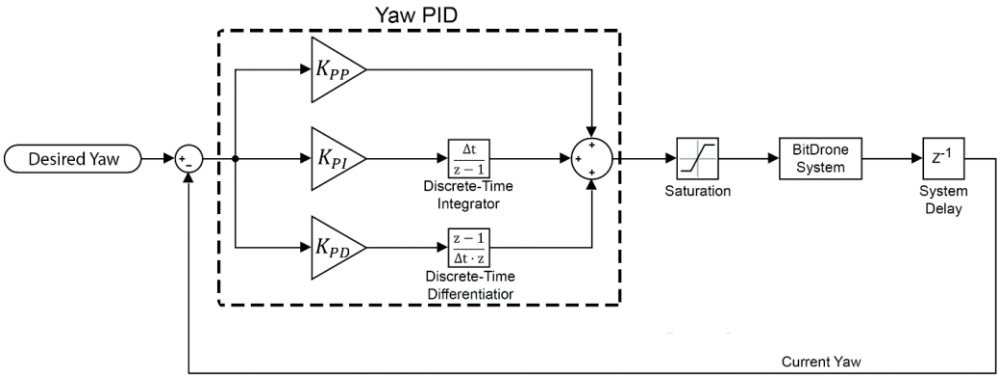


Figure 41: The Yaw Controller of an individual BitDrone

The *Pitch*, *Roll* and *Thrust Controller* shown in *Figure 42* was necessarily much more complicated than the *Yaw Controller* in order to achieve stable, precise control. It consisted of two, layered PID controllers, the first of which minimized positional error and the second of

which minimized velocity error. Each was structured similarly to the *Yaw Controller* in *Figure 41* and possessed qualitatively tuned gains and clamped output limits. Early BitDrones OS versions employed single-layered PIDs for pitch, roll and thrust control, but it was found that adding a velocity PID controller drastically improved performance.

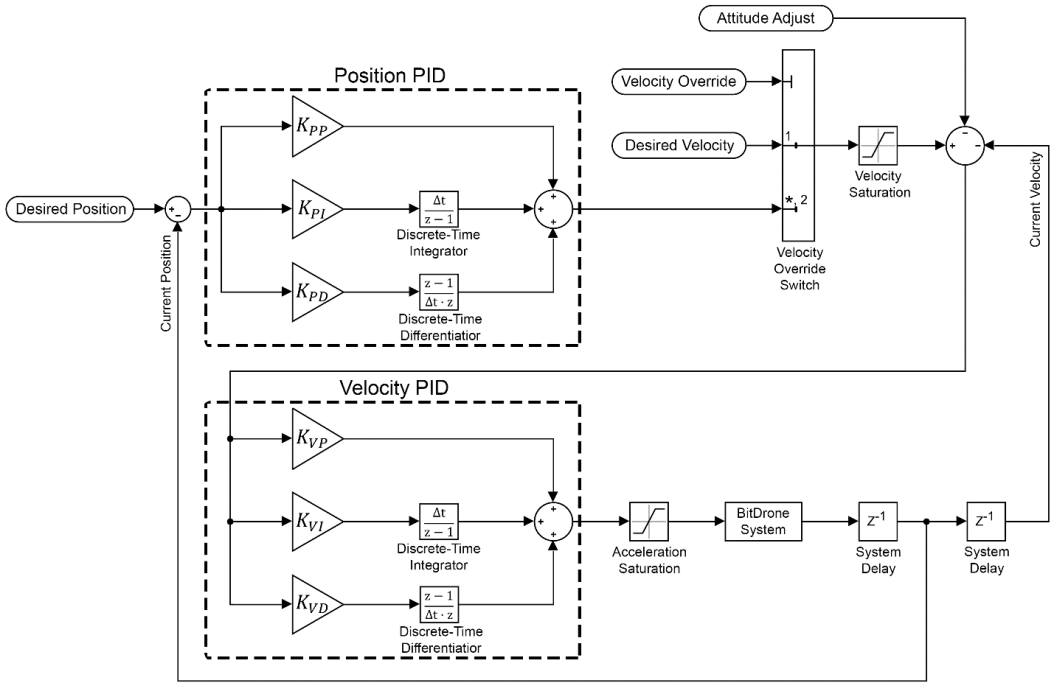


Figure 42: The Thrust Controller of an individual BitDrone

First, the *Position PID* was fed the positional error to output a desired velocity. This velocity setpoint could be overridden by the *Velocity Override Switch* in software so that the drones' velocity could be precisely controlled when necessary. This was done by simply scaling the velocity setpoint's unit vector. A velocity limit was then imposed on the magnitude of the desired velocity vector before calculation of error for the *Velocity PID*. The angle between the drones' normal vector and VICON *z*-axis was multiplied by a gain to determine an *Attitude Adjust* factor which was applied to the velocity error calculations of pitch and roll. This modified the velocity error to accommodate for pitch/roll overcompensation when the drone was already tilted in the desired direction of motion. The *Thrust Controller* did not possess this adjustment as staying

airborne was of highest priority, and the attitude adjustment was introduced to compensate for thrust's effect on pitch/roll.

To briefly demonstrate the capability of the system, *Figure 43* below shows a BitDrone's altitude over a window of time. The setpoint altitude was 1m and disturbances were introduced by forcefully hitting the drone with a hand while in-flight. The control system was exceptionally robust and could easily recover from significant disturbance such as being spun or hit at high velocity by masses in midair (including other Bitdrones).

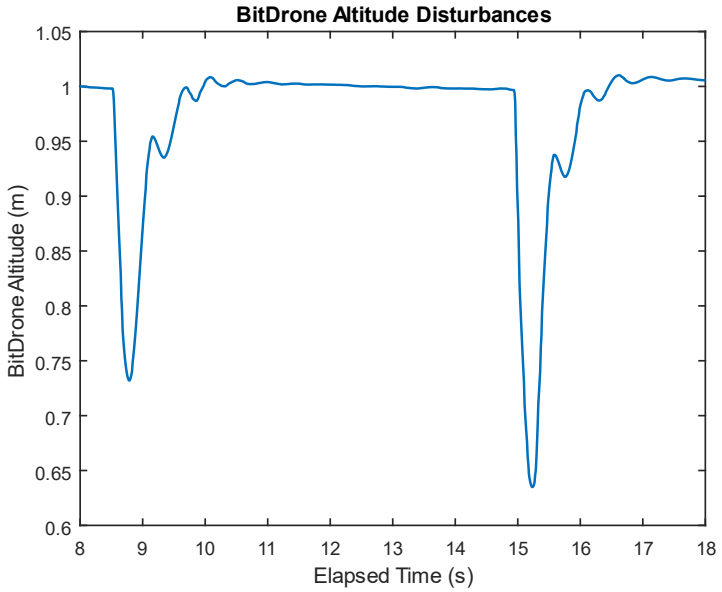


Figure 43: A BitDrone's altitude over time demonstrating physical response to disturbances

Figure 44 shows the response of the PI components of the dual-layered controller over the elapsed time of the second disturbance. The PI components defined the majority of the action of the control signals and were scaled as a percentage of their maximum values in order to show their combined behavior. Although there were somewhat significant oscillations in response to large disturbances, there was very little overshoot of position setpoints as shown in *Figure 43.* Observed oscillations resulted due to the *Velocity Controller's* regulation of the speed of the

drone. In general, the system was overtuned in order to be highly responsive to user input, which was important to perceive the swarm of BitDrones as a cohesive structure.

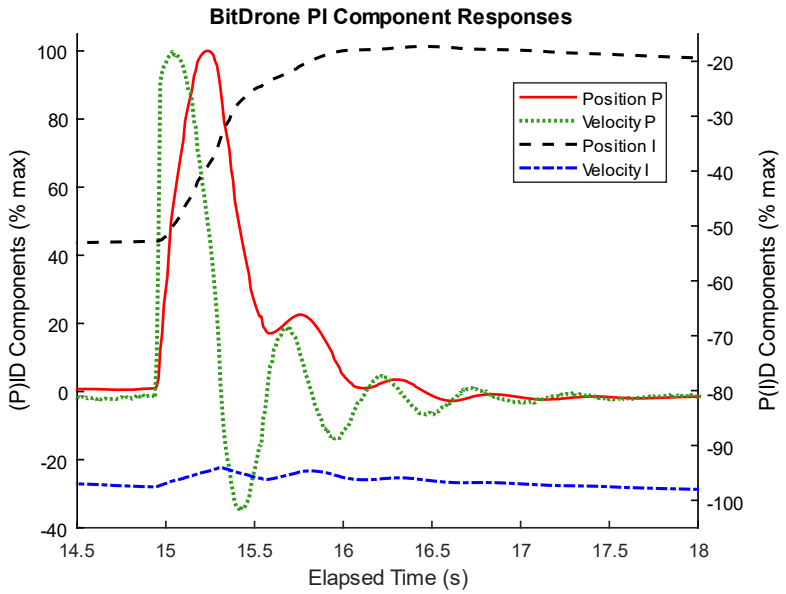


Figure 44: PI component response of a BitDrone's thrust controller over time in response to the second disturbance in Figure 43

4.5.2.2 Reference Correction

Pitch, roll, yaw and thrust commands were executed from the BitDrones' frame of reference. In order to calculate the correct drone control commands, it was necessary to rotate 3D coordinates from the VICON's reference frame to the BitDrones' reference frame as shown in *Figure 45.* The following reference correction applied to pitch and roll commands only. Yaw calculations utilized *xy* vector components exclusively, as the drones did not possess the capability for orientation invariant control e.g. being able to hover in any orientation. Thrust calculations were performed with single dimensional *z*-axis values. Initially, the inclusion of reference correction in the thrust calculations resulted in severe instability. Instead, they were modified according to copter orientation with the *Attitude Adjust* factor as described in *4.5.2.1 Multilayer PID Control*.

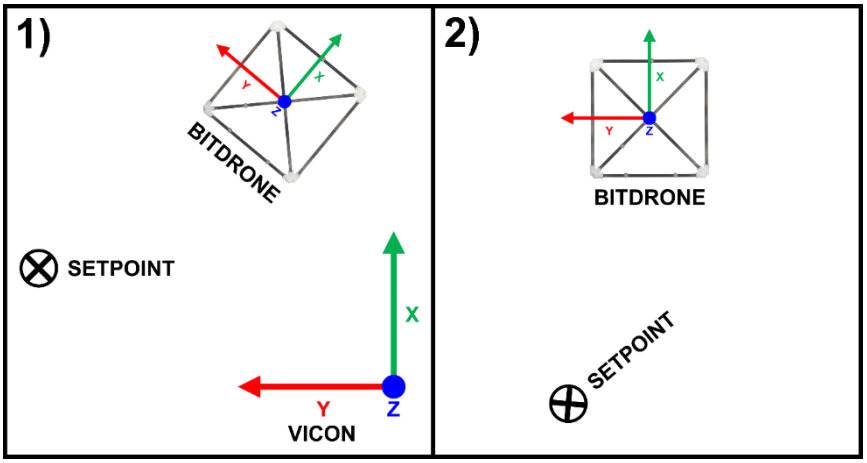


Figure 45: A BitDrone and its setpoint shown in the VICON's reference frame (1), and the same setpoint in the BitDrone's reference frame (2)

First, a rotation matrix $\boldsymbol{R}_{\boldsymbol{c}_{\boldsymbol{forward}},\boldsymbol{x}}$ was calculated via *(7)* that would align the drone's forward vector ($\boldsymbol{c}_{\boldsymbol{forward}}$) from *(1)* in *4.3.2 Marker Patterns* to a unit vector along the *x*-axis ($\hat{\boldsymbol{x}}$). The cross product *(4)* and dot product *(5)* of $\boldsymbol{c}_{\boldsymbol{forward}}$ and $\hat{\boldsymbol{x}}$, as well as the skew-symmetric matrix of $\vec{\boldsymbol{t}}$ *(6)* were necessary to calculate $\boldsymbol{R}_{\boldsymbol{c}_{\boldsymbol{forward}},\boldsymbol{x}}$.

$$\vec{\boldsymbol{t}} = \hat{\boldsymbol{c}}_{\boldsymbol{forward}} \times \hat{\boldsymbol{x}} \tag{4}$$

$$\vec{\boldsymbol{d}} = \hat{\boldsymbol{c}}_{\boldsymbol{forward}} \bullet \hat{\boldsymbol{x}} \tag{5}$$

$$[\boldsymbol{t}]_{\times} = \begin{bmatrix} 0 & -t_3 & t_2 \\ -t_3 & 0 & -t_1 \\ -t_2 & t_1 & 0 \end{bmatrix} \tag{6}$$

$$\boldsymbol{R}_{\boldsymbol{c}_{\boldsymbol{forward}},\boldsymbol{x}} = \boldsymbol{I} + [\boldsymbol{t}]_{\times} + \frac{[\boldsymbol{t}]_{\times}^{\,2}}{1 + \vec{d}} \tag{7}$$

Next, $\boldsymbol{R}_{\boldsymbol{c}_{\boldsymbol{forward}},\boldsymbol{x}}$ was applied to the drones' orientation vector ($\boldsymbol{c}_{\boldsymbol{orientation}}$) from *(2)* in *4.3.2 Marker Patterns* to calculate an intermediary vector ($\boldsymbol{c}_{\boldsymbol{orientation\_intermediate}}$) via *(8)*.

$$\boldsymbol{c}_{\boldsymbol{orientation\_intermediate}} = \boldsymbol{R}_{\boldsymbol{c}_{\boldsymbol{forward}},\boldsymbol{x}}\, \boldsymbol{c}_{\boldsymbol{orientation}} \tag{8}$$

Equations *(4)* through *(7)* were performed again to calculate a rotation matrix from $\boldsymbol{c_{o\_intermediate}}$ to a unit vector along the *y*-axis *($\hat{\boldsymbol{y}}$)* to obtain $\boldsymbol{R_{c_{orientation},y}}$. The multiplication of these two matrices via *(9)* yielded a rotational matrix that would rotate points from the VICON's reference frame to the BitDrone's coordinate frame.

$$\boldsymbol{R_{c,v}} = \boldsymbol{R_{c_{forward},x}}\,\boldsymbol{R_{c_{orientation},y}} \tag{9}$$

Next, a translation matrix was calculated which would be applied prior to $\boldsymbol{R_{c,v}}$ to find the centroid of the BitDrone frame. The copters' center was determined by multiplying a scalar with a normalized vector perpendicular to the plane formed by each BitDrone's marker pattern. This perpendicular vector was determined by *(10)* below.

$$\boldsymbol{C_{f,o}} = \boldsymbol{c_{forward}} \times -\boldsymbol{c_{orientation}} \tag{10}$$

Since the exterior frame of the drones was in the shape of a cube, its centroid lay at a distance of $L/2$ from each face, where L is the length of the cube's edge. $\boldsymbol{C_{f,o}}$ was normalized to obtain $\widehat{\boldsymbol{C}}_{\boldsymbol{f,o}}$, which was subsequently multiplied by $L/2$ and added to the center of the drone's marker pattern $\boldsymbol{c_{mid}}$ through *(12)*.

$$\boldsymbol{c_{mid}} = \boldsymbol{c_{forward}} - \boldsymbol{c_{rear}} \tag{11}$$

$$\boldsymbol{c_{centroid}} = \boldsymbol{c_{mid}} + \frac{L\widehat{\boldsymbol{C}}_{\boldsymbol{f,o}}}{2} \tag{12}$$

All 3D drone setpoints *(*SP*)* for pitch and roll command calculations, were translated by the $-\boldsymbol{c_{centroid}}$ vector and then rotated with $\boldsymbol{R_{c,v}}$ via *(13)* below.

$$\boldsymbol{SP_{corrected}} = \boldsymbol{R_{c,v}}(\boldsymbol{SP} - \boldsymbol{c_{centroid}}) \tag{13}$$

4.5.2.3 Obstacle Avoidance

Rudimentary obstacle avoidance was performed in order to minimize collisions between BitDrones, which could occasionally cascade through the system causing delays or complete

failure. Traditional swarm behavior was not implemented as the drones were meant to behave as individual catoms in a structure, remaining a fixed distance relative to each other despite the position and orientation of structure they composed. Basic path planning was usually only required at the beginning of complex and/or large constructions until the drones were in their appropriate position within the structure. A waypoint system was created to aid in coordination of the BitDrones in midair and was necessary to implement obstacle avoidance. This system allowed waypoints and target velocities to be queued and successively executed within a drone object in BitDrones OS. The drones were also unable to fly over or underneath each other due to turbulence from the drone above, so avoidance was performed in 2D.

C_r was a configurable parameter of BitDrones OS and indicated the radius of a virtual detection sphere around each drone. Coordinate or vector intersections with this sphere were used to determine a wide variety of factors, such as a whether a drone had reached a waypoint or if a user was interacting with a drone. *Figure 46* depicts three drones (red) within $1.5C_r$ of another drone's approach vector to a queued waypoint. Drones were considered obstacles if they were within $1.5C_r$ of an approach vector.

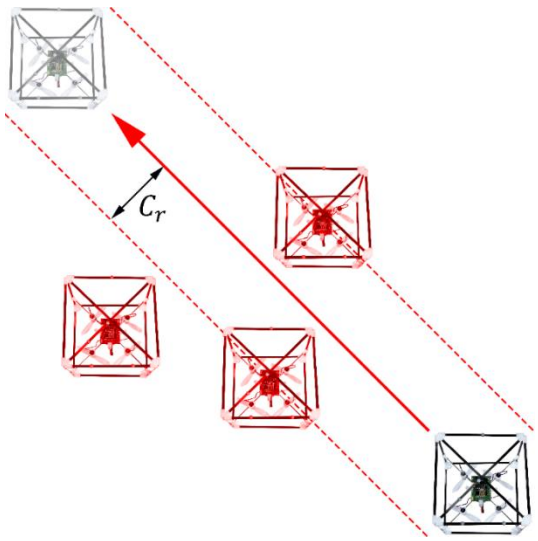
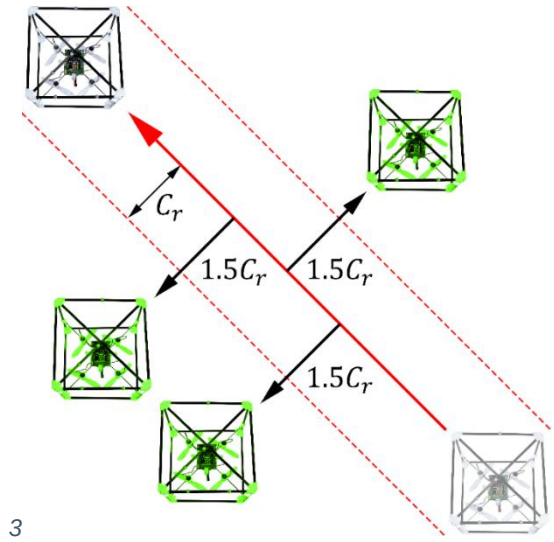


Figure 46: Three drones within $1.5C_r$ of a drone's approach vector

The avoidance method was simple, and due to the speed of the drones took relatively little time to execute. According to *Figure 47,* a perpendicular unit vector was found between the VICON *z*-axis and approach vector that indicated the desired direction of movement for impeding drones. Depending on which side of the approach vector the impeding drone resided on, the perpendicular unit vector was inverted and the offending drone was moved to a distance of $1.5C_r$ from the approach vector. Each drone object possessed a flag indicating whether it was moving to a regular waypoint, or a waypoint created for avoidance purposes. If peripheral drones blocked a drone moving to avoid impedance, then they simply moved along the same vector keeping a distance of C_r ahead. These waypoints then played back in reverse to close the path.



3

Figure 47: Drones moving $1.5C_r$ perpendicularly from another drone's approach vector

Due to the simplicity of this method, it was not appropriate for anything but the simplest cases of disorganization, as it was relatively slow compared to a parallelized strategy. Though it greatly reduced the number of collisions, occasional mishaps still occurred. These were of minimal concern with fully encaged BitDrones as they simply bumped or rolled off each other and quickly recovered.

4.6 Firmware

4.6.1 MultiWii

The BitDrone firmware was built on a heavily modified version of the fully-featured Multiwii open-source autopilot. As such, it is extremely complex and beyond the scope of examination in this thesis beyond some key features and modifications such as self-stabilization, safety features, colour control and communication packets.

4.6.1.1 Stabilization

MultiWii had the ability to control the attitude of the drone using the onboard IMU. The drone self-stabilized using onboard PID feedback loops, which were tuned manually for each BitDrone version. The control signals generated by BitDrones OS modified the attitude of the drones about its principal axes. Each time a drone was powered on, BitDrones OS automatically calibrated the onboard IMU before takeoff. If an IMU lost calibration, it would cause intense drift in the BitDrones operational volume which would strongly bias the control system and result in undesirable flight characteristics. Automatic calibration was important because the IMUs would lose calibration after crashes or even hard landings. This step ensured reliable operation of all drones without individual attention.

4.6.1.2 Safety Features

The largest contributors to system instability were communication latency and low update frequencies which usually presented at the same time due to wireless interference. The system was tolerant to a point, but severe instability caused spectacular crashes of all drones at high speeds when commands were temporally offset or were received below a minimum update frequency threshold. The drones were designed to be safe and unthreatening during normal operation, but these failures during testing often caused damage to the drones themselves and anxiety to uninitiated spectators/users during demonstrations and testing. To ensure graceful

failure, there was a kill command added to the firmware which immediately disarmed the drone if the command update rate dropped below 10 Hz, e.g. the time between packets exceeded 100 ms. As previously mentioned, 20 Hz was determined to be the optimal minimum command update rate for controlled flight, but it was observed that the system could function stably at 10 Hz albeit with decreased performance.

Another significant point of failure was the VICON tracking markers. If the IR reflective markers were occluded by a user or the markers were not within the VICON operational volume, this would cause the VICON system to report inaccurate 3D marker coordinates. If the calculated velocity of a drone was sufficiently high due to sudden significant changes in position or if it was found to be outside the operational volume then it was disarmed.

The rotation of the drones about the *x* and *y* axes were also monitored. If the drones exceeded a rotation of $\pm 50°$ about either of these axes then the drone was disarmed as it was presumed to have lost control.

4.6.1.3 RGB Control

To control the RGB illumination of individual BitDrones, additional commands were added to the MSP. These commands allowed for the control of any number of onboard RGB LEDs via a one-wire communication protocol. This enabled BitDrones OS to control the colour and light intensity of each BitDrone to grant them the appearance of illuminated voxels.

4.6.2 Communication Packets

The MultiWii Serial Protocol consisted of three packet types shown described in *Figure 48*. *Command Packets* were most frequently used to send real-time flight control data, RGB colour commands, and IMU calibration commands but could invoke additional functionality such as real-time tuning of onboard PID parameters. *Query Packets* requested specific data from a drone and were most often used to check onboard battery voltage and whether the drone was

armed or disarmed. *Response Packets* varied in size based on the requested data but could return information ranging from raw IMU data to individual motor control values from MultiWii. Checksums were used to ensure data integrity before command execution and were the product of a bitwise XOR operation between all bytes of the *payload info* and *command data* together.

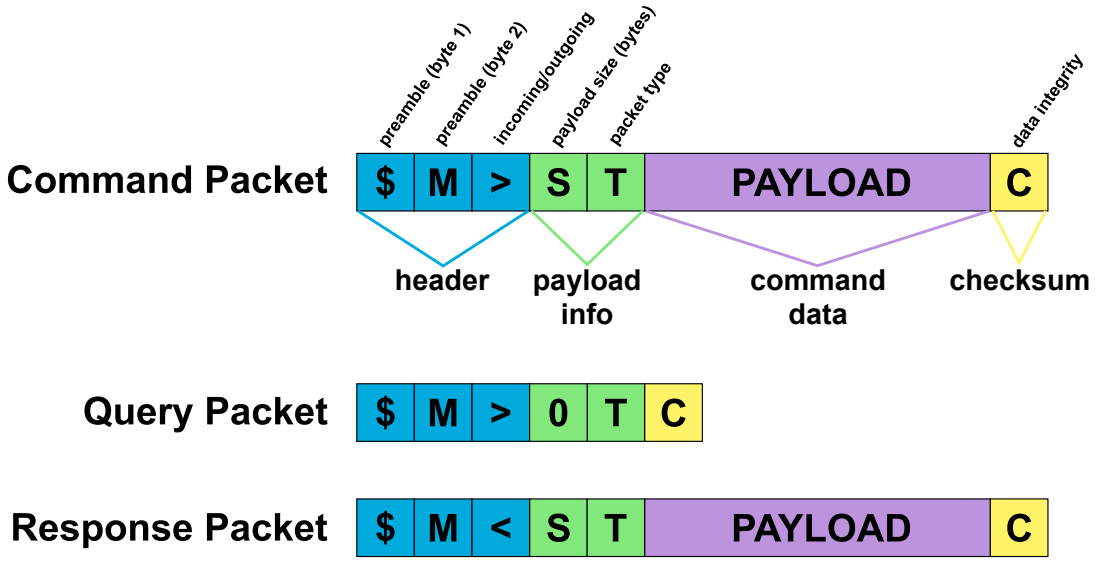


Figure 48: MSP packet types

The BitDrones OS packet wrapped *payload info* and *command data* for multiple drones into one large packet to address between one and twenty BitDrones, as shown in *Figure 49.* This packet would then be unwrapped by custom firmware running on the ESP-01 module and rewrapped for serial transmission to the flight controller in MSP format.

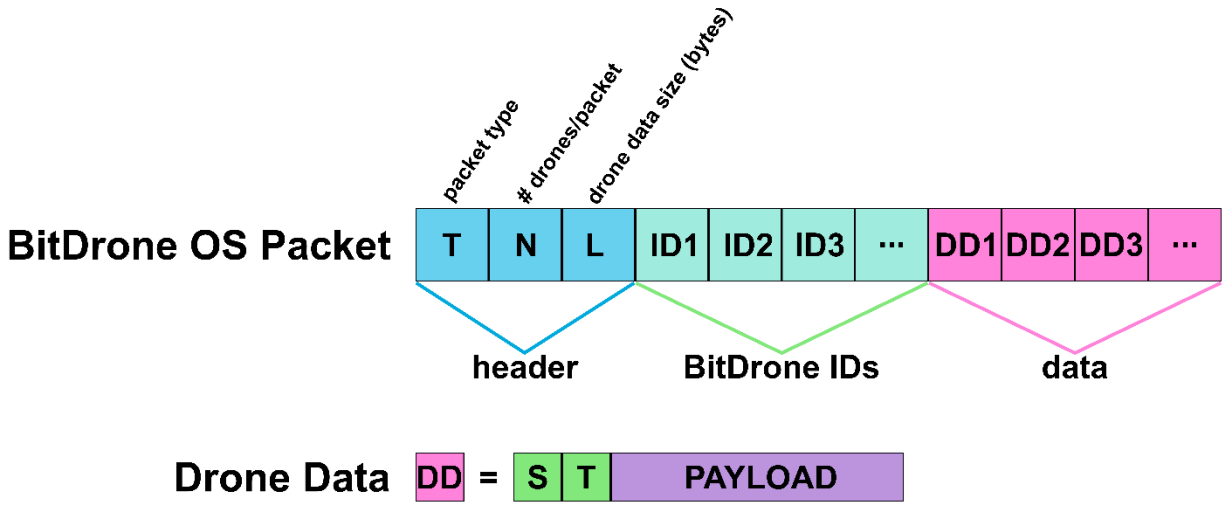


Figure 49: BitDrones OS command packet

4.6.3 Routing

The networking cost of sending individual packets was very high and as a result, addressing every drone individually was very slow. So slow in fact, that it was impossible to scale the BitDrone display to possess a meaningful resolution by increasing the number of BitDrone elements. The data size of the packet had a much lower impact on communications speed than the number of packets sent/received, so a new communication method was devised.

By combining the command information of up to twenty drones into one packet and sending a UDP broadcast to all drones, it was possible to achieve the update rate necessary for control over a Wireless-N 2.4 GHz network. BitDrone *Response Packets* had only one destination and were not as time-critical as BitDrone commands, so they were sent directly from each drone to the BitDrones server. The onboard ESP-01 would wait for a *Query Packet* from the BitDrones OS server and the origin IP of this broadcast was then saved as the destination IP for all outgoing packets from that BitDrone.

Custom firmware was created for the ESP-01 module to parse the packets, wrap the command data in MSP format and transmit it over a serial connection to the FC. Each drone first searched

the list of BitDrone IDs within a received *BitDrones OS Packet* to see if its ID was present using an effective but computationally simple binary search algorithm. If its BitDrone ID was found, the ESP-01 immediately calculated the bytewise offset within the *BitDrones OS Packet* where the corresponding command data was held according to *(14)*, below. It would then immediately copy the number of bytes as defined by *data size* at the calculated offset.

$$offset\ (bytes) = 3 + \#\ drones\ per\ packet + ID\ index \cdot data\ size \qquad (14)$$

The data was then packetized in MSP format and transmitted via serial at a baud rate of 115000 bits per second (bps). The ID index at which the drone identified its own ID was saved and was the first index checked in new packets. If the ID at the saved index did not match the drone's ID, then a binary search was initiated again.

Chapter 5. Exploration

5.1 BitDrones

5.1.1 Towards Self-Levitating Programmable Matter

BitDrones: Towards Self-Levitating Programmable Matter Via Interactive 3D Quadcopter Displays was an initial step towards the creation of programmable matter through the implementation of the first tangible interactive midair display [51]. A major goal was to bi-directionally synchronize virtual data with real physical catoms that possessed motility beyond what had previously been explored. Early tangible midair displays had limited mobility and interactive capabilities due to their magnetic [26] and acoustic [27] suspension techniques. Robotic swarm displays of the time either did not have interactive capabilities [35, 36, 37, 41, 42, 43] or could only function in 2D [28, 29, 30, 32]. It was proposed that catoms should be self-levitating by way of miniature autonomous drones (BitDrone V1s/PixelDrones) to ensure 3D operability.

The earliest BitDrone system utilized PixelDrones as self-levitating display voxels that could be manipulated in 3D via gestures or touch interactions, serving as a physical representation of virtual information. BitDrones ran as a custom C# application on Windows 7 while the VICON system tracked the drones as well marker patterns affixed directly to the user's hand and fingers. Users' "grab" gestures were detected when the distance between the finger marker patterns and hand marker decreased to a certain threshold. Simple molecular structures such as H_2O were created with three PixelDrones, where two drones represented hydrogen atoms in blue and the third represented an oxygen atom in red shown in *Figure 50*. This structure could be translated in 3D via a grab and drag gesture, and individual atoms could be removed from the molecule by tapping them with the hand. A tap was recognized when a user's hand was within a radius of a BitDrone then removed inside a certain window of time.

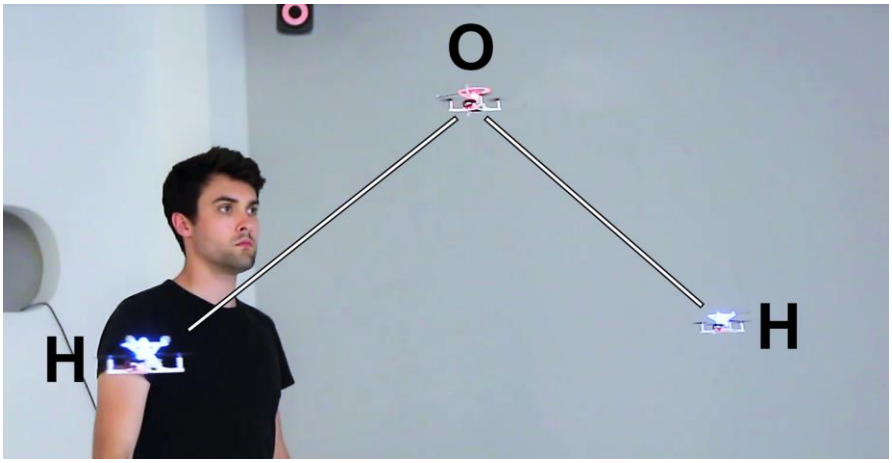


Figure 50: Three BitDrone V1s forming a water molecule (H_2O) [51]

BitDrones could also be used to represent simple mathematical functions, with individual drones serving as physical control points. The drones could render a function, while the same function was rear-projected on a screen behind for further context as depicted in *Figure 51*. Due to the overall unoptimized state of this early system only three voxels could be commanded at once. This low display density and low resolution meant that complex information could not be readily conveyed to users through BitDrones alone. Additionally, the drones were not completely tangible due to the limited contactable surface area and exposed propellers, which was not conducive with the design principles of programmable matter [2].

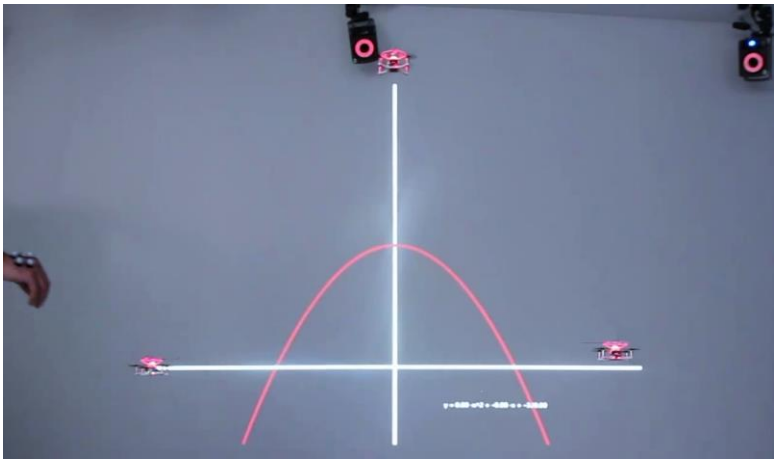


Figure 51: Three V1 BitDrones acting as physical handles to a parabolic equation [51]

5.1.2 Nanocopter Displays as Interactive Self-Levitating Programmable Matter

BitDrones: Towards Using 3D Nanocopter Displays as Interactive Self-Levitating Programmable Matter [1] focused on the implementation of a toolbox for drone-based real-reality interfaces and ran the first version of BitDrones OS. It utilized three different types of drones to address tangibility and resolution issues of the previous system, as well as increased functionality through the introduction of new interaction techniques. In this exploration, tangible interactions were prioritized and notable UI elements such as cone tree menus, compound structures and sub-voxel imaging techniques were implemented.

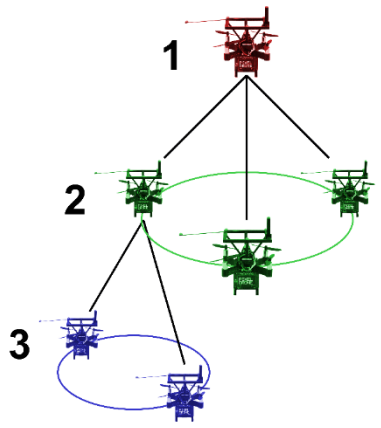


Figure 52: PixelDrones in a three-layer cone tree menu

3D cone trees were realized with second generation PixelDrones to browse through files, file systems and menus. Text could be displayed on their small onboard OLEDs indicating filenames, folders and menu options which allowed for greater information density. Each tier of the cone tree represented a successive layer of the system to navigate as shown in *Figure 52*. Physical limits of the operational volume only allowed up to two layers to be rendered at a time. Tapping a drone with a hand would select a file, option or submenu, while touching and dragging a drone would spin that layer with a decaying velocity to bring other options within reach.

ShapeDrones were designed to embody the traditional cuboid form factor of a voxel and to offer increased tangibility through larger surface area. Their carbon fiber and ABS frame shielded the spinning propellers within, and a synthetic mesh acted as a diffuser for the onboard LED in order to further resemble a cube. ShapeDrones could be easily grasped with one hand as well as touched from the sides thanks to the enclosing frame in combination with the diffusing mesh.

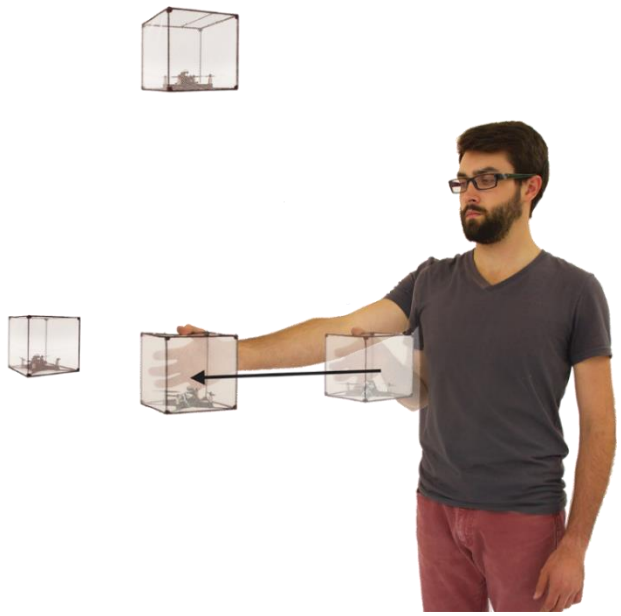


Figure 53: A user performing a drag operation to reposition a ShapeDrone [1]

Unimanual and bimanual interactions were also possible with ShapeDrones as inspired by previous tangible and gestural interactions with 2D tabletop displays and robotic swarms [10, 32, 50]. Individual drones could be unimanually tapped to elicit certain behaviors, as well as dragged to various positions to create compound structures from multiple drones such as in *Figure 53*. A drag operation was initiated when the user's hand was within a certain radius of a drone for longer than the time window which specified a tap. Physically dragging the drone in 3D and releasing it would set its new position in a structure.

Bimanual gestures on the other hand were used to translate, rotate and resize compound structures. Translation and resizing actions were inspired by pinch-to-zoom and two-finger pan

techniques respectively, which are employed by current touch-based UIs. Grabbing and dragging two drones of a compound structure would translate the structure in 3D, with all drones moving to keep the same relative positions to each other. To resize a structure, the user grabbed two drones, dragging them closer together or further apart to scale the entire compound structure accordingly as depicted in *Figure 54*.

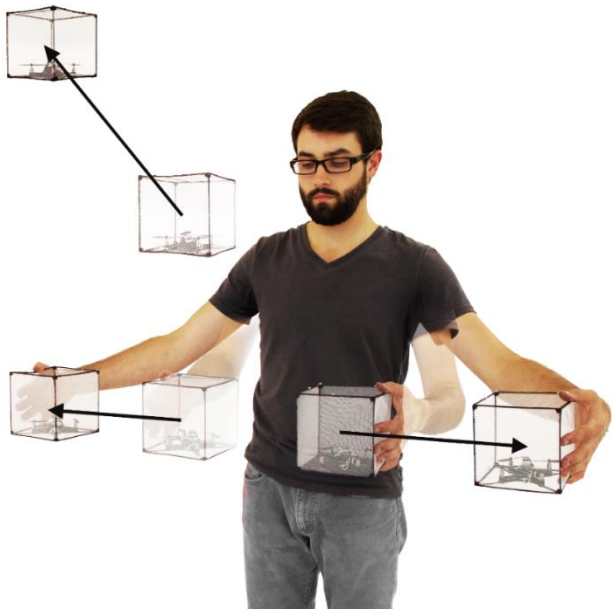


Figure 54: A user bimanually resizing a compound structure [1]

A tangible rotation technique was also implemented for compound structures. First, the user grabbed a drone which would define the axis of rotation for the model. This rotational axis extended from the drone along the *z*-axis. The user would then grab a second drone in the compound structure and drag it around the first drone in the desired direction of rotation. The entire structure would then rotate around the first drone the same number of radians that the second drone was dragged.

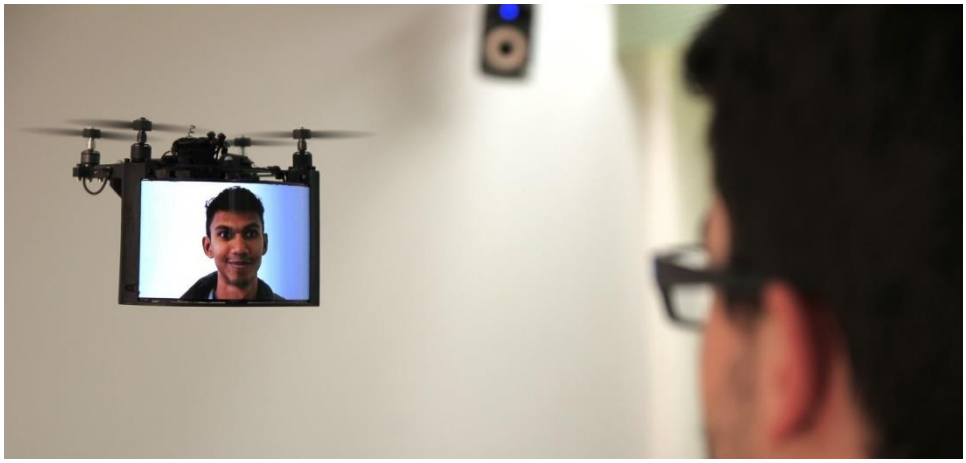


Figure 55: A BitDrones user interacting with a remote user with a DisplayDrone

More drones could be commanded simultaneously with the switch from XBee S2s to WiFi XBees, however the system could still only reliably support a maximum of eight drones in flight. This resolution was still too low to convey complex information, so DisplayDrones carrying high-resolution curved FOLED screens were created. These drones provided a contextually relevant personal display to augment the system, similar to the prototyped described by Schneegass et al. [44]. The DisplayDrone could run Android applications for teleconferencing (*Figure 55)* as well as be used to alter non-physical parameters of individual voxels or compound structures such as colour (*Figure 56*) via the onboard touchscreen. The display was integrated into a drone so that it could remain within easy reach of users' hands for touch interaction and within a user's field of view for collaborative teleconferencing.

Figure 56: A DisplayDrone running a custom Android colour palette application

Although more drones could be controlled simultaneously, the resolution of BitDrones was still far too low to physically convey complex information to users. Drones carrying displays did not possess the payload capabilities to support an exterior frame, so their tangibility was still limited for the same reasons as the first PixelDrones (BitDrone V1s).

5.2 GridDrones

5.2.1 A Self-Levitating Physical Voxel Lattice

GriDrones: A self-Levitating Physical Voxel Lattice for Interactive 3D Surface Deformations [52] had several major goals beyond previous BitDrones explorations:

1. To increase the resolution of BitDrones.
2. To increase the display density of BitDrones.
3. To create a UI that was not impacted by the drones' inability to occupy the same *xy*-coordinates.
4. To govern BitDrones by user-defined material properties.

Increasing the resolution of BitDrones was a multi-faceted challenge which involved significant optimization of the wireless network as well as parallelized routing in order to achieve the update rate and low latency required for the control of many drones. Increasing the display density necessitated the design of more advanced drones, a better control system and the design of a new BitDrones OS in C++ for implementation on a Ubuntu server for better real-time performance. Upgrade cycles continued throughout the development of *GridDrones*, starting with BitDrone Version 3 and ending with Version 5.1. These improvements granted the drones the agility and accuracy to fly in tight formations.

BitDrones could not occupy the same *xy*-coordinates due to being caught in the thrust streams of drones overhead. Unfortunately, this was simply a limit of the platform chosen as the basis for catomic elements in this research. In order to provide a more intuitive UI that would accommodate for this shortcoming, BitDrones were arranged in a grid pattern on a plane with arbitrary spacing between units, where no two drones could occupy the same grid coordinates. When individual units were freely arranged by a user, they would snap to grid coordinates which extended to infinity in all directions from the VICON origin. BitDrone elements would remain in

their relative positions in the grid despite transformations imposed by the user. BitDrones were operated at sufficient grid density so that their formations could be treated as a programmable matter material. The user could then define the stiffness of the GridDrone lattice so that this material could be tangibly modelled with realistic physical responses as depicted in *Figure 57*.

Figure 57: A user tangibly deforming the GridDrone lattice by pushing a corner downwards

From previous BitDrones explorations, it was found that many of the bimanual interaction techniques proved problematic when dealing with physically large and/or dense structures. An example presented when a user attempted to bimanually to rotate a large structure 90 degrees about the *z*-axis and the drones rotated into the user, colliding with them. Another example was when BitDrone elements were spaced too far apart to bimanually manipulate, or elements were so distant that it was laborious for users to physically move between them to initiate unimanual interactions. To eliminate physical impediments to the manipulation of the GridDrones system, the Wand was designed to augment tangible interactions.

GridDrone elements could be grouped in several ways in order to physically manipulate the group as a whole, or to change intangible aspects like their colour or stiffness. Groups of drones

could be moved vertically in the *z*-axis but were constrained to their relative *xy*-axes positions in the grid. The first way that users could select drones was through a unimanual discontiguous selection by simply tapping the drones sequentially as shown in *Figure 58*. The grouped drones would highlight upon being selected and the grouping was confirmed and saved with a press of the secondary wand button. All drones in the lattice could be selected by double tapping any drone.

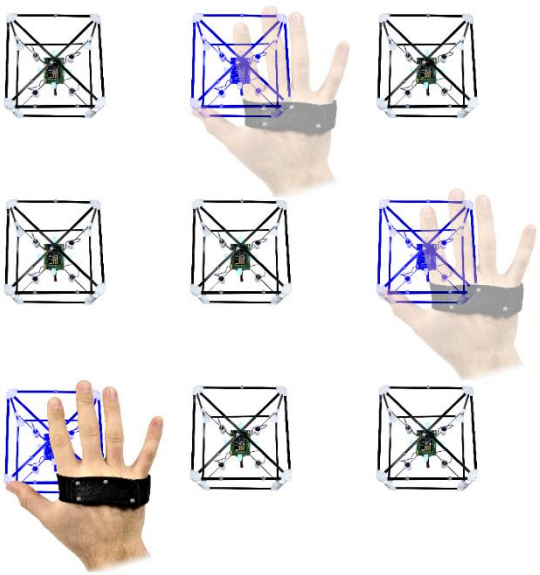


Figure 58: A user unimanually selecting muliple GridDrone elements

A larger area of the GridDrones lattice could be selected similarily to drag-select techniques found on traditional touch and mouse-based GUIs. Such techniques would select all elements within a rectangular perimeter, its two diagonal corners defined by the start and end positions of a click and release action respectively. By utilizing the left and right hand simultaneously to define the corners of a rectangular selection with a tap, users could select a 2D area of the GridDrones lattice as shown in *Figure 59*, and confirm that group with a press of the secondary wand button.

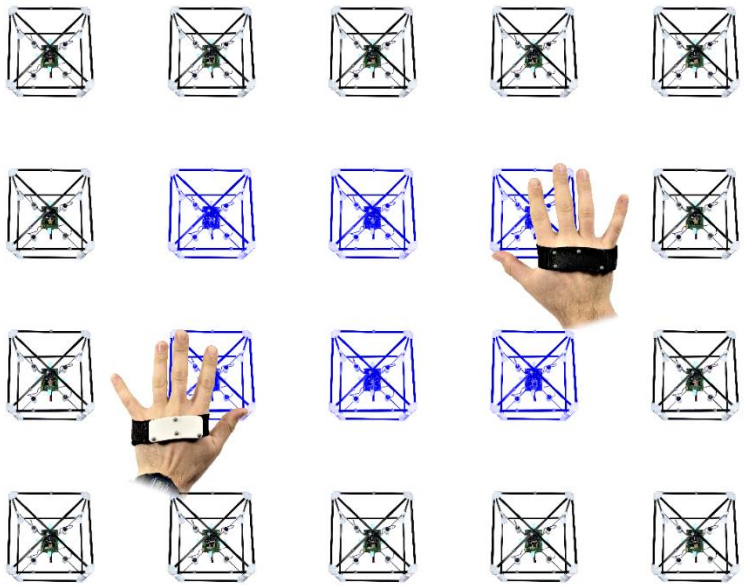


Figure 59: Bimanual selection of a group within the rectangular area defined by the left and right hand

The wand was used to select larger numbers of drones that would otherwise be difficult or impossible to select manually. By utilizing a virtual ray casting technique by projecting an infinite vector from the main axis of the wand, this ray could be used to select multiple drones by painting them with the virtual ray. Drones could also be grouped within a bounding path created by the ray in a lasso-select as depicted in *Figure 60* .

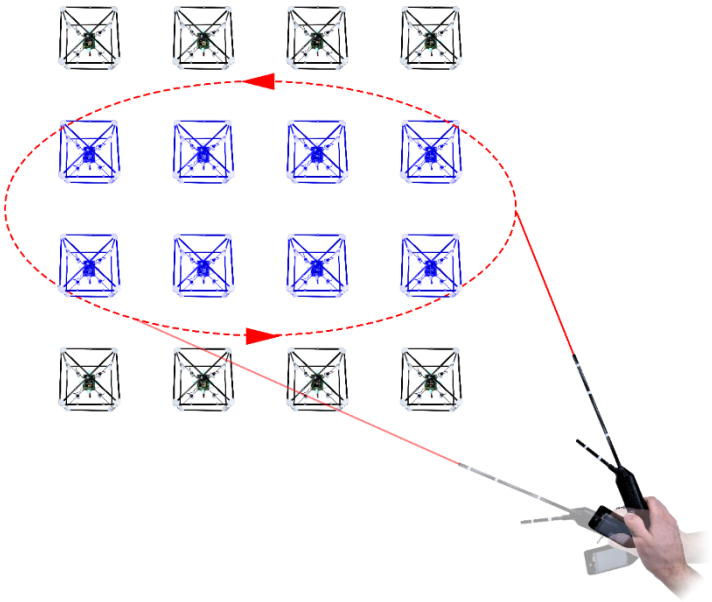


Figure 60: A user lasso selecting a group of drones with the wand

To paint the drones, the shortest distance from the infinite ray vector to each drone was calculated in 3D via *(15)* below, where $\boldsymbol{c_{centroid}}$ was the center of the BitDrone, $\boldsymbol{W_f}$ was the 3D position of the wand's front vector marker and $\boldsymbol{W_b}$ was the Wand's rear vector marker. Any drones within an adjustable distance of the Wand's ray were selected.

$$distance = \frac{\left|\left[\boldsymbol{W_f} - \boldsymbol{W_b}\right] \times \left[\boldsymbol{c_{centroid}} - \boldsymbol{W_b}\right]\right|}{\left|\left[\boldsymbol{W_f} - \boldsymbol{W_b}\right]\right|} \qquad (15)$$

Lasso selection was performed by recording samples of the Wand ray's intersection with the GridDrones lattice plane over time. First, the user initiated a lasso select by clicking the Wand's primary button and pointing the Wand at the intended start point of the lasso select in empty

space. Next, plane intersection points were sampled over time until the user released the primary button according to *(16),* where $\boldsymbol{P_n}$ denoted the lattice plane's normal vector and $\boldsymbol{P_o}$ was the plane's origin location in the VICON reference frame.

$$\boldsymbol{intersection} = \boldsymbol{W_b} + \left[\frac{-\boldsymbol{P_n} \bullet [\boldsymbol{W_b} - \boldsymbol{P_o}]}{\boldsymbol{P_n} \bullet [\boldsymbol{W_f} - \boldsymbol{W_b}]}\right] \cdot [\boldsymbol{W_f} - \boldsymbol{W_b}] \qquad (16)$$

A line scanning method was used to scan horizontally and vertically across the GridDrone plane between intersection points to determine if a drone lay within the boundaries defined by the ray's path. It was not a true 3D selection method, but rather a 2D selection method on a plane rotated in 3D. This was done to increase the accuracy of the lasso selection method. The GridDrone lattice was only one drone layer thick, which meant that projections closed by the lattice's plane would only ever bound one layer of drones instead of potentially infinite drones if projected along the *x*-axis or *y*-axis.

Individual elements could also be selected and repositioned via the Wand. A drone was temporarily selected as long as it was intersected by the wand's ray, where intersection was determined by a drone's distance from the Wand's ray as found by *(15)*. If the user clicked the Wand's primary button and rotated or translated the Wand, the drone would follow the Wand's ray at a fixed distance, demonstrated below in *Figure 61.*

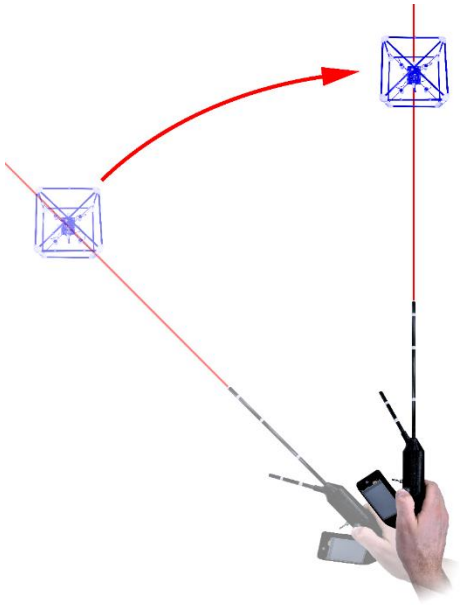


Figure 61: A BitDrone being repositioned by a user via raycasting with the wand

The Wand could also select multiple drones through intersection with its ray. These drones could be grouped with the wand's secondary button or repositioned simultaneously in the same way as a single drone by clicking, dragging and releasing with the wand's primary button as depicted in *Figure 62*.

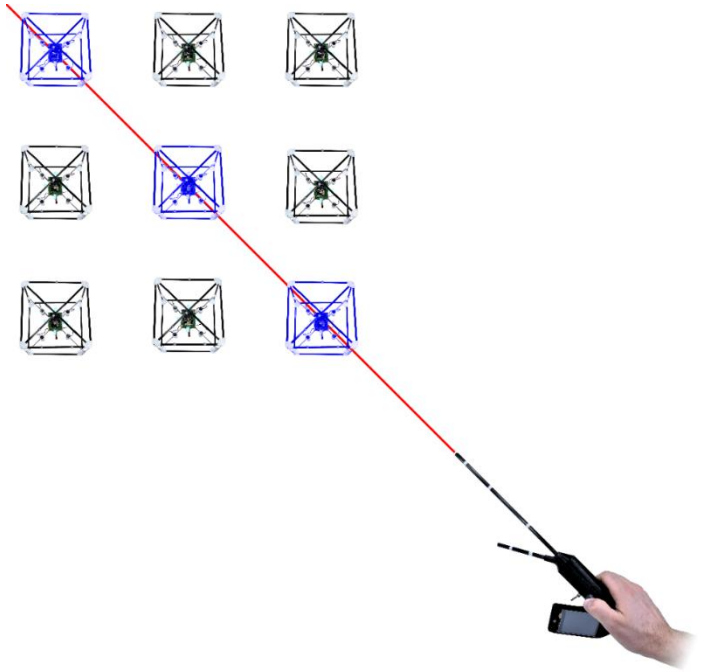


Figure 62: A user selecting multiple drones along the wand's ray

The entire GridDrone lattice could be freely transformed in 3D by clicking the wand's primary and secondary buttons simultaneously and moving the wand. Moving the wand in the desired direction of motion would constitute a translation as shown in *Figure 63* while reorienting the wand would constitute a rotation as depicted in *Figure 64*. The relative transformation of the wand in the VICON frame from its initial position and orientation was constantly calculated and applied to the entire lattice until the user released the wand buttons. Translations and rotations were applied in the VICON's frame of reference, but rotations were centered around the origin of the lattice. This allowed the orientation of the lattice to be manipulated while it remained in place unless translated by the user.

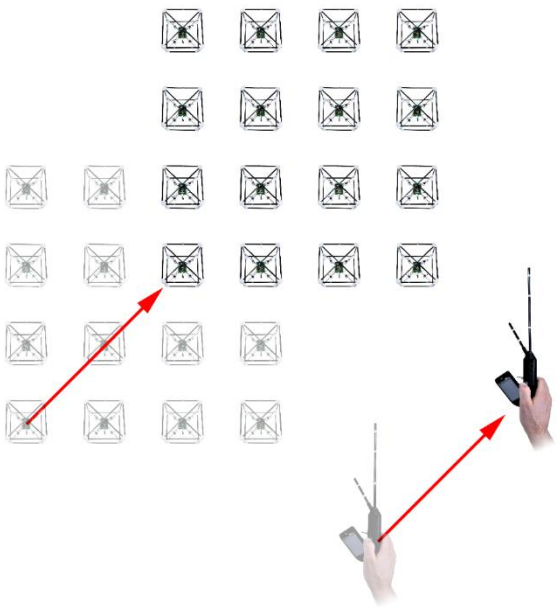


Figure 63: A user translating the entire GridDrone lattice with the wand

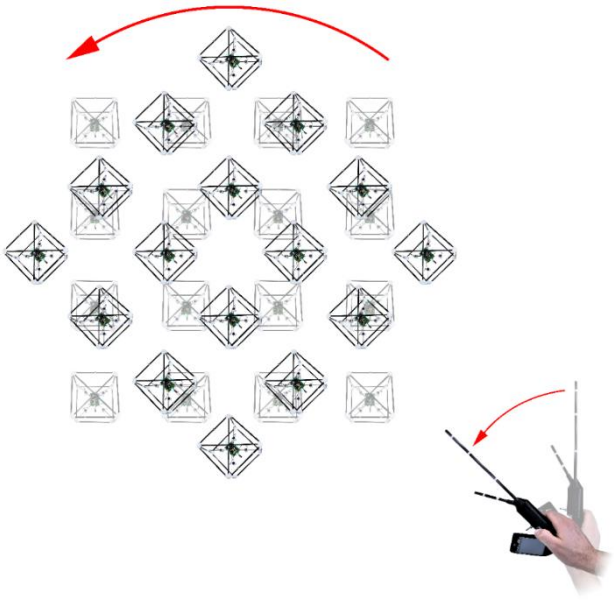


Figure 64: A user rotating the entire GridDrone lattice about its origin with the wand

A novel feature of the transformations permitted by the Wand was the ability to rotate the GridDrone lattice about arbitrary vectors defined by the Wand's ray. The Wand's rotation θ about its pointing axis was found, and a rotation matrix $\boldsymbol{R}$ was generated from this rotation and the Wand axis unit vector $\boldsymbol{W_v}$ according to *(17)*.

$$\boldsymbol{R} = (\cos\theta)\boldsymbol{I} + (\sin\theta)[\boldsymbol{W_v}]_\times + (1 - \cos\theta)[\boldsymbol{W_v W_v}^{\boldsymbol{T}}] \quad (17)$$

The lattice's origin was virtually translated to the VICON origin where $\boldsymbol{R}$ was applied, then the lattice was translated back to its original position. When enacted, the lattice would rotate about a vector that intersected its origin and was parallel to the Wand's ray as shown in *Figure 65*. Twisting the Wand about its pointing axis was desirable in many instances because a user has a greater range of motion when rotating their wrist about their forearm versus flexing their wrist up/down or left/right.

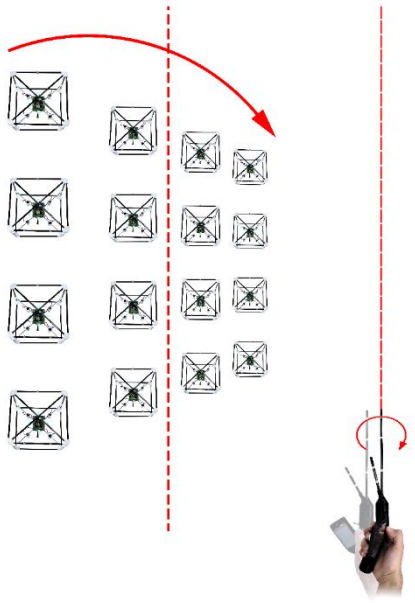


Figure 65: A user rotating the GridDrone lattice about the wand's pointing vector

The illumination colours of groupings within the GridDrone lattice could be altered via RGB sliders from 0-255 on the Wand interface as seen in *Figure 66 (right)*. These sliders would update dynamically to both reflect and apply the colour settings of the selected group. A different settings screen in the wand interface also provided a slider from 0-100% to adjust the stiffness of a group also shown in *Figure 66 (left).* The stiffness of a GridDrones grouping specified the magnitude of relative movement of adjacent BitDrones in response to user input on one or more drones.

For example:

- A 0% stiffness relationship meant that grouped drones had no connection to adjacent drones and could all be moved completely independently by the user.
- A 50% stiffness relationship meant that grouped drones moved half of the magnitude of adjacent drones. This relationship radiated uniformly outwards from drones that were being interacted with.
- A 100% stiffness relationship meant that grouped drones had a “solid” connection and moved together as a unit, even if just a single drone in the grouping was being interacted with.

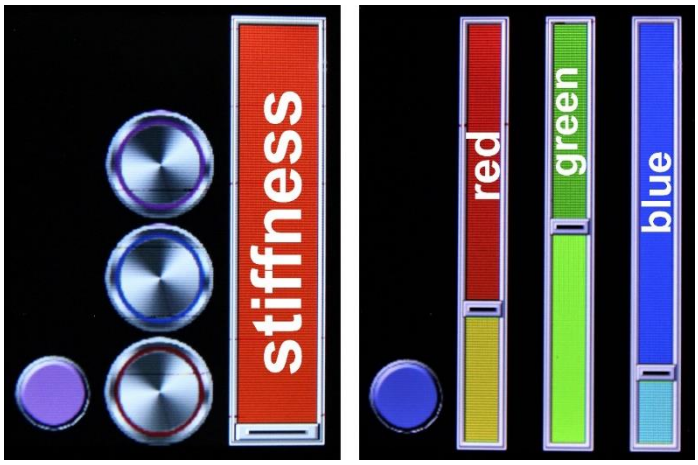


Figure 66: The wand’s interface showing the stiffness slider (left) and RGB sliders (right)

During drag operations the desired velocity of the BitDrones was set to 0 m/s. The drones resisted applied motion in all directions and provided tangible feedback when sculpting compound structures. This was a very basic method of drone admittance similar to that described by Augugliaro and D’Andrea [49] and impedance could be dynamically altered through manipulation of the PID gains. Groups with different stiffnesses could also be overlapped, which could result in interesting scenarios such as “rigid” groups existing in the middle of “soft” groups. *Figure 67* and *Figure 68* depict a pyramid construction from a stiff and flexible material respectively by manipulating the center drone.

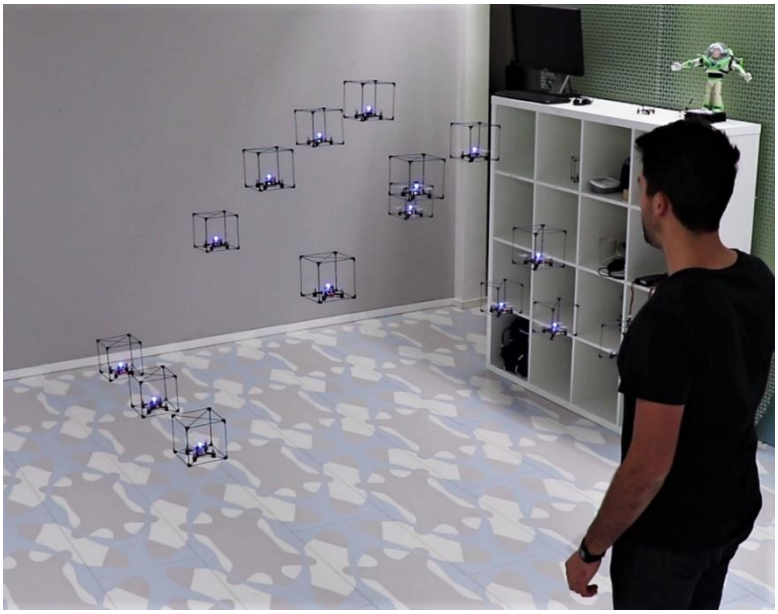


Figure 67: A user constructing a pyramid shape with a stiffer material relationship

Figure 68: A user constructing a pyramid shape with a softer material relationship

A common critique in the form of peer reviews from the academic community was that although the Wand afforded increased utility to the system, the reliance on a gesture-based tool subtracted from the tangible nature of this rudimentary programmable matter system. For this reason, increased effort was taken to replace Wand interaction modalities with tangible ones.

Global bimanual 3D translation and resize operations on the GridDrone lattice were implemented in the same way as in BitDrones, shown in *Figure 69* and *Figure 70* respectively.

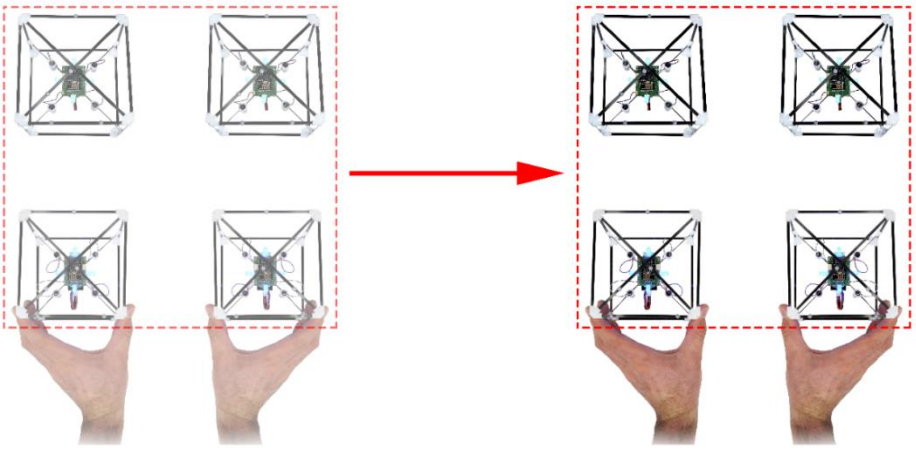


Figure 69: A user bimanually translating the GridDrone lattice

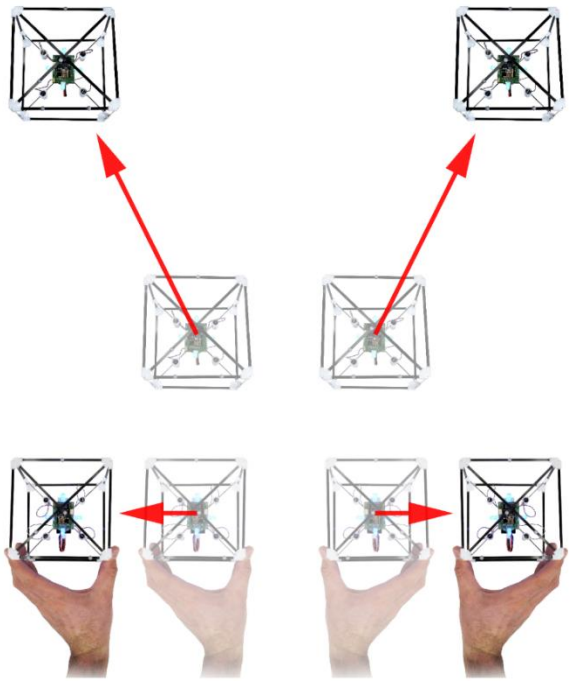


Figure 70: A user bimanually resizing the GridDrone lattice

Bimanual rotation of BitDrones was also redesigned to be less cumbersome for the user. Instead of one stationary drone indicating the axis of rotation, both drones could be freely moved. The vector between the two drones upon grasping them indicated the starting orientation vector which was saved, while the current orientation vector between them was updated continuously. The relative angular difference between the starting orientation vector and current orientation vector was applied as a rotation matrix about all axes to the entire lattice as depicted in *Figure 71*. Translations, rotations and resize operations occurred simultaneously just as they would when working with real material, making global transformations extremely intuitive.

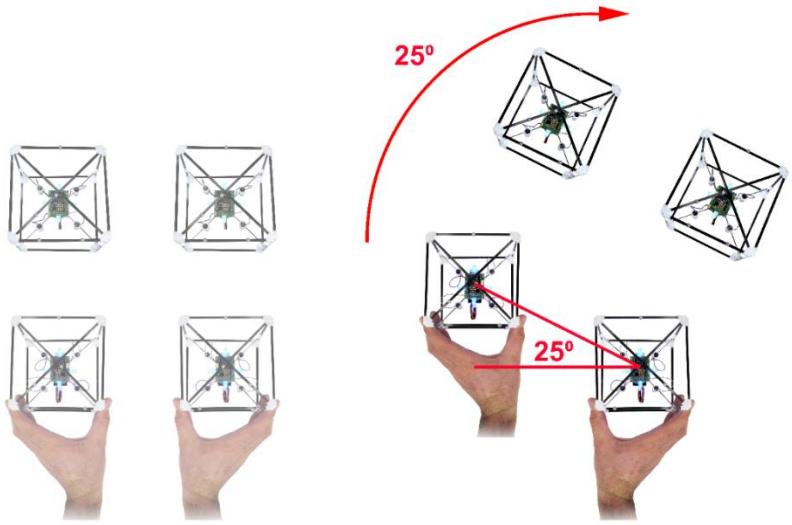


Figure 71: A user bimanually rotating the GridDrone lattice

To avoid previous issues with bimanual manipulation, features of the Wand such as ray casting and gestural interactions were transferred to users' hands and fingers. This required three additional markers to indicate a vector extending outwards from the index finger, and three additional markers to indicate a similar vector on the thumb, visible in *Figure 72*. A ray was cast from the *Pointing Vector* for all ray casting interactions, while the functionality of the primary Wand button was replaced by motion of the user's thumb. When the angle between the *Thumb Vector* and *Pointing Vector* fell below a certain threshold, a "click" action was performed.

Figure 72: Right hand showing the hand pattern strap, pointing vector and thumb vector

In order to control non-tangible aspects of the system such as colour and stiffness, a custom smartphone app was used with sliders that mimicked those in the original Wand GUI. The Wand's secondary button functionality was also transferred to this smartphone app and new control buttons, sliders and switches were added and removed as necessitated during further prototyping. The reasoning behind utilizing a smartphone in order to control non-tangible system parameters was because it was much simpler to develop a prototype and was deemed less intrusive than a purpose-built interaction device.

The last feature of the GridDrones system was the ability to record and play back user-created animations. First, the user summoned a drone-based tangible menu from the GridDrone lattice, wherein the lattice would return to its original flat configuration and orientation and extend tangible record and playback buttons as shown in *Figure 73*. The record button was red and the playback button was green, but if no animations were stored the playback button was illuminated white to indicate the program's memory was empty.

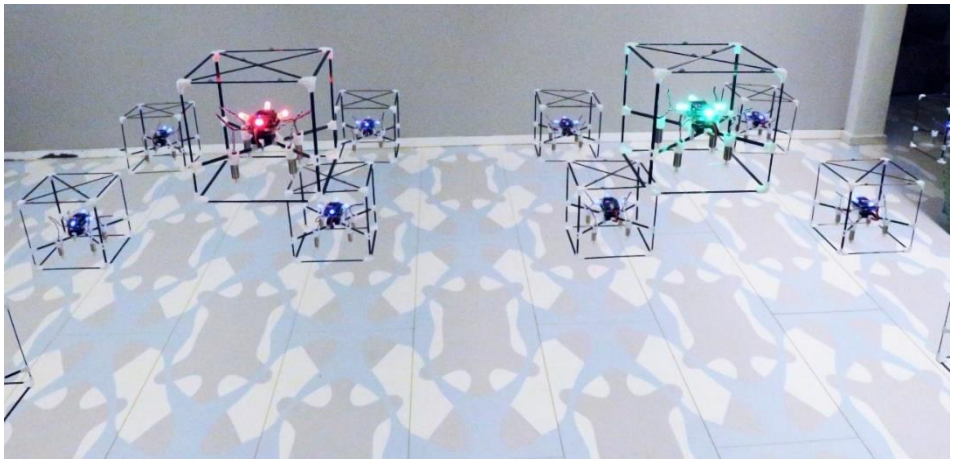


Figure 73: A record button (red) and playback button (green) as part of a tangible menu extended from the GridDrone lattice

To record an animation, the user would manually tap the record button, and the drone-buttons would return to their place in the flat lattice. The user was then free to interact with GridDrones to create animations through any combination of interaction techniques previously described. During the recording phase, desired drone positions were recorded as waypoints within BitDrones OS. When the user was finished, they could confirm the animation from the smartphone app with the press of an on-screen button. Summoning the menu again, the user could then opt to play back the recorded animation. During playback, the user was free to gesturally transform the entire lattice dynamically with the drones in motion. Animations would loop infinitely and their duration was limited only by the BitDrone OS server's available memory. Though the resolution of the display was limited by the interaction volume, it was sufficient to create compound animations from multiple stiffness relationships and dynamic transformations of replayed animations. In one instance, a simple but compelling pair of flapping wings was animated which would lead to further explorations in embodied control of a BitDrone swarm.

5.3 DroneZ

A critical issue with BitDrones as catoms was that they were not able to fly stably over top of each other to occupy the same *xy*-coordinates. This was an obvious problem that hindered the rendering of 3D structures. BitDrone Version 4, or the *XY Drone,* was designed with ducted propellers angled 27 degrees outwards from the drone's yaw axis to redirect and redistribute thrust. The theory was that by reducing the magnitude of the thrust stream directly below the drone, other BitDrones could fly stably underneath. In order to prove this design concept, it was necessary to visualize and compare the thrust beneath standard BitDrones and the *XY Drone*. Measuring air velocities at a precise 3D location required specialized equipment, such as a 3-axis hot-wire anemometer that was capable of high update frequencies and external interfacing for time synchronization between measurements and triangulated 3D coordinates from the VICON. Such equipment exceeded the budget at the time of this exploration, so a new sensing method was devised.

Pitot tubes are differential pressure sensing fluid flow devices that are commonly used to measure the airspeed of aircraft. They would not normally be useful for this application but were deemed an appropriate solution if the pitot tube was kept parallel to the drone's yaw axis (*z*-axis) to measure the main component of thrust. The airflow in the thrust stream was also expected to be highly turbulent which, along with the generally unconventional application of the sensor, would greatly affect the accuracy of velocity measurements. Therefore, precise velocities were not estimated, and only relative air pressures were examined. The logic being that high velocity airflow resulted in a large pressure differential in the sensor giving a high reading, while the opposite was true for low velocity airflow. As long as relative pressures could be examined, that would be adequate to determine if the design concept demonstrated the intended functionality to redirect thrust.

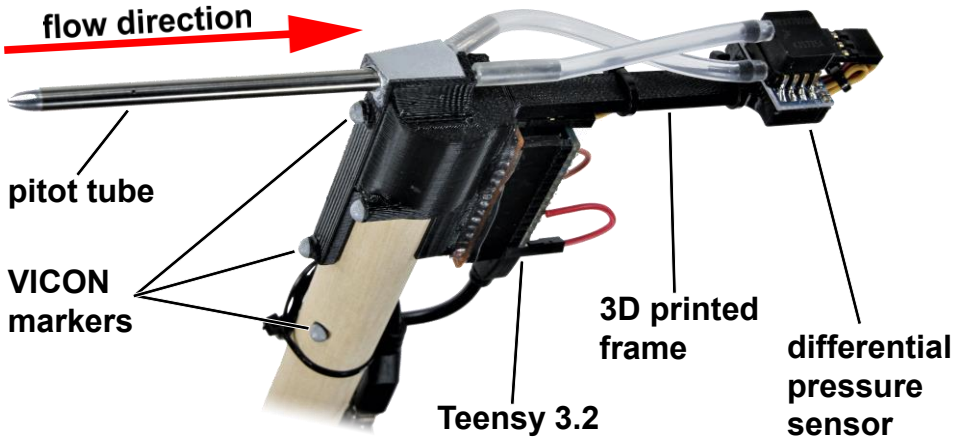


Figure 74: A diagram of the pressure probe showing its main components

Figure 74 above shows the pressure probe designed around a pitot tube, an MPXV7002DP differential pressure sensor and a Teensy 3.2 ARM® development board with a Cortex-M4 MCU running custom firmware. It was interfaced via serial with a laptop running MATLAB R2018b, for which data collection scripts were written. 3D positions of all IR VICON markers on the device were logged with associated pressure measurements taken at those positions. In post-processing, the exact position and orientation of the sensing tip was computed, as well as the 3D configuration of the pitot tube in relation to the drone's 3D configuration. Any data collected while the pitot tube was not within 3 degrees parallel to the drone's yaw axis was discarded. Data offsets, sensor noise and extreme outliers were eliminated, and the data was filtered prior to visualization with MATLAB's built-in implementation of a 1st order Savitzky-Golay filter.

Figure 75 shows measured thrust magnitudes under a BitDrone V5.1 plotted in 3D. High pressures detected by the pitot tube pointing positively along the *z*-axis equated to higher velocity airflow moving along the *z*-axis. These pressures were colour-mapped to arbitrary units in order to serve as a visualization of the drone's thrust. As can be seen, the thrust was concentrated exclusively below the drone and extended downwards.

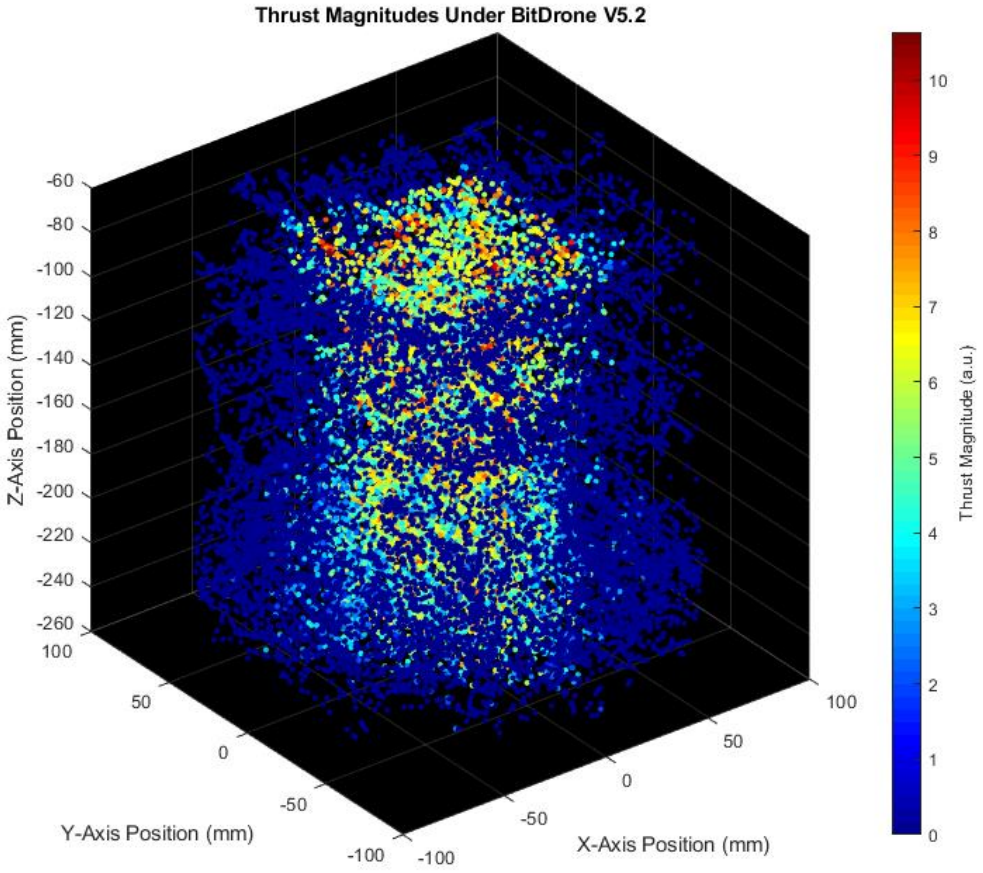


Figure 75: Thrust magnitudes visualized in 3D under a standard BitDrone V5.2

The thrust magnitudes below the *XY Drone* shown in *Figure 76* clearly show that the angled propellers redirected thrust better than the downwards-facing propellers of traditional BitDrones. In order to further highlight this, *Figure 77* and *Figure 78* show 2D slices of the thrust data along the *x*-axis of a traditional BitDrone 5.2 and an *XY Drone* respectively. The magnitudes of the arbitrary thrust units were also generally lower for the *XY Drone*.

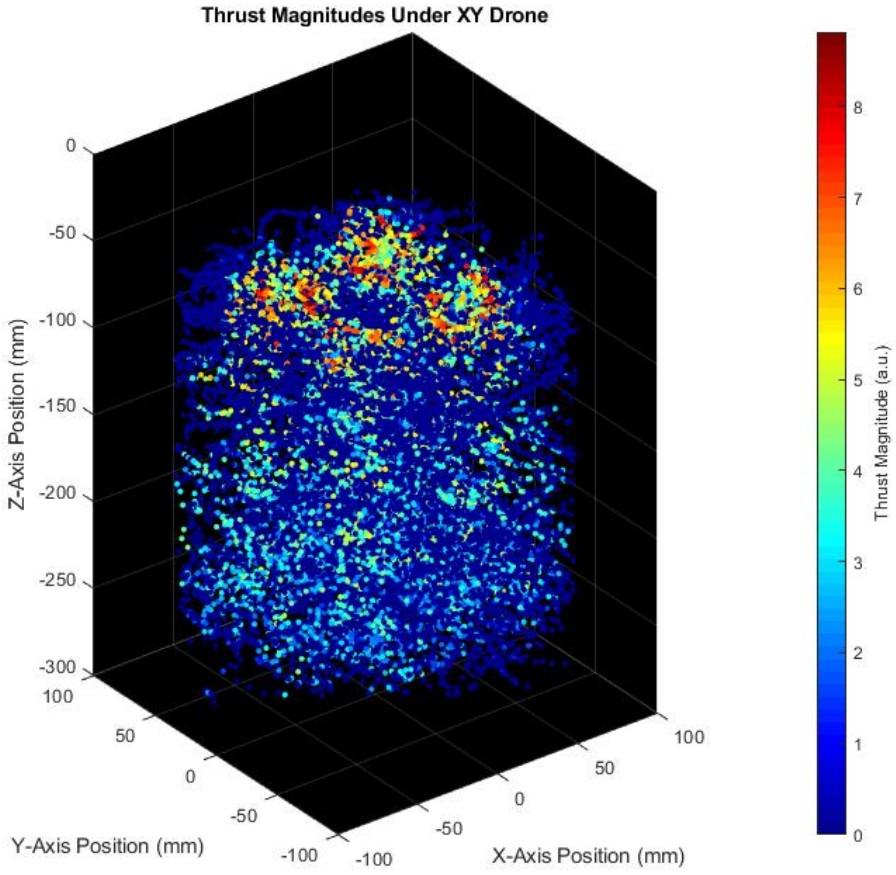


Figure 76: Thrust magnitudes visualized in 3D under a BitDrone V4 (XY Drone)

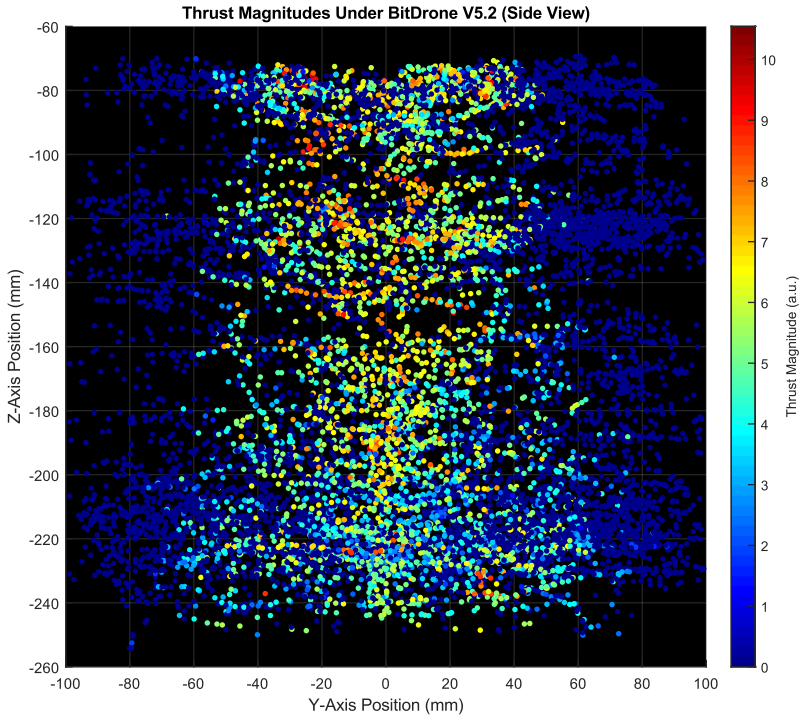


Figure 77: Thrust magnitudes visualized as an x-axis projection under a standard BitDrone V5.2

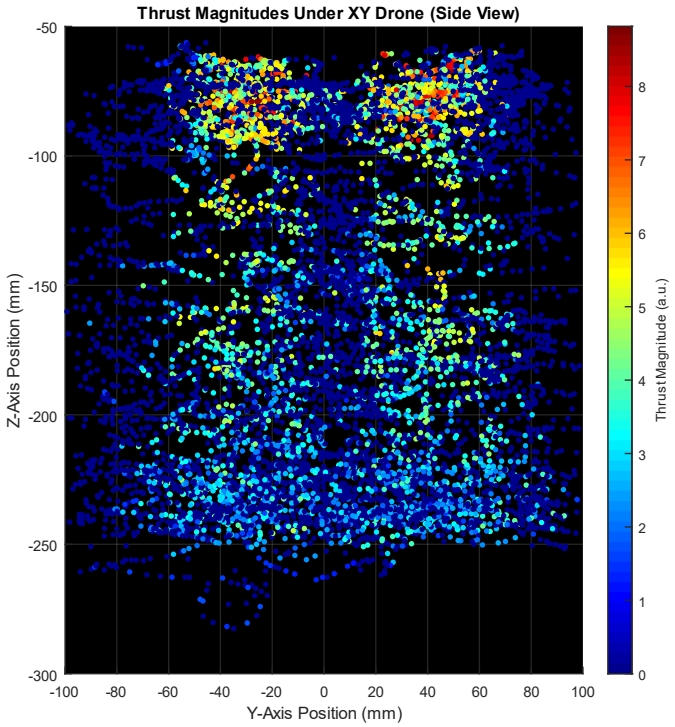


Figure 78: Thrust magnitudes visualized as an x-axis projection under an XY Drone (BitDrone V4)

Relative thrust magnitudes were plotted over the *x*-axis range of a traditional BitDrone in *Figure 79* and confirmed that the majority of the thrust was concentrated directly below the drone. This data showed identical behavior when viewed from along the *y*-axis as the drones were symmetric.

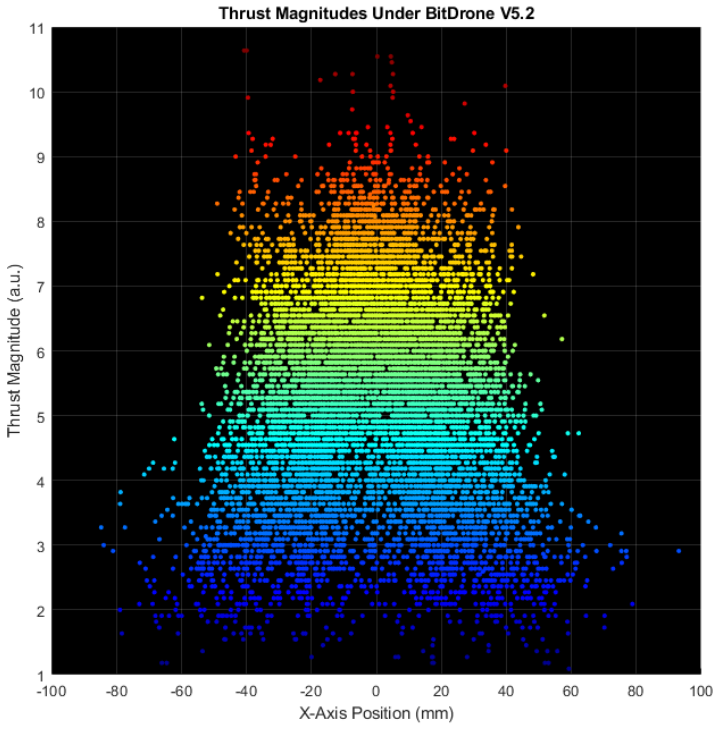


Figure 79: Thrust magnitudes visualized below a standard BitDrone V5.2 along its x-axis

Thrust magnitudes of the *XY Drone* were also plotted along the *x*-axis in *Figure 80*, which clearly showed lower thrust magnitudes in comparison to the standard BitDrone in *Figure 79.* Additionally, the number of thrust readings that had a large enough *z*-component to register via the sensor were significantly lower, resulting in a less dense plot. This indicated successful redirection of the thrust outwards from the drone. Most importantly, there was very little thrust extending downwards from the center of the drone, as indicated by the sparse data about $x = 0$.

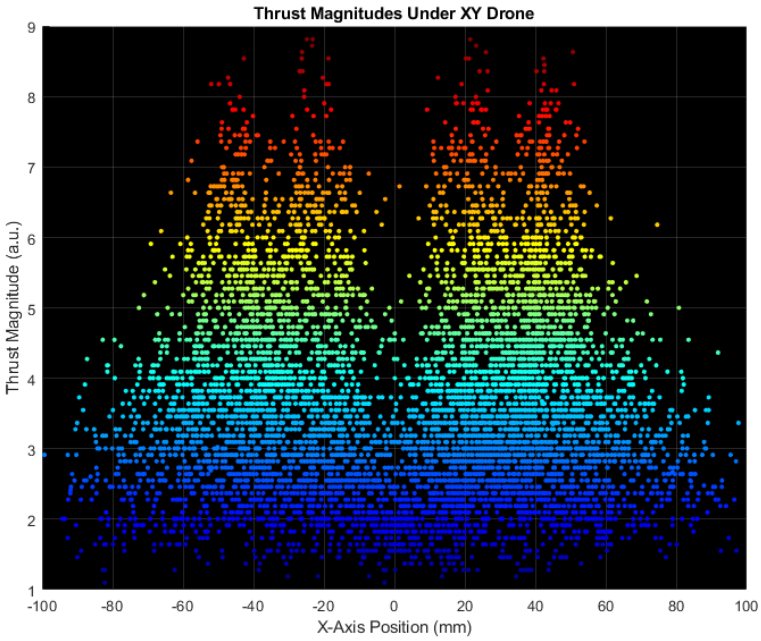


Figure 80: Thrust magnitudes visualized below an XY Drone (BitDrone V4) along its x-axis

The *XY Drone* was tested and it was found that drones were able to hover approximately 50cm directly below it. Path planning would be required to direct and orient drones in order to navigate between thrust streams in dense arrangements, but this proof of concept nonetheless demonstrated a worthwhile design feature to consider for drones used in 3D arrangements or where minimal turbulence directly below an aircraft would be required.

5.4 LEGO Embodied Control

Inspired by the animated pair of flapping wings created with GridDrones in *5.2.1 A Self-Levitating Physical Voxel Lattice*, an entire scenario was created around the live animation of programmable matter. Direct manipulation had already been explored for both tangible modelling and animation which worked well for experienced users, however complex interactions were difficult for the uninitiated. This was not necessarily due to the interaction techniques utilized but was believed to be due to the intimidating nature and technical shortcomings of the system itself (e.g. marker occlusion, swarm size, ominous noise, perception of fragility, etc). This exploration marked the first opportunity to design a highly accessible experience for the average user. More specifically it was tailored to appeal to children through the integration of a construction step that used LEGO bricks to design a controller for the BitDrone swarm. The position and colour of each LEGO brick specified the relative position and colour of a BitDrone in the VICON operational volume. The hardware of the controller is discussed in detail in *4.2.2.4 Embodied Controller*. Safety was of primary concern, so users were physically isolated from the drones behind glass. Accessibility was ultimately increased through distant embodiment of the BitDrone swarm in a tangible interaction object.

LEGO bricks were chosen as the construction medium for the embodied controller as they have a widely recognized appeal as a construction toy. LEGO bricks were also one of the primary inspirations for the initial *ShapeDrone* in *4.2.1.6 Version 2.4*, the design of which carried forward to subsequent cuboid BitDrone versions. In order to create a consistent and reliable experience, the scenario was built upon *GridDrones* from *5.2.1 A Self-Levitating Physical Voxel Lattice* which accommodated for the drones' inability to occupy the same *xy*-coordinates. With *GridDrones*, structures had to be built on a plane before being manipulated in 3D by the user and so the physical build experience with LEGO bricks was limited to flat 2D patterns on modified LEGO base plates.

The controller took the shape of a pair of wings cut out of studded LEGO construction base plates in order to influence the user's construction of the brick pattern**.** By suggesting the shape of the construction with a recognizable pre-existing form, it was believed that this would influence the user in their construction. Producing a more recognizable brick pattern would lead to a more recognizable drone formation, thereby giving the user a greater sense of accomplishment and increasing their satisfaction with the experience. Several different pairs of wings from birds' to bats' were prototyped, but because the wings were being decorated by multi-coloured bricks it was decided that butterfly wings were the most convincing final form. The set of wings was connected with flexible plastic to allow users to flap the wings of the controller.

When a user had finished the construction of their butterfly, the controller was scanned by the *LEGO Creation Scanner*. The scanner's hardware is described in *4.2.2.5 Creation Scanner.* The controller was physically aligned via embedded magnets on the flat scanning surface of the scanner underneath an overhead USB camera. Custom C++ software that utilized the OpenCV 3.4 computer vision library determined the relative locations of bricks to each other, as well as discerned their individual colours. These brick positions and colours were then faithfully mimicked by live BitDrones in the tracked VICON volume, at which point the user could animate their butterfly with the controller.

Because BitDrone structures were relatively large compared to the tracked VICON volume, giving users translational control of the structure would not have made for a compelling experience. It was noted that the Wand in GridDrones provided very responsive control which granted a high level of precision for global spatial rotations, so it was decided that interactions with the embodied butterfly controller should also be based on the ability to precisely manipulate rotations. Since the controller was separated from the drones and therefore the tracked VICON area, another method of obtaining orientations was needed. This was accomplished by using an

embedded 6-DOF IMU in both controller Wings that reported orientation quaternions of each wing via WiFi to BitDrones OS. These orientation quaternions were then mimicked by the BitDrones as in *Figure 81*. Each wing of the BitDrone butterfly could be oriented separately according to its orientation quaternion which allowed users to flap the wings of the butterfly.

Figure 81: A user flying their creation through the embodied controller

This build and play experience was refined for exhibition at the 2018 LEGO World Expo where hundreds of children interacted with the system. Rotation speed and angular limits were imposed on the BitDrone butterfly in order to create a robust experience for even the most exuberant participants.

Due to their relatively short battery life, it was desired for BitDrones to automatically land to charge themselves as well as to automatically assign fully charged BitDrones to new butterfly constructions. The goal of this charging and live replacement strategy was to automate the exhibit and minimize down-time between participants. BitDrone V5.1 was specifically designed for this demonstration with onboard lithium polymer battery charge management circuitry, as well as a simple contact-based power delivery solution. By landing at least two corners of a BitDrone on each of two 12 VDC powered rails introduced in *4.2.1.9 Version 5.1*, copper contacts at the bottom of the drone would provide charging power. Integrated bridge rectifiers

maintained the correct polarity to the drone's systems despite varied physical orientations and connections on the charging rails. These charging rails could be connected together in parallel to create charging space as needed. Due to thermal throttling of the main charge management IC, drones' batteries were not able to be charged in a practical amount of time for use in this scenario. Drones automatically landed when their batteries ran low and were manually replaced for the LEGO exhibit.

Post-exhibition, a passive heat sink was attached to the battery charging IC and its supporting circuitry was modified. A variable resistance allowed the charging circuit to be tuned in order to find the highest thermally sustainable current delivery limit to the battery. With simple passive cooling pictured in *Figure 82*, a maximum thermally sustainable charging current of ~150 mA was attainable. This meant that it would take over two hours for each 350 mAh drone battery to charge. In order to achieve faster battery charging rates it was envisioned that in future drone versions the battery charging circuitry would be relocated to the underside of the drone to actively cool it. When connected to power, the charge management IC offered passthrough functionality in order to power the drone while also isolating and charging the battery separately. Future versions could spin the drones' propellers in reverse to blow air upwards to actively cool the charge management IC when charging the battery to attain higher charge currents.

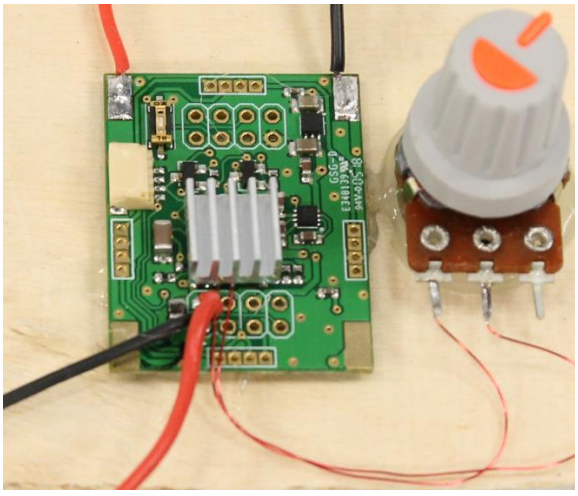


Figure 82: WiFly Junior board modified with a passive heat sink and variable tuning resistor to alter charging current

5.5 DroneForm

Another downfall of BitDrones as catomic elements was their inability to form dense, physically interconnected structures. This was rectified with the creation of BitDrone V5.2 which contained embedded spherical magnets in the eight corners of its cuboid frame, inspired by the design of Kinetic Blocks [7] and DynaBlock [9]. These magnets could freely rotate and self-orient in the presence of other magnetic fields in order to connect to other drones from frame corners and along their edges and faces. A very simple modification to the drones' control system allowed them to fly in connected structures, as well as to build traditionally unstable structural models on flat surfaces.

Drones could be individually oriented while being grabbed in order to line up edges and faces for magnetic midair connections. This user-defined orientation was implemented by disabling the yaw PID while being grabbed and resuming its operation on release. In order to initiate construction from the ground or from existing platforms instead of in midair, drones' motors were disabled if a drone was grabbed with two hands. A disabled drone could then be placed on the ground or a supporting platform as a structural seed to which more drones could be connected. Since drones could not fly when vertically connected to other drones as discussed in *5.3 DroneZ*, they could only be stacked on top of structurally supported drones that were not in flight. To create vertically stackable elements, drones could either be grabbed with two hands to disable their motors or would automatically disable their motors if dragged within a certain radius and height of a disabled drone, at which point they could be manually connected by the user.

The desired altitude of BitDrones in a structure was modified in order to pitch/roll the entire structure, enabling it to hold its 3D position. According to *Figure 83*, the structure's center of mass $\boldsymbol{C_{m1}}$ was found, as well as the structure's desired center of mass $\boldsymbol{C_{m2}}$. The displacement vector between $\boldsymbol{C_{m1}}$ and $\boldsymbol{C_{m2}}$ was calculated as the structure displacement vector $\boldsymbol{S_d}$. A gain K_s

was applied to the magnitude of $\boldsymbol{S_d}$ to obtain the desired angle of attack of the structure in order to move towards $\boldsymbol{C_{m2}}$ as described by *(18)* below.

$$\theta_d = K_s \cdot |\boldsymbol{S_d}| \tag{18}$$

θ_d specified the structure's angle of rotation about a bisecting axis of rotation perpendicular to $\boldsymbol{S_d}$.

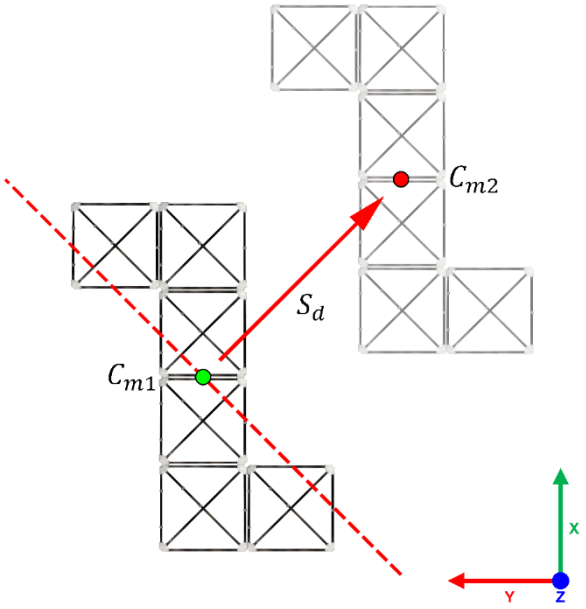


Figure 83: The structure's current position, desired position, displacement vector and a dashed red line indicating the bisecting axis of rotation

As depicted in *Figure 84*, the structure was then virtually translated to the origin and rotated to align $\boldsymbol{S_d}$ with $\hat{\boldsymbol{x}}$ through the application of equations *(4)* through *(7)* in *4.5.2.2 Reference Correction*. The displacements of each rotated BitDrone position from the *y*-axis ($\boldsymbol{B_d}$) was found, and *z*-axis offsets were calculated according to *(19)* which would rotate the structure by θ_d about the vector of bisection. z_{offset} was finally added to individual drones' altitude setpoint.

$$z_{offset} = -sgn(\boldsymbol{B_{d,x}}) \cdot \boldsymbol{B_d} \tan \theta_d \tag{19}$$

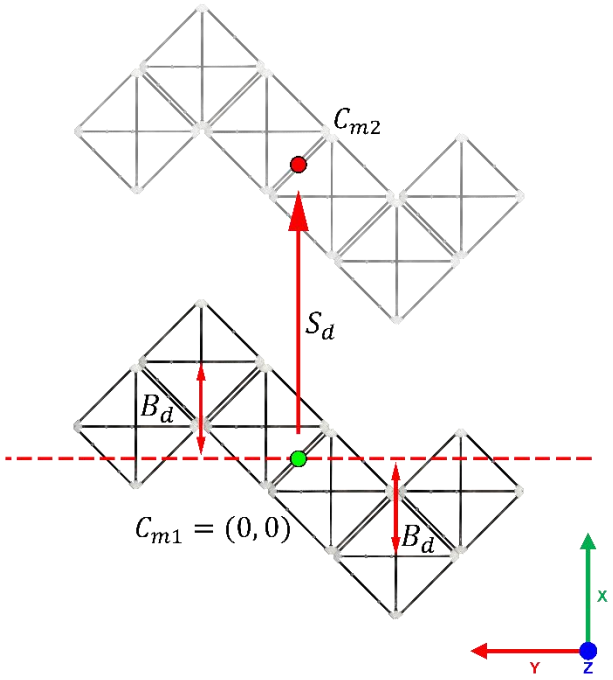


Figure 84: The components of Figure 83 rotated to align $\boldsymbol{S_d}$ with $\hat{\boldsymbol{x}}$

This accommodation to the existing control system allowed dynamic structures of physically connected drones to maintain their position in 3D. Complex configurations could be constructed midair as shown in *Figure 85*.

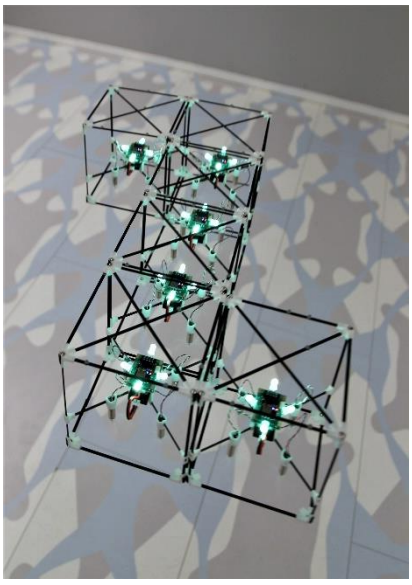


Figure 85: A physically connected BitDrone structure in flight

Structurally unstable ground-based assemblies that defied gravity could also be built as the elements were self-supporting. BitDrones could be joined at their faces to produce aggressively cantilevered rigid architectural features such as in *Figure 86*. *Figure 87* also demonstrates their ability to be connected along edges to create staircase-like features. The colours of individual drones were also mapped to their altitude in order to better discern topology [13].

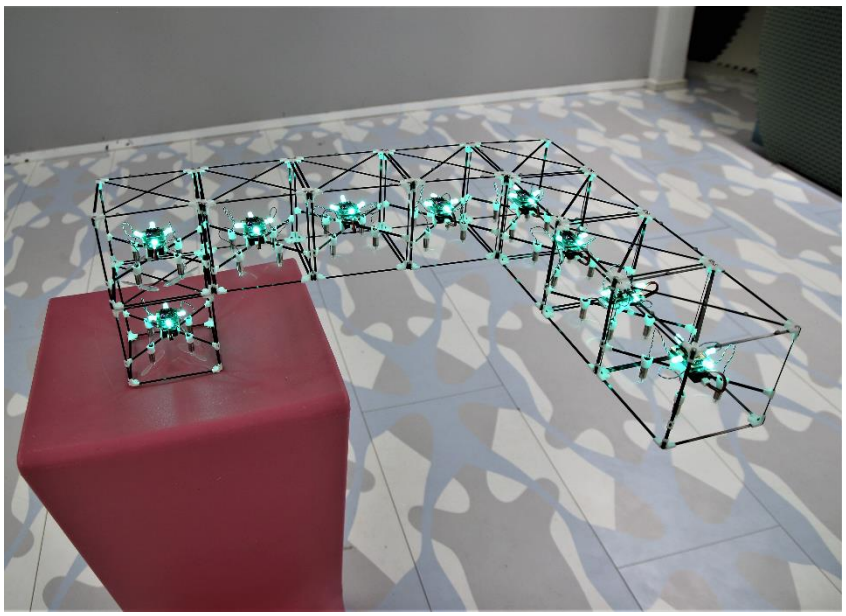


Figure 86: A self-supporting structure extending outwards in midair

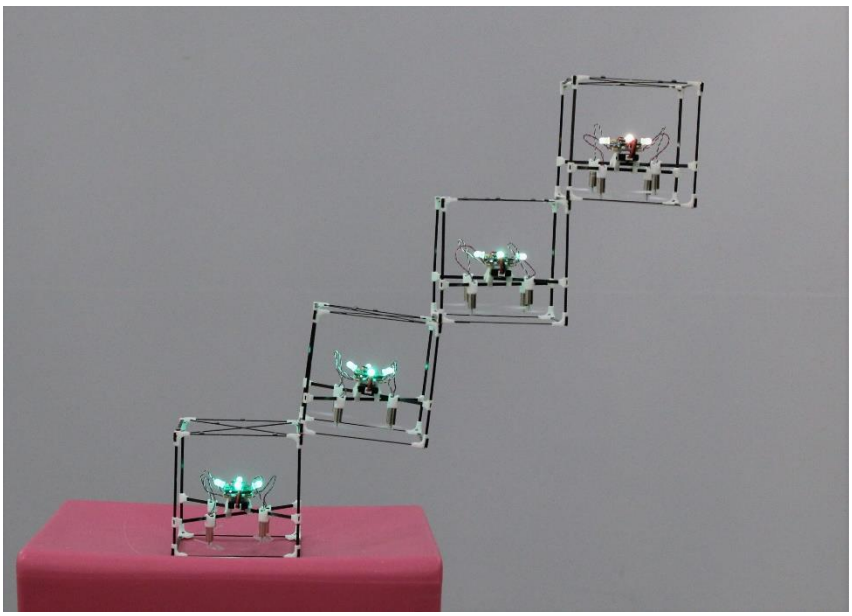


Figure 87: A self-supporting structure connected along BitDrones' edges

Truly dense 3D structures were possible with DroneForm, as can be seen in *Figure 88* with one BitDrone supporting six edge connections with six other drones. Larger constructions were limited by the lack of additional V5.2 BitDrones, as well as functionally by the unsophisticated method of position control.

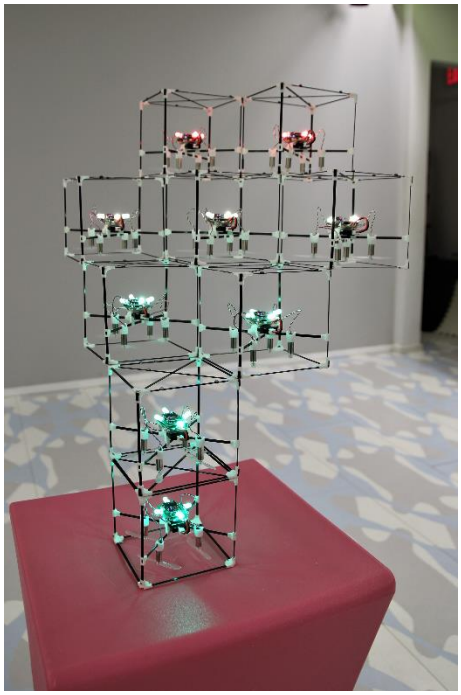


Figure 88: A unique DroneForm structure with both face and edge connections

The control system was far from an ideal solution and did not provide the same level of performance as it did for an individual drone. This particular method for positional control of a structure was implemented due to its simplicity and in the interest of providing a proof-of-concept for catoms of dense programmable matter. Interconnected drone structures should be considered as a single (potentially asymmetric) drone with *n*-engines [43], with a single control system to command its attitude via those *n*-engines. This system would equivalently apply directional commands also through the utilization of all *n*-engines. DroneForm's approach was to control all individual units in a structure instead of reorganizing the control scheme to govern the structure itself.

6. Conclusions and Future Work

6.1 Conclusions

Inspired by early visions of future computing, BitDrones marks the creation of the first tangible drone-based 3D programmable matter interface. It was made possible through the combination of innovative hardware and interactions as well as the application of centralized high-speed low-latency control software. Truly tangible drones were created for immersive physical interaction, new purpose-built drones accommodated for inherent limitations of drone-based displays and personal tools were devised for the manipulation of a programmable matter interface. Accompanying control software and a streamlined network made it possible to develop and demonstrate 3D tangible and gestural interaction techniques in live programmable matter usage scenarios. These scenarios ranged from sequences of simple interaction primitives to the live animation of programmable matter and tangible assembly of flying structures. The technical details of the BitDrones system presented in this thesis provide important general guidance for the design of new tangible drone swarms and drone-based PMIs in the future.

Prior work in several related areas was examined and the design rationale of the system was logically based on this established research. The drones and purpose-built interaction tools that constituted the hardware of the system were introduced which was followed by a breakdown of the system's essential network components, control software and device firmware. Most importantly, the functionality and potential of BitDrones was demonstrated through multiple explorations where each successive exploration incorporated new improvements over the previous one. The initial *BitDrones* system went through two design cycles. They showed the creation and refinement of basic 3D interactions with low-resolution programmable matter structures, improvements in drone and system design, increases in functionality and tangibility, and additionally highlighted early drawbacks and limitations of the system. *GridDrones* was the

third design effort which accommodated for inherent limitations of drones as programmable matter that were accentuated during the creation of *BitDrones*. It introduced much more advanced 3D interaction techniques, more capable drones, a scalable and responsive system as well as physical recording and animation. *DroneZ* was a tangential design exploration and preliminary attempt to design a novel type of multirotor. The resulting *DroneZ* prototypes redirected thrust to allow them to hover above and below each other in close proximity at the expense of efficiency. *LEGO Embodied Control* provided an accessible and playful interaction experience by granting users the ability to remotely construct and control BitDrone models through embodied interaction with a specialized LEGO-based device. Lastly, *DroneForm* showed the possibilities for dense self-levitating programmable matter constructions via magnetic midair connection. This final exploration demonstrated flying interconnected BitDrone structures that were dynamically assembled by a user in flight, as well as the assembly of impossible structurally unstable architecture.

This project began with a memorable lament following a brainstorming session in the summer of 2014: “I want magic levitation stuff”. Obviously this ideal “stuff” was purely conceptual and so research utilizing it only had theoretical potential. This changed when the first BitDrones achieved a stable hover and since then the system has developed into a usable rudimentary programmable matter prototype. Conclusively, BitDrones represents a viable research platform with which to practically explore and perfect new interfaces on the path to *The Ultimate Display*.

6.2 Future Work

As a prototype PMI, significant advancements are necessary to increase the functionality and practicality of BitDrones. The two main areas of development would be the decentralization of the drones' control systems as well as the creation of a drone-based renderer. Decentralization would improve reliability, performance and scalability, while a renderer would automatically construct 3D models from virtual information. There is also a shortlist of more immediate improvements.

A sporadic but critical issue was that of wireless interference. Since the control of all drones was centralized, almost any communication disruption resulted in system failure. Sensor fusion utilizing onboard sensors such as the drones' IMU would allow for continuous operation in the event of brief communication loss. This would mean moving the positional controllers of BitDrones OS to the drones themselves. A preliminary inertial tracking system (INS) was created for the BitDrones and compared with the VICON in *Figure 89*. The 3D samples were taken over 5 seconds and positional estimates became increasingly inaccurate over time, though were quite accurate for approximately 2.5 seconds. This is a good indicator that sensor fusion utilizing IMUs would provide the robustness necessary to accommodate for short communication disruptions.

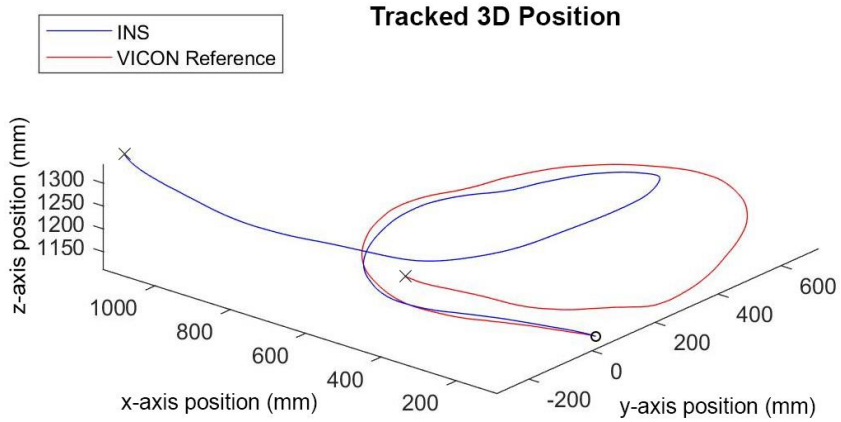


Figure 89: Inertially tracked 3D path compared to a VICON reference path. The circle indicates the start position while the X's represent the end points of each path.

Sensor fusion could also scale the operational volume much more efficiently than existing vision-based motion capture systems such as the VICON. Through the combination of less accurate but less restricting positioning methods such as RF, onboard vision-based and/or inertial based solutions, the VICON could be eliminated and the system's operational area expanded.

BitDrones should also have the ability to dynamically adopt a new control system that would allow them to integrate with conjoined drones for interconnected flight such as in *5.5 DroneForm*. This would necessitate the control of individual motors to contribute to the flight of the whole structure versus stabilizing and orienting individual drones. This decentralized control scheme would allow drones in a structure to regulate the angle and thrust of that entire structure at a much higher frequency than would be possible via decentralized control, thereby resulting in more stable and maneuverable structures.

A drone-based renderer would be the final step for general-purpose use BitDrones. This would give the system the ability to autonomously construct, deconstruct and animate dense, complex 3D models from virtual information. With BitDrones, construction was controlled almost entirely by the user and only rudimentary path planning and obstacle avoidance was employed. To do this more efficiently, true distributed swarm behavior would have to be implemented for parallel self-organization. This is the only way the system could be scaled to larger resolutions. Decreasing the size of individual BitDrones remains an enduring engineering goal, which would additionally enable denser rendering.

In order to give BitDrones the ability to render dense 3D models, a new *XY-Drone* should be developed. Instead of rotating the engines outwards from the yaw axis at a fixed angle, it is conceivable that the next generation drone would be capable of thrust vectoring on a per-engine basis. This would allow for redirection of thrust not only to maneuver the drone itself, but also to redirect thrust to avoid nearby drones. This strategy would require a novel control system to

optimize ideal thrust vectors for maximum flight performance in conjunction with ideal thrust vectors to minimize turbulence for surrounding drones.

In the short-term, several aspects of the drones as well as their associated interactions require improvement. These changes include refining the onboard battery charging and live replacement strategy, physical alterations as well as removing the need for specialized wearable interaction hardware.

The battery charging circuitry requires adequate cooling in order to function at a practical capacity. A lightweight passive radiant cooling solution should be installed on the underside of the drone to accomplish this. The addition of bi-directional motor control would allow the motors to spin in reverse to direct air upwards at the drone. This would create a more effective active cooling solution. The addition of self-orienting magnets for adhesion as in *4.2.1.10 Version 5.2* could allow for the BitDrones to magnetically perch on charging rails positioned on walls or ceilings. The benefit of mounting these charging rails in such a way would be to remove the charging rails and charging drones from the interaction volume so as to not physically obstruct or visually interfere with the functionality of the system while in operation.

BitDrone V3 onwards eliminated the mesh covering that was stretched over their frames in order to decrease maintenance requirements. This mesh served to diffuse light across their surface and highlight the intended volumetric shape of the drone while also providing a flexible, tangible surface. The removal of the mesh was an unfortunate but necessary accommodation, as it was easily damaged in crashes by the spinning propellers when the drones' frames deformed. While maintaining large numbers of drones for testing, it was time consuming to repair and refit mesh diffusers. Since this was a research endeavor, it was decided to exclude them from future designs to make them more serviceable and to increase the brightness and number of LEDs instead. Even so, the appearance of each BitDrone as a voxel could be

improved by the addition of more RGB LEDs strategically placed on the frame for maximal illumination of the drones' structure.

A diffusing layer should be reintroduced to grant the drones the appearance of a solid 3D shape, with special focus on maximizing airflow through the material. An issue faced during the design of the *ShapeDrone* was that many diffusers were not air permeable enough which "choked" the propellers by limiting the pressure immediately above them during operation. This reduced thrust, as well as significantly decreased motor lifespan due to higher motor RPM. The issue was even more apparent in the design of drones with impermeable surrounding structures which was the case in *LightBee* [53]: a spin-off project of BitDrones that utilized a drone enclosed in a hollow cylindrical projection screen for telepresence applications. This screen severely impacted the efficiency of the drone's engines and necessitated the use of powerful motors and aggressively pitched propellers.

Lastly, specialized wearable interaction hardware should be eliminated. Wearable devices allowed easy and accurate tracking within the VICON volume but were cumbersome and crude. In future BitDrone iterations, vision-based methods should be explored that do not require user augmentation to recognize gestures and intentions. A drone-centric approach to this would be to include onboard capacitive-based touch detection.

References

[1] A. Gomes, C. Rubens, S. Braley and R. Vertegaal, "Bitdrones: towards using 3d nanocopter displays as interactive self-levitating programmable matter," in *Proc. 2016 SIGCHI Conference on Human Factors in Computing Systems (CHI '16)*, San Jose, CA, USA, 2016.

[2] I. E. Sutherland, "The ultimate display," *Proceedings of the International Federation for Information Processing Congress 65,* vol. 2, pp. 506-508, 1965.

[3] T. Toffoli and N. Margolus, "Programmable matter: concepts and realization," *Physical: Nonlinear Phenomena,* vol. 47, no. 1, pp. 263-272, 1991.

[4] S. C. Goldstein and T. C. Mowry, "Claytronics: a scalable basis for future robots," *RoboSphere,* 2004.

[5] S. C. Goldstein, J. D. Campbell and T. C. Mowry, "Programmable matter," *IEEE Computer,* vol. 38, no. 6, pp. 99-101, 2005.

[6] H. Ishii, D. Lakatos, L. Bonanni and J.-B. Labrune, "Radical atoms: beyond tangible bits, toward transformable materials," *Interactions,* vol. 19, no. 1, pp. 38-51, 2012.

[7] P. Schoessler, D. Windham, D. Leithinger, S. Follmer and H. Ishii, "Kinetic blocks: actuated constructive assembly for interaction and display," in *Proc. 28th Annual ACM Symposium on User Interface Software & Technology (UIST '15)*, Charlotte, NC, USA, 2015.

[8] J. W. Romanishin, K. Gilpin, S. Claici and D. Rus, "3d M-Blocks: self-reconfiguring robots capable of locomotion via pivoting in three dimensions," in *IEEE International Conference on Robotics and Automation (ICRA '15)*, Seattle, Washington, USA, 2015.

[9] R. Suzuki, J. Yamaoka, D. Leithinger, T. Yeh, M. D. Gross, Y. Kawahara and Y. Kakehi, "Dynablock: dynamic 3d printing for instant and reconstructable shape formation," in *Proc. 31st Annual ACM Symposium on User Interface Software and Technology (UIST '18)*, Berlin, Germany, 2018.

[10] G. W. Fitzmaurice, H. Ishii and W. Buxton, "Bricks: laying the foundations for graspable user interfaces," in *Proc. 1995 SIGCHI Conference on Human Factors in Computing Systems (CHI '95)*, Denver, Colorado, USA, 1995.

[11] H. Ishii and B. Ullmer, "Tangible bits: towards seamless interfaces between people, bits and atoms," in *Proc. 1997 SIGCHI Conference on Human Factors in Computing Systems (CHI '97)*, Atlanta, Georgia, USA, 1997.

[12] J. Underkoffler and H. Ishii, "Urp: a luminous-tangible workbench for urban planning and design," in *Proc. 1999 SIGCHI Conference on Human Factors in Computing Systems (CHI '99)*, Pittsburgh, Pennsylvania, USA, 1999.

[13] B. Piper, C. Ratti and H. Ishii, "Illuminating clay: a 3-D tangible interface for landscape analysis," in *Proc. 2002 SIGCHI Conference on Human Factors in Computing Systems (CHI '02)*, Minneapolis, Minnesota, USA, 2002.

[14] S. Jordà, G. Geiger, M. Alonso and M. Kaltenbrunner, "The reacTable: exploring the synergy between live music performance and tabletop tangible interfaces," in *Proc. 1st International Conference on Tangible, Embedded and Embodied Interactions (TEI '07)*, Baton Rouge, LA, USA, 2007.

[15] J. Patten and H. Ishii, "Mechanical constraints as computational constraints in tabletop tangible interfaces," in *Proc. 2007 SIGCHI Conference on Human Factors in Computing Systems (CHI '07)*, San Jose, California, USA, 2007.

[16] H. Benko, R. Jota and A. Wilson, "Miragetable: freehand interaction on a projected augmented reality tabletop," in *Proc. 2012 SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*, Austin, Texas, USA, 2012.

[17] R. V. David Holman, "Organic user interfaces: designing computers in any way, shape, or form," *Communications of the ACM,* vol. 51, no. 6, pp. 48-55, 2008.

[18] H. Iwata, H. Yano, F. Nakaizumi and R. Kawamura, "Project FEELEX: adding haptic surface to graphics," in *Proc. 28th Annual Conference on Computer Graphics and Interactive Techniques (CG '01)*, Los Angeles, California, USA, 2001.

[19] H. S. Raffle, A. J. Parkes and H. Ishii, "Topobo: a constructive assembly system with kinetic memory," in *Proc. 2004 SIGCHI Conference on Human Factors in Computing Systems (CHI '04)*, Vienna, Austria, 2004.

[20] D. Leithinger and H. Ishii, "Relief: a scalable actuated shape display," in *Proc. 4th International Conference on Tangible, Embedded, and Embodied Interaction (TEI '10)*, Cambridge, Massachusetts, USA, 2010.

[21] M. Blackshaw, A. DeVincenzi, D. Lakatos, D. Leithinger and H. Ishii, "Recompose: direct and gestural interaction with an actuated surface," in *CHI '11 Extended Abstracts on Human Factors in Computing Systems (CHI EA '11)*, Vancouver, BC, Canada, 2011.

[22] S. Follmer, D. Leithinger, A. Olwal, A. Hogge and H. Ishii, "inFORM: dynamic physical affordances and constraints through shape and object actuation," in *Proc. 26th Annual ACM Symposium on User Interface Software and Technology (UIST '13)*, St. Andrews, Scotland, United Kingdom, 2013.

[23] A. F. Siu, E. J. Gonzalez, S. Yuan, J. B. Ginsberg and S. Follmer, "shapeShift: 2d spatial manipulation and self-actuation of tabletop shape displays for tangible and haptic interaction," in *Proc. 2018 SIGCHI Conference on Human Factors in Computing Systems (CHI '18)*, Montreal, QC, Canada, 2018.

[24] D. R. Sahoo, T. Neate, Y. Tokuda, J. Pearson, S. Robinson, S. Subramanian and M. Jones, "Tangible drops: a visio-tactile display using actuated liquid-metal droplets," in *Proc. 2018 SIGCHI Conference on Human Factors in Computing Systems (CHI '18)*, Montreal QC, Canada, 2018.

[25] M. Karagozler, B. Kirby, W. J. Lee, E. Marinelli and T. C. Ng, "Ultralight modular robotic building blocks for the rapid deployment of planetary outposts," in *Revolutionary Aerospace Systems Concepts Academic Linkage (RASC-AL) Forum*, Cape Canaveral, FL, USA, 2006.

[26] J. Lee, R. Post and H. Ishii, "ZeroN: mid-air tangible interaction enabled by computer controlled magnetic levitation," in *Proc. 24th Annual ACM Symposium on User Interface Software and Technology (UIST '11)*, Santa Barbara, California, USA, 2011.

[27] Y. Ochiai, T. Hoshi and J. Rekimoto, "Pixie dust: graphics generated by levitated and animated objects in," *ACM Transactions on Graphics (TOG),* vol. 33, no. 4, pp. 85:1-85:13, 2014.

[28] J. Alonso-Mora, A. Breitenmoser, M. Rufli, R. Siegwart and P. Beardsley, "Multi-robot system for artistic pattern formation," in *IEEE International Conference on Robotics and Automation (ICRA '11)*, Shanghai, China, 2011.

[29] J. Alonso-Mora, A. Breitenmoser, M. Rufli, R. Siegwart and P. Beardsley, "Image and animation display with multiple mobile robots," *The International Journal of Robotics Research,* vol. 31, no. 6, pp. 753-773, 2012.

[30] M. Rubenstein, C. Ahler and R. Nagpal, "Kilobot: a low cost scalable robot system for collective behaviors," in *IEEE International Conference on Robotics and Automation (ICRA '12)*, Saint Paul, Minnesota, USA, 2012.

[31] M. Rubenstein, A. Cornejo and R. Nagpal, "Programmable self-assembly in a thousand-robot swarm," *Science,* vol. 345, no. 6198, pp. 795-799, 2014.

[32] M. Le Goc, L. H. Kim, A. Parsaei, J.-D. Fekete, P. Dragicevic and S. Follmer, "Zooids: building blocks for swarm user interfaces," in *Proc. 29th Annual Symposium on User Interface Software and Technology (UIST '16)*, Tokyo, Japan, 2016.

[33] R. Suzuki, J. Kato, M. D. Gross and T. Yeh, "Reactile: programming swarm user interfaces through direct physical manipulation," in *Proc. 2018 SIGCHI Conference on Human Factors in Computing Systems (CHI '18)*, Montreal, QC, Canada, 2018.

[34] M. S. C. Lab, "Flyfire," Massachusetts Institute of Technology, Feb 2010. [Online]. Available: http://senseable.mit.edu/flyfire/. [Accessed 12 December 2018].

[35] ARS Electronica, "Drone 100 - futurelab," Intel, 2016. [Online]. Available: https://ars.electronica.art/futurelab/en/project/drone-100/. [Accessed 12 December 2018].

[36] R. Swatman, "Intel launches 500 drones into sky and breaks world record in spectacular style," Guinness World Records, 4 November 2016. [Online]. Available:

http://www.guinnessworldrecords.com/news/2016/11/intel-launches-500-drones-into-sky-and-breaks-world-record-in-spectacular-style-449886. [Accessed 12 December 2018].

[37] "Intel breaks guinness world records title for drone light shows in celebration of 50th anniversary," Intel, 18 July 2018. [Online]. Available: https://newsroom.intel.com/news/intel-breaks-guinness-world-records-title-drone-light-shows-celebration-50th-anniversary/#gs.eyf24k. [Accessed 14 December 2018].

[38] M. Burns, "Intel powered the drones during lady gaga's super bowl halftime show," Tech Crunch, 5 February 2017. [Online]. Available: https://techcrunch.com/2017/02/05/intel-powered-the-drones-during-lady-gagas-super-bowl-halftime-show/. [Accessed 12 December 2018].

[39] "Intel drone light show breaks guinness world records title at olympic winter games pyeongchang 2018," Intel, 9 February 2018. [Online]. Available: https://newsroom.intel.com/news-releases/intel-drone-light-show-breaks-guinness-world-records-title-olympic-winter-games-pyeongchang-2018/#gs.ev8blv. [Accessed 13 December 2018].

[40] A. Nanduri, "Beyond fireworks: the next generation of drone light shows," Intel, 16 April 2018. [Online]. Available: https://newsroom.intel.com/editorials/beyond-fireworks-next-generation-drone-light-shows/#gs.ev6sw7. [Accessed 13 December 2018].

[41] J. A. Preiss, W. Honig, G. S. Sukhatme and N. Ayanian, "Crazyswarm: a large nano-quadcopter swarm," in *IEEE International Conference on Robotics and Automation (ICRA '17)*, Singapore, 2017.

[42] "Verity studios," [Online]. Available: https://veritystudios.com/. [Accessed 14 December 2018].

[43] D. Saldana, B. Gabrich, G. Li, M. Yim and V. Kumar, "Modquad: the flying modular structure that self-assembles in midair," in *IEEE International Conference on Robotics and Automation (ICRA '18)*, Brisbane, QLD, Australia, 2018.

[44] S. Schneegass, F. Alt, J. Scheible and A. Schmidt, "Midair displays: concept and first experiences with free-floating pervasive displays," in *Proceedings of The International Symposium on Pervasive Displays (PerDis '14)*, Copenhagen, Denmark, 2014.

[45] W. Yamada, K. Yamada, H. Manabe and D. Ikeda, "iSphere: self-luminous spherical drone display," in *Proc. 30th Annual ACM Symposium on User Interface Software and Technology (UIST '17)*, Québec City, QC, Canada, 2017.

[46] J. R. Cauchard, J. L. E, K. Y. Zhai and J. A. Landay, "Drone & me: an exploration into natural human-drone interaction," in *Proc. 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UBICOMP '15)*, Osaka, Japan, 2015.

[47] P. Knierim, T. Kosch, V. Schwind, M. Funk, F. Kiss, S. Schneegass and N. Henze, "Tactile drones - providing immersive tactile feedback in virtual reality through quadcopters," in *Proc. 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems (CHI EA '17)*, Denver, Colorado, USA, 2017.

[48] K. Nitta, K. Higuchi and J. Rekimoto, "Hoverball: augmented sports with a flying ball," in *Proc. 5th Augmented Human International Conference*, Kobe, Japan, 2014.

[49] F. Augugliaro and R. D'Andrea, "Admittance control for physical human-quadrocopter interaction," in *European Control Conference*, Zürich, Switzerland, 2013.

[50] J. Alonso-Mora, S. H. Lohaus, P. Leemann, R. Siegwart and P. Beardsley, "Gesture based human - multi-robot swarm interaction and its application to an interactive display," in *IEEE International Conference on Robotics and Automation (ICRA '15)*, Seattle, Washington, USA, 2015.

[51] C. Rubens, S. Braley, A. Gomes, D. Goc, X. Zhang, J. P. Carrascal and R. Vertegaal, "Bitdrones: towards levitating programmable matter using interactive 3d quadcopter displays," in *Adjunct Proc. of the 28th Annual ACM Symposium on User Interface Software & Technology (UIST '15)*, Daegu, Kyungpook, Republic of Korea, 2015.

[52] S. Braley, C. Rubens, T. Merritt and R. Vertegaal, "Griddrones: a self-levitating physical voxel lattice for interactive 3d surface deformations," in *Proc. 31st Annual ACM Symposium on User Interface Software and Technology (UIST '18)*, Berlin, Germany, 2018.

[53] X. Zhang, S. Braley, C. Rubens, T. Merritt and R. Vertegaal, "LightBee: a self-levitating light field display for hologrammatic telepresence," in *Proc. 2019 SIGCHI Conference on Human Factors in Computing Systems (CHI '19)*, Glasgow, Scotland, Uk, 2019.

[54] I. Poupyrev, T. Nashida, S. Maruyama, J. Rekimoto and Y. Yamaji, "Lumen: interactive visual and shape display for calm computing," in *Proc. ACM SIGGRAPH 2004 Emerging Technologies (SIGGRAPH '04)*, Los Angeles, California, 2004.