# Week 10 Project Summary – James Davies

# **Servo Specifications**

An important aspect of the project is to understand the limitations of the flywheel response. I started by looking at the servo RPM changes with increased flywheel angular velocity, since the torque required to rotate a flywheel will increase the faster it is spinning. Testing this out at 3 arbitrary speeds of the servo between 44 and 7 RPM, with the flywheel stationary, then spinning at 2500, 5000, 7000. It was noted that the servo response RPM dropped slightly, but very little (under 0.1RPM at most), as the maximum servo output torque was still quite far from the current torques. The servos were just drawing more current at higher loads. This currently isn’t an issue with the small flywheel, but may have to be revisited when larger flywheels are fitted.

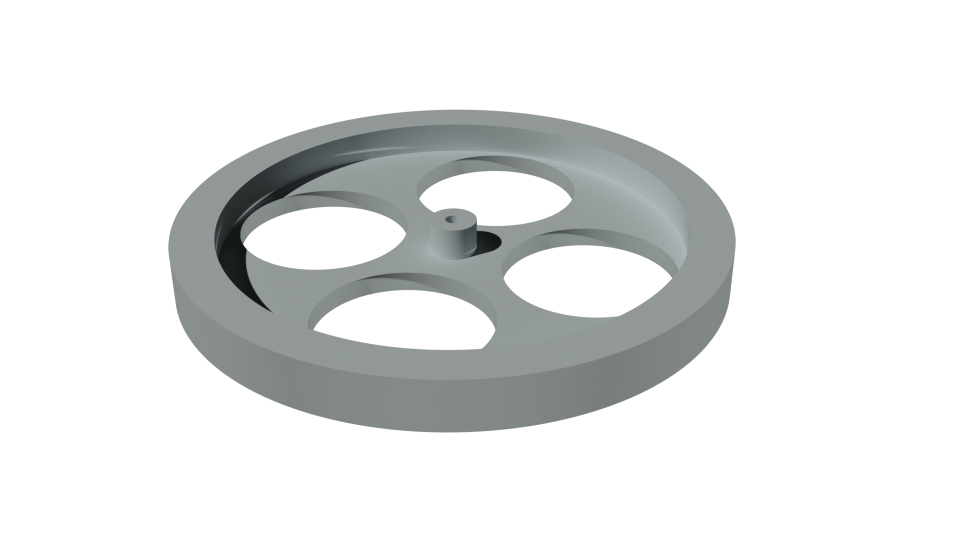
More accurate flywheel control was developed. I discovered in the C++ code that makes up the servo library for arduino’s that if a value of greater than 544 was passed to the “Servo.write()” function, then the output PWM signal changes from the positional defaults, that is very useful for 180-degree servos, to a pulse width of the PWM signal. This is particularly useful due to the integer nature of the functions. This led onto the investigation of the servo RPM response from different pulse widths (i.e voltage), which resulted in a Lorentzian response of the motor.

Even though this correctly produces the servo angular velocity from a pulse width, it is very hard to solve for tpwm quickly, especially only using an Arduino. To solve this, I took advantage of the symmetric function and only consider half the data, from stationary to max RPM. Using this data, I then fitted a polynomial to the function, which was only appropriate over the range of data, but this is all that is required for the function:

This function can be calculated very quickly by the Arduino, and is still very appropriate over the operational range of the servo.

# **Flywheel design**

After the maximum angular velocity of the flywheel was determined, equation 10 could be plotted for Flywheel angular velocity vs Flywheel mass. For this plot the assumption was made for the forces required to correct from a 35 degree lean angle (no inertial component) and plotted for several servo RPM’s.

The graphs revealed a diminishing returns of an increased flywheel mass vs the flywheels angular velocity, but allowed an effect choice of ideal flywheel RPM. With this value for RPM, the equation was rearranged to give the mass of a ring flywheel (considering the moment of inertia of different shapes) to 0.515kg

This mass was then used to design three flywheels. The resulting design as shown here was picked due to its high moment of inertia with minimised mass (for ease of RPM changes), but also due to the considerations of production. This design can be produced from one side in a mill, and be easily mounted to the 4mm rod spun by the motors.

The design was produced used AutoCAD and accurate measurements of the shape, mass and materials have been passed to the technicians for production.

# **Basic PID and Code Responses**

With the servo outputs now understood, work on the system responses to movements started. This began with a simple servo response to and angular changes from the accelerometer in the Arduino controlled by the CurieIMU library’s.

Taking data from the IMU, including accelerometer and gyro data, both measured in 32768 bits then converted to accelerations in units of g and rotation in degrees per second. Using both of these, and taking advantage of an Arduino library sourced from a github repository called Madgwick filter, to produce an angle of roll for the rotation of the frame.

This roll angle and the gyro roll are two contributing factors that need to be corrected for by the induced torques from the rotation of the flywheel. Equations of motions for correcting to the weight contribution of the offset angle and the role contribution have been added to the code with two different functions. This is then passed to the equation above, to change the pulse width of the signal being sent to the servos. I am then using a PID algorithm to produce and multiplicative correction factor which for this pulse change. A stable system was produced, which could correct for small perturbations of the system, but that would fail for larger ones.

# **Mount Changes**

With the simple system having a stable output, I have changed the system to more accurately model a motorbike. Keeping the bike inside the frame but mounting the system higher, by roughly 7-8cm’s produced a much more unstable system, that require greater torques.

The frame is no longer stable with the current code, but can balance for under a second currently before failing.

# **Code Redesign**

The code has been cleaned up and optimised, and is now being versioned as the code being written currently is now being used to achieve the goal.

It has far superior layout and organisation, and more optimised operation.

# **Angle Measurements**

Current work has lead onto the measurement of angle for the flywheels. This is an important factor as the torques produced from the flywheels rotation drop off by a factor of cos(θ), it is one of the main reasons for the system failures.

After discussions with Phil Lightfoot, we have a plan of action to measure this. We intend to use a continuous rotation potentiometer geared up to the servo and to measure a change in voltage (provided from the Arduino) and relate this to the angle of the flywheel.

The code is already written to allow for this angle, and angle measurement code to be added, without large re-writes of functions and running logic.

# **RPM measurement**

Another issue that will come to light with the new flywheels will be the changing in RPM due to losses to friction as the flywheels rotate. To measure the RPM, I am going to take advantage of the flywheel design and shine a led light onto the wheel, with an LDR facing it on the opposite side. Measuring gradient changes (as the spokes pass by the resistor) we can actively measure the RPM. With the work previously done in the circuit design, it is already setup that the gate PWM signal can be changed to dynamically vary the control the flywheels angular velocity.

# **Gantt Chart and plan**

Looking at the Gantt chart below, which was created at the start of the project for the project review, this term has been very successful, and I have achieved everything I planned. However, looking at the goals set for term 2, my aims have shifted, and the controlled lean is not as big of priority currently and will need to be delay to a later part of term 2.

My next few goals I wish to achieve are:  
1. Have a working angle measurement system for the flywheel

2. Have a working RPM measurement and response system into the code.

3. Have a stable system inside the frame, on the heightened system.

4. Have the frame mounted on the wheels before week 8.

# 