D4 Individual Report

Joseph Hindmarsh

## Contribution

During the initial design meetings of the first few days of the project, design responsibilities of subsystems were split up amongst the team. I was tasked with the interface of the gyroscope/accelerometer unit with the micro-controller, the payload system and integrating the two microcontrollers together. By the end of the project I had set up the gyroscope to send values to the Arduino Leonardo; designed and produced the motorised cargo hook and investigated the use of an SD card to log flight data. I also assisted the team with the tuning of the PID controller over the final weekend of construction.

## Specification

For the gyroscope and its interface, I was trying to design a system that could reach 100 readings every second and then convert them from angular velocities into angular offsets from the vertical, so to be used by the PID controller to stabilise the UAV while in flight. The specifications for the payload system were: the hook had to keep the cargo secure in flight with pitches of up to 10 degrees, and the ring mounted package had to be able to slide off the hook at its lowest position while the UAV was in level flight.

The specification for the SD card interface was that it should be capable of logging data packets at a rate of 1 packet/second, where the packets would consist of the battery level and the gyroscope data. As the data logging was relatively low priority compared to the critical path of data from the receiver to the PID controller, a fast protocol was required to keep the overheads as low as possible.

## Design and Simulation

The Gyroscope was the first task on the list to be designed, as it was a key part of the core flight systems of the UAV. The gyroscope module chosen was the MPU-6050, as it is a common choice of IMU (Inertial Measurement Unit) for low power applications, and therefore is well supported with software libraries especially among the Arduino community.

To make the budget stretch further additional cost savings were considered with the option of just buying the IMU and constructing my own breakout board from stripboard, but the IMU was only available from the suppliers in QFN (Quad Flat No-Lead) packages. This would have required a separate pinout board and learning the new soldering techniques to solder them, and so the idea was rejected as too risky for only a small cost saving of £2.

The I2C protocol is a common interface for sensors, as often in electronic systems where they are deployed there will be a suite of different sensors for the controlling microprocessor to communicate with. I2C uses a simple 2 wire bus (one clock line, one data line) independent of the number of devices in the system which makes it an excellent choice for systems with many inter-connected devices. It was not ideal for the UAV project as there was only two devices on the bus – and so sending a slave device address for every read is unnecessary overhead for the communications. This was the only communications protocol available on the selected sensor, and as the specification only required 100 packets per second the I2C did the job fine, running in 400kHz mode. The register ordering helped alleviate this problem, as the gyroscope, accelerometer and thermometer registers are all sequential, which allowed all the required data to be read in a single ‘Burst Read’ that only requires the acknowledge/not acknowledge handshaking between each sequential byte read rather than needing to resend the slave address byte every time. This helps to reduce the overhead of unnecessary data on the bus.

The initial design for the interface was based around communicating with an Il Matto atmega 644p microcontroller, using the I2C library at [1] to attempt to open communications with the gyroscope. To start the investigations, the datasheet was used to

## Testing and Results

## Management and Team Working

The work was planned out in order of priority to the finished product. Any subsystems part of the ‘critical path’ of the UAV control were undertaken first, followed by systems that other members of the team needed to test integration. Last on the list was the low priority tasks that were standalone extensions to the core scope of the project. This approach led to the gyroscope interfacing being tackled first. By this point the UART interface between the two microcontrollers had already been completed by another team member to remove a potential bottleneck caused by the gyroscope interface taking longer than initially planned. Developing the motorised hook then took precedence in order to fulfil the specification. Once this was complete, all the main features of the UAV were either completed or in development by other members of the team. This led to the start of investigation into the additional extension features such as the introduction of an SD Card to the il Matto for additional onboard logging.

Team Ganges was managed effectively and flexibly, enabling the team to react to new events and challenges whilst keeping the project on schedule. Each team member was assigned an additional management-focussed role alongside their main design responsibilities to spread the load. These additional roles were:

* Team Leader: Ben Rowlinson, managed the overall team progress, priorities and performance. Set the agenda for meetings.
* Secretary: Joel Trickey, took minutes in meetings
* Financial Manager: Lawrence Gray, worked out costings and managed the budget
* Stores Liasion: Joseph Hindmarsh, managed interactions with the lab staff
* Documentation: Mohammed Ibrahim, organised the Git Repository for document sharing and code collaboration

It was decided at the outset of the project that a daily meeting structure was the most effective way of ensuring that all team members knew how the project and its subsystems were progressing. This frequent interaction ensured that interfacing between subsystems would be as seamless as possible and enabled the team to flexibly reassign resources as needed to deal with unexpected issues an. An example of this would be the chassis fracturing during testing on the final weekend of the project. Improvements to the chassis were designed while the damaged chassis was strapped up to allow continued tuning of the PID controller.

Work was initially split evenly amongst the group based on a brief skills audit of the members, although this did change slightly over the course of the project to keep all team members with tasks to work on. The work was planned through the use of the high-level block diagram to establish the critical path of the systems – mainly focussing on the communications links between the various microcontrollers. Assigning all team members to a section of this critical path ensured that the core functionality of the UAV would be operational as quickly as possible. This meant that work on integrating the work across the team could begin early on to allow time for solving any compatibility issues that arose between subsystems.

## Critical Evaluation and Reflection

Although the final submitted prototype UAV cannot achieve stable flight, this was due to the PID tuning not being completed in time for final testing before the submission deadline rather than any electronics or programming not functioning or being incomplete. We knew that the PID tuning would take time, but didn’t count on wasting 8 hours of the final weekend due to uncalibrated ESCs. This level of delay could not have been anticipated by the project manager and occurring too late on in the project for our coping strategies to have an effect.

Overall the team worked well together.

With regard to my own contributions, I would say that from a programming perspective they were limited. My initial approach to the gyroscope interface was flawed as I wanted to experiment with a communications protocol to see if I could interface with the IMU without a dedicated library. Although the I2C knowledge I gained was interesting, due to the very time intensive nature of the project my time would have been better spent looking at the MPU-6050 specific library to get something workable as quickly as possible. This was made apparent by the code I had written being unable to read data correctly or access the DMP as there were internal registers that needed to be configured before any non-zero data could be read.

Nearly all the rest of my time working on the software was based around attempting to understand device libraries, which would often not compile easily. As a result of this my limitations from a lack of understanding of makefiles led to further delays as I tried to work out the correct command to use the provided makefiles. The libraries that I selected to use for the project were large compilations of device libraries for either the Arduino platform or generic AVR microcontrollers, so most of the packaged library was unnecessary. I found the use of generic makefiles with definitions to be very confusing I wasted a lot of time before resorting to manually compiling the required code file by file.

My design of the motorised hook was flawed by lack of true consideration of the materials involved. Making the hook from a thick coat hanger was fine for maintaining its shape under load, but this same property led to it being difficult to work with when bending out the axles for the hook to rotate around. These were not perpendicular to the rest of the hook and caused the hook to jam frequently after only a short distance, not fulfilling its specification of a full range of movement - when mounted on the chassis, the hook could neither retract enough to drop off a load while stable nor properly secure the payload for the full range of angles.

## References

1. http://davidegironi.blogspot.co.uk/2013/02/avr-atmega-mpu6050-gyroscope-and.html