Implementation of the Task Delegation Interface for the Automation Supporting Prolonged Independent Residence for the Elderly (ASPIRE) program

Sebastian S. Rodriguez, Wyatt McAllister, Ambika Dubey, Amber Zhang

Department of Computer Science

University of Illinois at Urbana-Champaign

Abstract

The Automation Supporting Prolonged Independent Residence for the Elderly (ASPIRE) program, a multidisciplinary approach in using drones to enhance aging in place, has shown promising results into the future of coexisting with autonomous drones, particularly in the fields of psychology and robotics. In order to connect computer devices to the drones, a base command-recognition interface is described and developed, dubbed the Task Delegation Interface (TDI). The Task Delegation Interface works by taking advantage of modularity; if the user requests a drone to perform a particular action, the action is decomposed into smaller actions, and executed independently in order to track completion of the action, and in case of failure, know the location, time and reason of failure, and apply error handling routines as necessary. Finally, the Task Delegation Interface serves as an abstraction of low level drone control and flight routines, allowing a simple network connection to serve as a gateway for commands. Any device capable of connecting to a network and sending/receiving data over the TCP protocol is able to control a set of drones in a household, providing a more practical environment for future testing.

Introduction

The Automation Supporting Prolonged Independent Residence for the Elderly (ASPIRE) program aims to develop a framework for developers that will enable an assistive environment for the elderly using autonomous drones. The research group has made notable advances in the fields of psychology (a VR testbed for drones, a VR approach in measuring arousal when drones are nearby), and robotics (implanting emotion into drone movement, real-time collision avoidance, pathfinding with moving targets. The method to join these architectures together was theorized through a hierarchical task manager, where actions taken by the drones can be decomposed into subtasks. In this report, we implement said interface, dubbed the Task Delegation Interface.

The Task Delegation Interface serves as an abstraction of the low level routines executed inside the drones' architecture (movement, pathfinding, synchronicity); a device need only to connect to the network hosting the drones, and send structured packets through a pre-defined port. The Task Delegation Interface will decompose, validate, and ensure execution of each action requested in the safest way possible, taking proactive measures to ensure the user's safety. This document specifies the Task Delegation Interface's development and logic, with the aim that future work will expand it.

Task Delegation Interface Implementation

The Task Delegation Interface (TDI) was developed with one principle in mind: Allow developers the freedom to design proprietary applications that control a set of drones in a household, as is the goal of the ASPIRE program. Figure 1 details the control flow from user input from a Human Interface Device, to movement output to the drones.

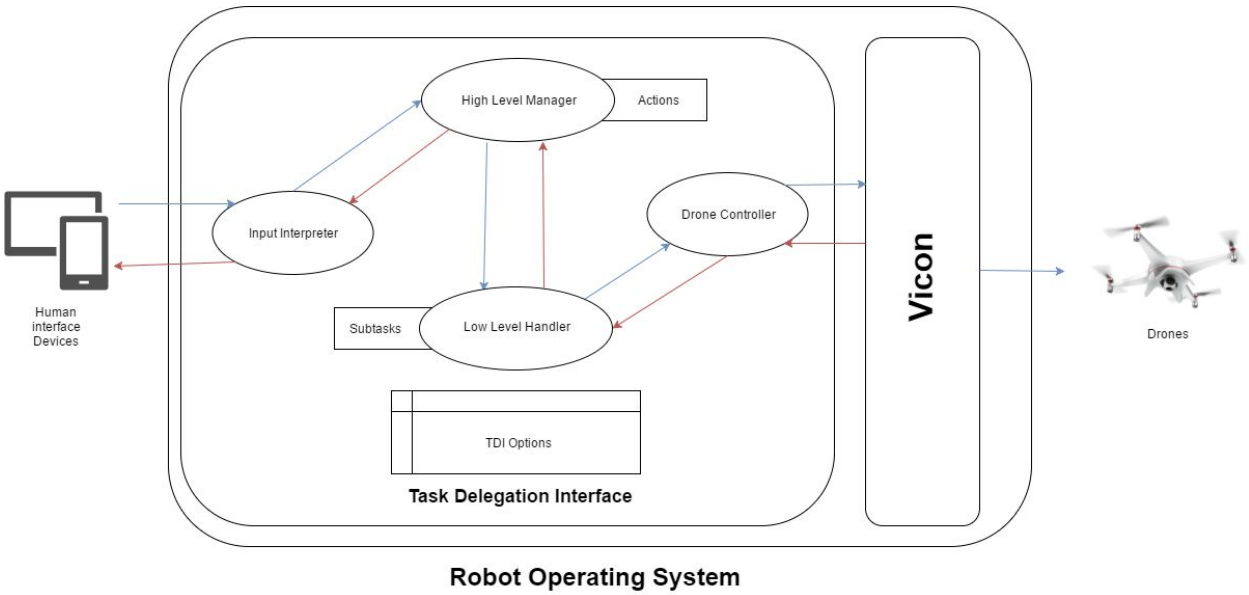

Figure 1. Task Delegation Interface control flow diagram

Input Interpreter (I2)

The Input Interpreter (pronounced *eye-two*) serves as the main application gateway for Human Interface Devices to send/receive information about the TDI. A Human Interface Device (HID) is any piece of technology capable of connecting to a network and sending/receiving information through the Transmission Control Protocol (TCP). An HID connected to the TDI will be referred to as a "client" from herein.

To establish a connection between a client and the TDI, TCP's 3-way handshake (SYN-SYN-ACK) will be utilized (Figure 2). The client requests a connection by sending a packet containing a synchronization request. The TDI will respond by acknowledging the request, accepting it. The client then responds with acknowledgement of the acceptance. After the TDI receives the client's final acknowledge, the TDI will begin accepting request from that client; thus, the final acknowledge is non-negotiable.

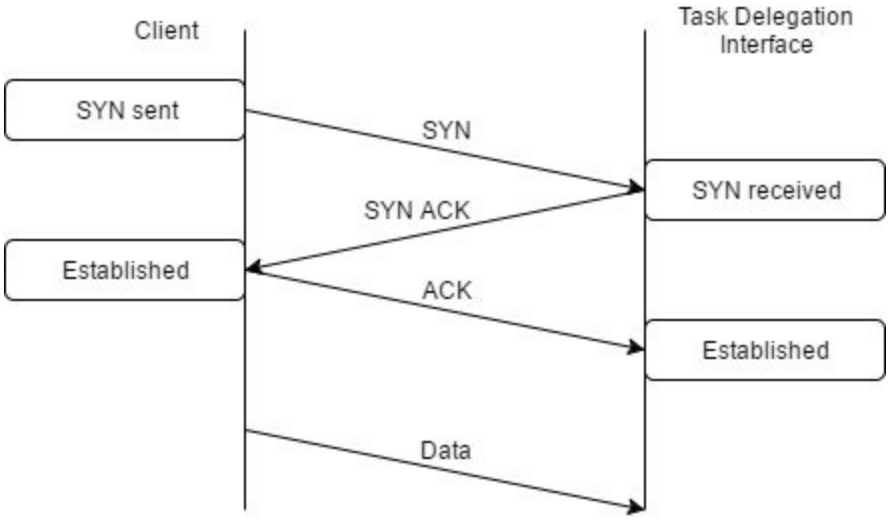

Figure 2. TDI 3-way handshake connection method

The I2 will then accept any incoming data from the connected client and attempt to execute any request sent. Clients will be required to send action requests to the TDI following a specific format, else it is to be rejected. Requests must contain an action's string identifier (e.g. "return_home"), the number of arguments (this number should be the sum of all parameters of each subtask), and arguments. After the I2 determines the request is valid, it sends it to the High Level Manager. Additionally, it sends the client an acknowledgment, which can be used drive the client's interface.

High Level Manager (HLM)

The High Level Manager deconstructs the request sent by the I2, looks up the execution timeline pertaining to the action requested, and constructs the argument list used by the Low Level Handler. An execution timeline is a list of subtask identifiers, signifying the order of subtasks to be executed to resemble the action. Allowing modularity, we can create any action by combining subtasks, as long as we have enough arguments to cover each subtask's parameters. An argument list is constructed from the original request to be sent to the Low Level Handler, in order of parameters with respect to each subtask (Figure 3).

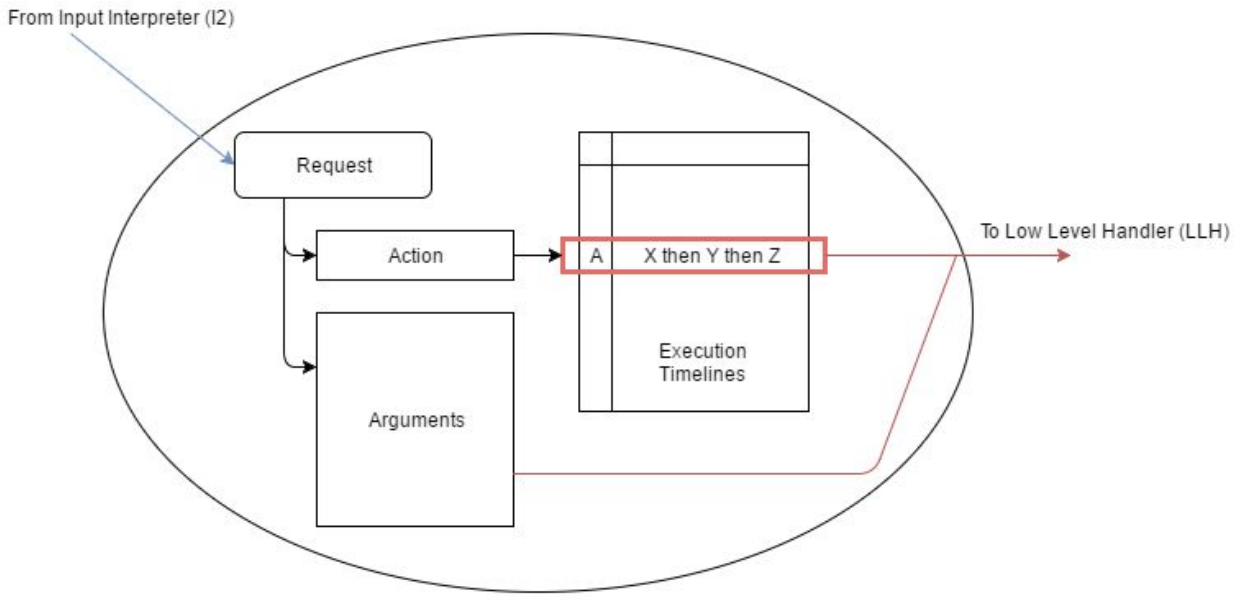

Figure 3. High Level Manager node control flow diagram

Low Level Handler (LLH)

The Low Level Handler receives an execution timeline and an ordered list of arguments, and executes each subtask with a set of arguments. For each subtask, the LLH retrieves the amount of

arguments needed from the list, and sends an individual request to the Drone Controller. The Drone Controller responds if that individual subtask was executed successfully. If it was, the LLH sends the following request to the Drone Controller with the next subtask (Figure 4). If at any point a subtask fails, the LLH knows what part of the action was unable to be completed, and sends it back to the HLM for error handling.

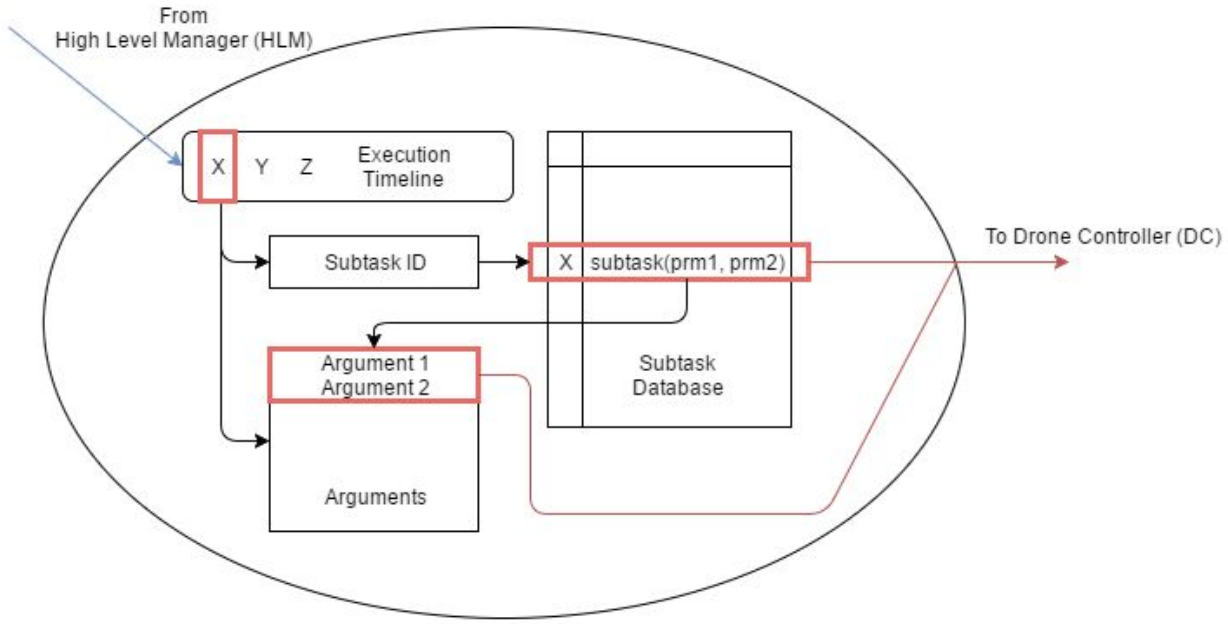

Figure 4. Low Level Handler node control flow diagram

Drone Controller (DC)

The Drone Controller resolves the subtask identifier, ensures integrity of the arguments, and interfaces with the bridge between the TDI and the Vicon system. The subtask is sent to the drone asynchronously; it returns success when the bridge is able to initiate the subtask in the drones. Because the nodes run inside ROS, we are able to access drone position in a three-dimensional space, potentially allowing the DC to track subtask progress and return success. This will be implemented in the future.

Task Delegation Interface Settings

Constants and options are defined in a separate file accessible by all TDI nodes. Constants such as hostnames and ports for connectivity, positions for the drone stations and interactables, and developer debug options are stored here. Additionally, an interface can be developed to edit these constants as users set up their own drone systems.

Test Client Implementation

Upon completion of the TDI, an Android application was built to serve as the first client to verify the full control flow from a Human Interface Device to movement of a drone in a secure space. Using Android Studio, Google's integrated development environment (IDE) powered by IntelliJ IDEA to develop Android applications, we tested the integrity of commands and connection reliability.

Upon launch of the application, the user is greeted with a connection page to input an IP address and port, with the other side being a valid TDI server listening to a port (Figure 5). This page is only used for debugging purposes, as connecting to the TDI should be seamless and invisible to the user. After connecting, the root menu launches, with various actions in a grid layout (Figure 6). For purposes of this implementation, only "Return Home" and "Move To Position (Debug)" are fully functioning. Upon execution of an action, the interface presents a "Cancel" button (Figure 7).

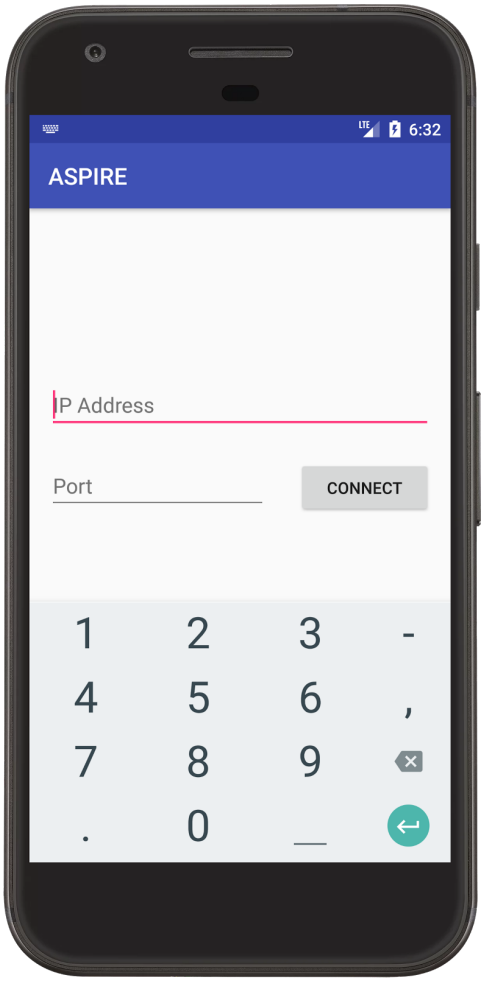

Figure 5. Connection activity

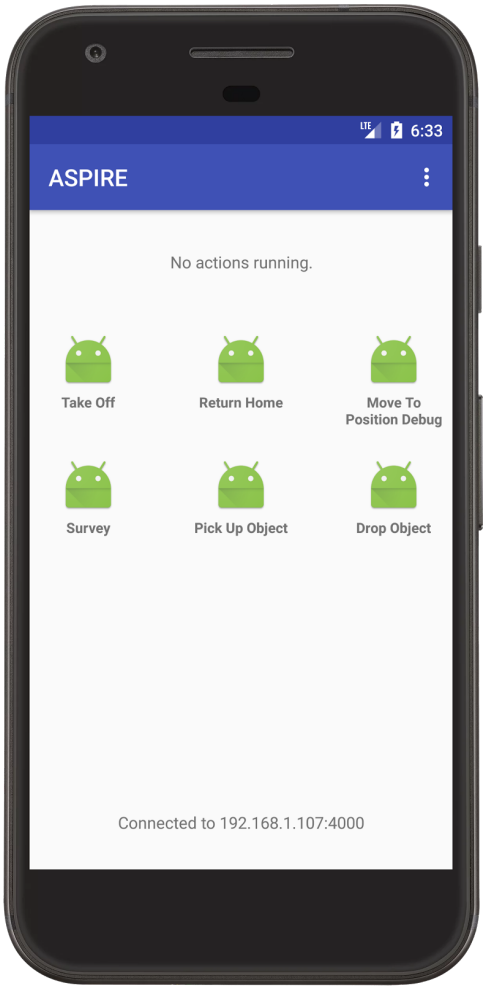

Figure 6. Root Menu activity

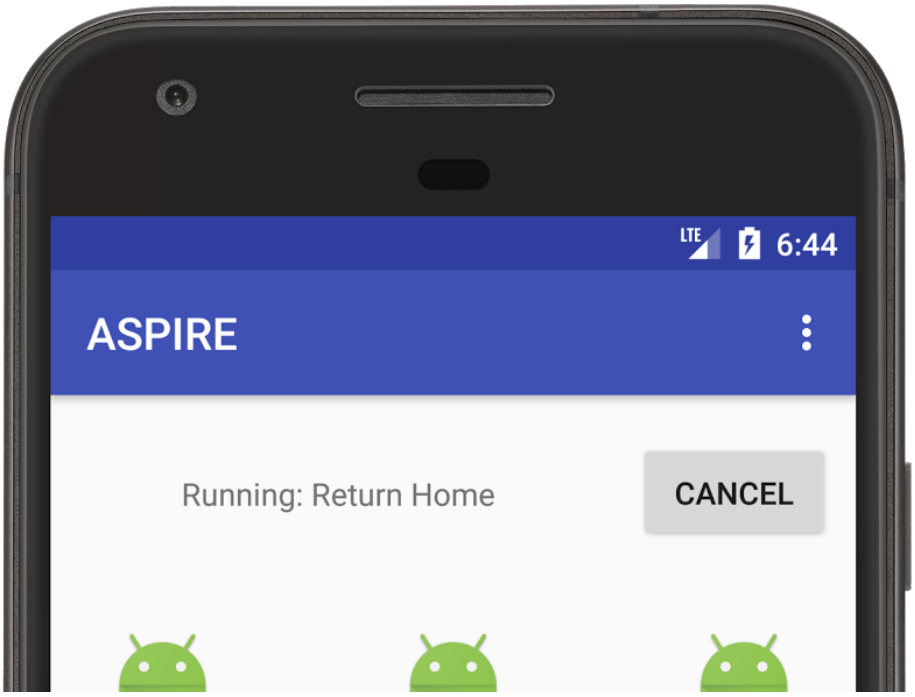

Figure 7. Task update in UI and Cancel option

"Return Home" is a zero-argument action that returns the active drone to a predefined position set in the Task Delegation Interface Settings. Home can be defined as the default position for a drone or the location of a drone station. "Move To Position (Debug)" is a debugging tool that directly moves the drone to a specified position in a three-dimensional space, using Vicon infrared sensors to track position of the drone.

For an initial test run, we began by launching the drone to a low position near the ground. We raised its altitude by passing a Move To Position (Debug) command with the Z argument equaling 0.50 meters (Figure 8). We continued raising the altitude to 1 meter with the appropriate Z argument (Figure 9). Finally, we moved the drone through the Y-axis by passing the appropriate parameter (Figure 10). Thanks to the drone's self-stabilization module, it was able to fly around the space with no apparent issues.

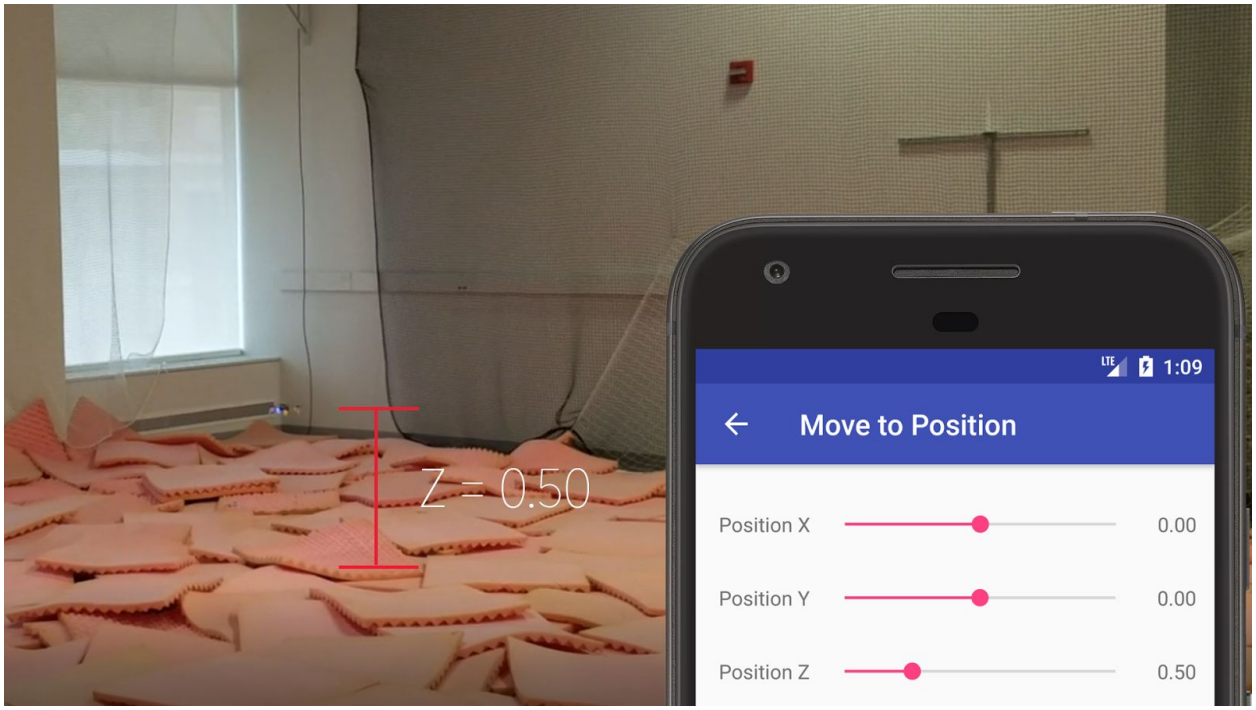

Figure 8. Move To Position (Debug) command, with default X, Y parameters, Z=0.50

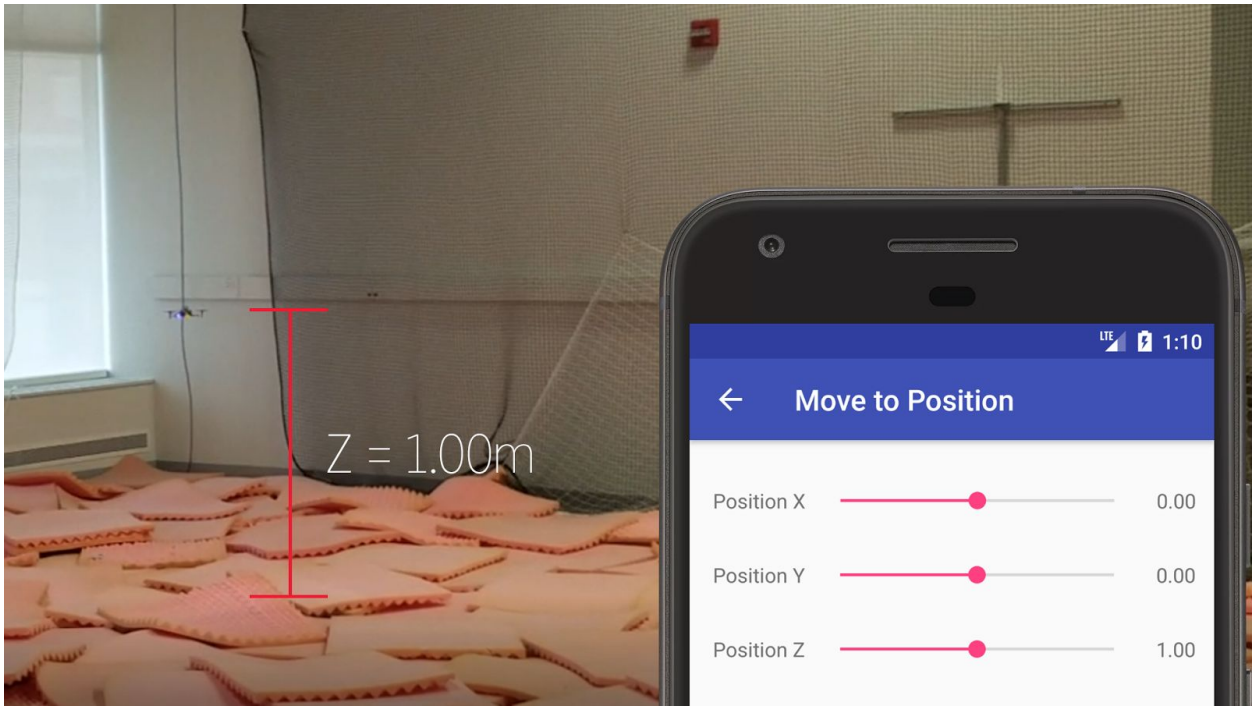

Figure 9. Move To Position (Debug) command, with default X, Y parameters, Z=1.00

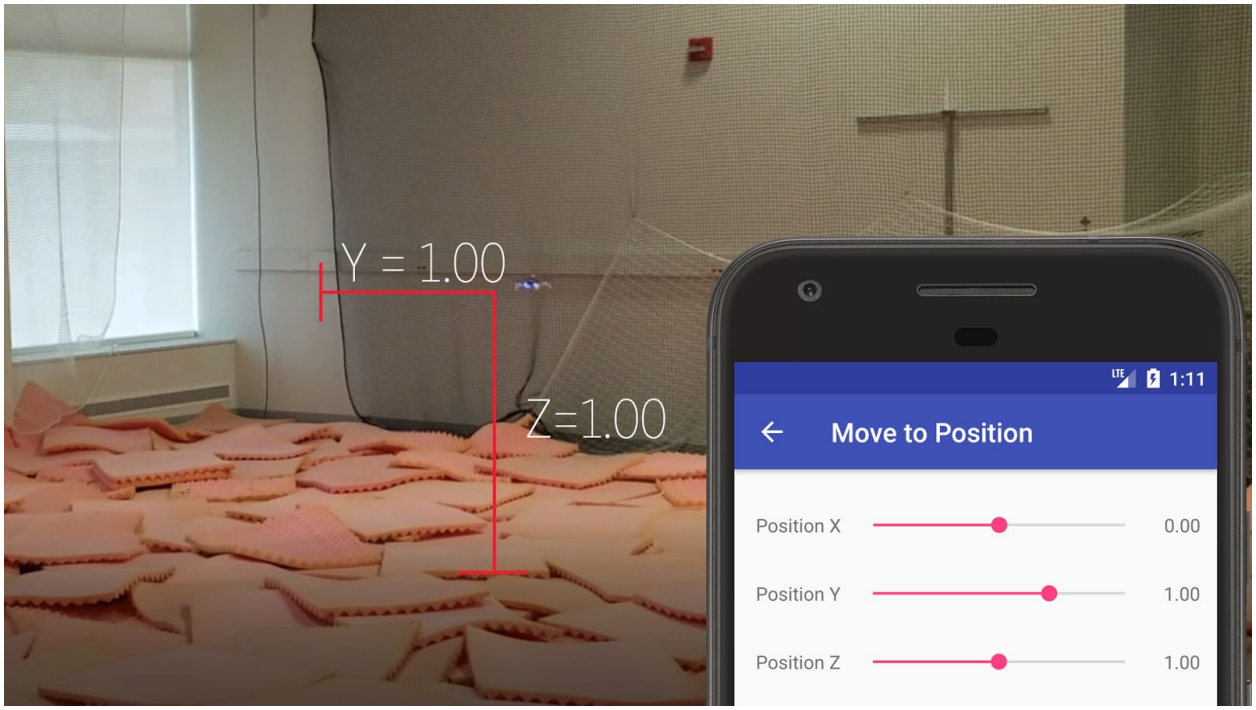

Figure 10. Move To Position (Debug) command, with default X parameter, Y=1.00, Z=1.00

Future Work and Conclusion

The Task Delegation Interface is fully open to the implementation of supporting future features. Currently on our scope, we aim to develop:

- Allow worker threads to complete task requests asynchronously
    - Currently, the TDI only serves one request at a time, and is not scalable for multiple-client connections
- Allow connection of multiple clients to the TDI simultaneously
    - Multiple clients mimic real world conditions (i.e. multiple devices controlling a set of drones)
- Allow queuing of tasks from multiple clients

- Different users can interact with the drones, and by defining priority, the order of incoming actions can be modified.

With completed development of the Task Delegation Interface, developers can now design their own applications that will control a set of drones around a household. The device must only need the capability to establish a TCP connection and send data through it. Human Interface Devices are limitless: laptops, tablets, smartphones, wearables, voice interfaces, microcontrollers are all possible candidates. Developers will be provided with a specification guide as a foundation to connect their applications to the Task Delegation Interface.

Acknowledgements

This research was primarily supported by the National Science Foundation through the University of Illinois at Urbana-Champaign Intelligent Robotics Laboratory, under Award Number 1525900.