# Embedded Systems Coursework Specification

**Coursework 2: Brushless Motor Controller**

## Functional specifications

1. The motor will spin for a defined number of rotations and stop without overshooting.
2. The motor will spin at a defined angular velocity, either continuously or as a maximum while carrying out specification 1.
3. The normal precision is:
   (a) The nearest one rotation for number of rotations
   (b) The nearest one rotations per second for angular velocity
4. Optionally, the motor can operate at high precision:
   (a) The nearest 0.01 rotations for number of rotations
   (b) The nearest 0.01 rotations per second for angular velocity
5. Optionally, the motor can play a melody while it is spinning by modulating the control voltage.
6. Optionally, the motor can automatically tune its control parameters to optimise for a change in the connected moment of inertia.

## Implementation specifications

7. The system will be commanded by instructions sent from a PC over a serial interface.
8. The syntax for rotation commands is the regular expression `(R-?\d{1,3}(\.\d{1,2})?)?(V\d{1,3}(\.\d{1,2})?)?`
9. The syntax for melody commands is the regular expression `T([A-G][#\^]?[1-8]){1,16}` (where # and ^are characters)
10. The system will be implemented using interrupts and robust threading techniques to leave the maximum possible CPU time for background tasks

Notes:

- Examples of rotation commands are R100 (spin for 100 rotations), V-20 (spin backwards indefinitely at 20 rotations per second) and R-350.34V1.32 (spin backwards for 350.34 rotations at a maximum speed of 1.32 rotations per second). If R and V are both specified then the sign of V is ignored.
- An example melody command is TA4C8G4F#8 (T followed by pairs of notes and durations). At the end of the sequence the melody repeats.

## Documentation specifications

The report should contain:

11. A description of the control algorithm used.
12. An analysis of all the tasks that are performed by the system with their minimum initiation intervals, deadlines and maximum execution times.
13. An analysis of inter-task dependencies to show that there is no possibility of deadlock.
14. An analysis of the scheduling system used, showing that all deadlines are met.
15. A quantification of maximum and average CPU utilisation to show what is available for background tasks.

## Implementation notes

- The starter code (`https://developer.mbed.org/users/estott/code/ES_CW2_Starter/`) contains a controller that spins the motor continuously as fast as it can. It uses a polling loop to read the photointerrupters and set the motor field angle to lead the rotor by 2 states (approximately 120°). At startup it records the position of the optical disc with the motor field angle set to 0 and applies this correction in future to translate optical disc position into rotor position.
- Specification 3 should be achievable using only the photointerrupters (I1,I2,I3). Specification 4 will require the use of the incremental encoder (CHA and CHB) as well.
- Do not rely on the rotation states being evenly spread out. For example, the photointerruptors may not change from state 0 to state 1 at exactly 60°. The same is true for the incremental encoder. You should not calculate angular velocity from the time taken to change between successive states (e.g. state 0 → state 1) because your result will vary with the actual angle of each state. Instead, measure the time for complete cycles (e.g. state 0 → state 0, state 1 → state 1). You could try to characterise the actual state transition points and apply corrections from a table but bear in mind that some of the variation is due to the response time of the sensors so the transition angles may vary with velocity.
- The incremental encoder should not be relied upon for keeping track of the motor position alone. The alignment tolerance of the optical disc means that it may occasionally miss states. Also, at high speeds it may be outside its frequency specification, and potentially be overloading your processor with interrupts. Use it at low speed and use the photointerrupters to resynchronise, bearing in mind the previous note.
- If there is a jumper between two pins of your board, please remove it.
- For usage examples of the mbed libraries look at the many example programs. The RTOS is documented with example snippets here: `https://docs.mbed.com/docs/mbed-os-api-reference/en/latest/APIs/tasks/rtos/`