Objectives
The main objective of our project was to enable an AR.Drone to autonomously fly a planned flight using GPS data. To do this we needed to create a program that was able to run on-board the drone and control the drone’s movements. We also needed to enable the drone program to read from a GPS unit to acquire GPS coordinates. We also wanted to create an Android tablet application to both manually control the drone and have the ability to send goal GPS coordinates to the drone to perform a flight.

Approach
Our approach was to utilize the already existing hardware at our disposal and minimize the addition of new hardware (additional microcontrollers etc..). This is desirable for a number of reasons. One of which is that less hardware means less weight and in an aerial vehicle, weight is always a primary concern. Additionally more hardware means higher cost. Naturally minimization of cost is a high priority in any endeavor.

Implementation
Our implementation of our project consisted of implementing the GPS modification, Android tablet application, and on-board flight program. The GPS modification was accomplished by sending the GPS device’s signals directly to the serial RX pin located on the bottom of the drone. The GPS device’s natural output voltage was 3.2V, so we had to use a logic level converter to drop-down this voltage to around 1.8V. Our drone program then created a thread to read from the serial port (/dev/ttyO3) once every second.

The Android application was designed with two operating modes. Manual, and autonomous flight (autopilot). Communication to the drone was done via sockets using Asynchronous tasks to create threads on which the communication was done. This is so that if the thread locks or crashes, it doesn’t freeze the main application.

For the manual controls, the tablet’s magnetometer and accelerometer was used, so that flight was controlled by tilting the tablet. The degree of the tilt controls the speed of the drone, so the further you tilt the tablet, the faster the drone will go. Rotations and vertical movement was achieved using onscreen buttons. There is also a take off and land command, as well as an emergency mode button, which was later disabled and replaced as a connection test button, as it only flashes the LEDs (despite still being labeled Emergency). The actions are executed by using the drone’s AT commands.

For the autopilot, the Google Maps API was implemented into the application, so that a
window was visible of the local area, so that one could see where they’re selecting the coordinates for the drone to fly to. Selecting the coordinates on the map places a marker. Upon using the Set button, the coordinates are saved onto a table, and denoted on the map with a marker indicating the order of stored coordinates. The coordinates in the table can be selected and deleted. Upon using the send command, the coordinates are written into a string using a separator ‘|’ to denote individual points. The string is then sent to the drone to be parsed, and the flight executed.

The onboard flight program synthesized the incoming data streams and computed the appropriate flight control commands to send to the drone’s driver software using UDP and the command protocol established by the drone manufacturer.

These incoming data streams include the GPS updates at 1 Hz as well as the drone's sensor readings at around 100 Hz. The combination of these data streams as well as the uploaded flight plan coordinates are used to compute the desired heading, the adjustments necessary to angular speed to achieve that heading and a determination of whether the current goal waypoint has been reached.

Experiments

Our group performed a number of experiments to test different aspects of our implementation. To test our GPS setup, we had to perform a number of tests viewing the sending signal using an oscilloscope. It was there that we noticed a dangerous spike that would have fried the drone if plugged in at that point. We eventually came up with the solution of attaching a resistor to the outward TX wire. We then did a number of non-flying tests to test the accuracy of the GPS signals our drone was receiving. One such test brought up an error in how we read the longitude coordinates. We were only reading the value for our coordinate and not the heading information (ex. E or W) and thus were leaving out an important negative sign with our coordinate. This caused our drone to calculate incorrect headings to a goal coordinate. Another test we performed after our demonstration was to walk around with the drone and collect GPS coordinates data. After analyzing the data we found that our GPS was not producing accurate coordinates when we approached any building.

Lessons Learned

We learned a great deal about successfully planning and executing a project. We learned how to find the tools we need as well as to recognize the time and place to create our own tools. Making an idea come to fruition over a long period of time was a daunting but rewarding experience.