ARTeMIS: An Autonomous AR.Drone Project
Renner Brown and Alexander Eggers
The Episcopal School of Dallas

ARTeMIS
An Autonomous AR.Drone Project

This year, the two of us along with a third student, Michael, took an advanced computer science course at ESD. The main focus of the course was a project of our choice, as long as it was approved by our teacher, Deb Goudy. After some thought, Michael and Alexander, who are both also on the school's rowing team, were inspired by a remote-controlled GoPro quadcopter and suggested we make an autonomous drone that could follow and film boats during practice. This would allow an easy way to critique and improve form without the coach having to watch the boat the entire time. Initially, we came up with a bunch of ideas-- it could fly over water and would have pontoons for emergency landings, it would be able to fly back to shore automatically when battery ran low, and we even thought about making a charging dock so that it could land, charge up quickly, and head back out when it was ready. Of course we were getting a bit ahead of ourselves, and after further thought, we decided to focus on making a drone that could simply follow a GPS tracker. And after a class of brainstorming, we decided to name the project ARTeMIS, an acronym for **A**erial **R**econnaissance **Te**trarotor **M**obile **I**ntelligent **S**ystem.

During our initial project planning, we identified a few aspects of the project that we thought would be most challenging to allow us to better focus our efforts. Initially, we thought that creating working tracking algorithms would be difficult. We also were concerned we would be limited by the drone's onboard processing power, which led us to utilize external hardware for processing of GPS data and tracking calculations. Finally, with extra hardware added, we were concerned that the weight of these components would be a serious concern. All these concerns affected our initial hardware decisions, but as it turned out, they were the least of our problems.

Throughout our work on ARTeMIS, Mrs. Goudy required that we document our decisions, progress, setbacks, and any problems. Our main form of documentation took the form of a blog in which we wrote posts and posted pictures and diagrams throughout the project. To supplement this blog, we uploaded a few videos to YouTube and posted pictures to an Imgur account. We also made charts and graphs in Google Drive to show pros and cons of different hardware options, to keep a log of test flights, and to generate a testing protocol to follow to prevent damaging the hardware. By creating documentation we were forced to organize our thoughts in a way that was easy to refer to when we needed to remind ourselves of any aspect of the progress. However, as we neared our end-of-year deadline for finish the project, our documentation became much less thorough since we were spending all our time working on the system itself.

The first major hardware decision we had to make was what kind of quad copter we wanted to get. Our selection was fairly obvious and unanimous. We chose the Parrot AR.Drone because of its user-friendly design, its popularity, and its widespread use in DIY autonomous projects, which meant that it already had a wide technical support base on the internet. Also, because of its use in the KIPR Autonomous Aerial Robot Game, we figured we would have information and support through our friends and contacts within KIPR if we needed it. We chose the AR.Drone 2.0 over the original AR.Drone simply because it is newer (perhaps not the best choice). We also got the special "power edition" from Amazon because of its packaging with

spare parts and more powerful batteries that would offer an increased flight time of around twenty minutes.

With the drone purchased and having done plenty of preliminary “testing” with the phone app, we had to figure out how to control the drone autonomously. We decided that a good step toward this goal would be to learn how to send flight commands from one of our laptops, a process that could later be applied to whatever external processor we used on the drone. We were wary of hardwiring into the drone’s board and risking frying the electronics, so we decided to communicate with the drone via Wi-Fi, much as the iPhone app does. After some research, we thought our best option was a JavaScript platform called node.js. With JavaScript already designed for web-based applications, an open-source API for node.js called NodeCopter offered a simple way to communicate with the drone over Wi-Fi through either live commands or pre-programmed JavaScript.

Next, we had to select our external processing hardware. Confronted by a plethora of microcomputers and DIY electronics boards, we created a set of criteria to narrow down the options and make a selection. Because we wanted to communicate with the drone over Wi-Fi, we wanted a board that had built-in Wi-Fi. Also we wanted the ability to run node.js, since it offered such easy communication with the drone. Finally, we wanted the board to have an easy way to interface with external hardware such as GPS sensors. These criteria led us to the pcDuino v2, a small board with a built-in Wi-Fi module, USB, HDMI, Arduino headers and functionality for external sensors, and a full Linux operating system. Finally, we selected a GPS sensor, choosing the Parallax PMB-688, which is a common choice for those using Arduino-based GPS projects.

Our project faced many problems throughout its slow and often backwards progress. Along the way, we fried multiple electronic components including a pcDuino, a GPS sensor, and even at one point the main board on our AR.Drone. Luckily, through incredible financial support from Mrs. Goudy and ESD, we were able to quickly replace these parts. Most of the other problems that we faced came from the pcDuino. It is a fantastic board for some things but for our purposes, it was a nightmare. It had limited storage capacity, so we often had trouble installing the necessary software and development kits. Also, we had extensive trouble with the serial communications through Arduino pins, making it very difficult to use the GPS sensor. In addition, some Arduino software libraries, primarily SoftwareSerial, a library for utilizing standard digital pins as Serial ports, had serious compatibility issues with the pcDuino. Finally, we simply could not find a way to install node.js on the pcDuino in a form that was compatible with the software libraries needed for communication with the drone. After three quarters of the school year beating our heads against these problems, we finally decided to give up the pcDuino and explore a different option.

Because of all the problems with pcDuino, we decided to switch to Arduino, a platform with which we already had some experience. Instead of trying to work with Wi-Fi communication again, we decided to switch to a hard-wired approach that would communicate directly with the drone through the serial debug port on its main board. We quickly found MiruMod, a very common modification to the AR.Drone that allows it to communicate with an Arduino board without Wi-Fi. Although MiruMod is intended primarily for using a radio controller with the drone, it can also be used in conjunction with GPS. After wiring the Arduino into the drone and toying around with the software for a week or so, we had the drone taking off and reading GPS coordinates from the attached sensor, more progress than we had made the whole year with pcDuino. To create a tracking beacon for the drone to follow, we got another

Arduino Uno board and GPS sensor, and equipped the beacon and the drone's Arduino each with an XBee, a radio frequency communication module. These allow the tracking beacon to transmit its GPS coordinates to the board on the drone, where they are compared to the drone's coordinates to create flight commands for the AR.Drone.

The system now consists of two Arduino Uno boards, two PMB-688 GPS sensors, two XBee modules, a SparkFun bidirectional logic level converter which simply shifts the voltage between the Arduino (5V) and the AR.Drone(3.3V), two wireless proto shields—Arduino expansion boards that allow mounting of the XBee modules and soldering of any external wires into a clean, permanent circuit—and of course a Parrot AR.Drone 2.0 (with a replaced main board).

The tracking beacon reads GPS coordinates from its PMB-688, parses out the necessary North and West longitude and latitude values and its speed from the strings of information, and sends them over the onboard XBee module to the board on the drone. The XBee on the drone's Arduino board receives the coordinates and the Arduino compares them to coordinates from its own GPS sensor. Because the drone is operating over only short distances, we consider the longitude and latitude coordinates as if they are one a standard Pythagorean plane. This allows the drone to use simple trigonometric calculations to adjust its heading to face the beacon and then to adjust its forward speed, going faster or slower until it reaches a desired following distance and then matching it to that of the object it is tracking.

The tracking isn't perfectly smooth or accurate for a few reasons. First there are considerable latencies in receiving GPS coordinates and the communications between the beacon and drone. Also, because of packet loss between the two XBee modules, we had to create a system for checking the values that involves taking multiple readings and creating a composite. This allows the coordinates to update approximately once every second, which is far from ideal, but we wouldn't want an errant value to send the drone flying off towards Canada.

Although the project isn't nearly as successful as we initially envisioned—it definitely isn't ready to fly over water and record rowing practices—it has still offered an incredible opportunity to develop our electrical engineering, computer science, and problem-solving skills. We got to work with C, C++, JavaScript, Linux, and Arduino, and we even got some practice in circuit design and soldering. Most importantly, we learned crucial project management skills from our failures. We learned that extensive initial research is incredibly important to any successful project, and that if you are facing a seemingly impossible problem a better alternative to trying to solve it is avoiding it entirely. We also learned that if you don't know much about what you are working on, you should find someone who does rather than searching around yourself for information.

Of course, all of this would not have been possible without the help and support of Mrs. Goudy and the Computer Science Department and the support of ESD as a whole. Therefore, we would like to extend to them our deepest thanks for allowing this opportunity to expand our knowledge of Computer Science and to attain some real-world project management experience.

Further Reading:

Blog: http://esdartemis.wordpress.com/
YouTube: https://www.youtube.com/channel/UCF4SKWay996ljimaE_CAEwA
Imgur: http://esdartemis.imgur.com/all/