### **Computer Systems 723 Cruise Controller**

William Chao  
Leighton Jonker

Department of Electrical and Computer Engineering

University of Auckland, Auckland, New Zealand

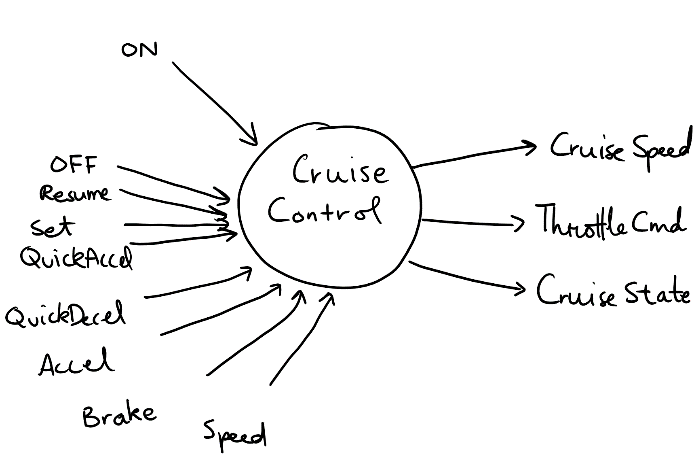
Abstract  
When it comes to the creation of embedded software, there is multiple ways to approach the design process. One such process is the model-based approach. The model based approach involves the development of functional specification models which are then implemented using a synchronous language. For this assignment we will be using the synchronous language Esterel and will be designing a functional cruise control model which will then be implemented through the use of Esterel.

# Introduction

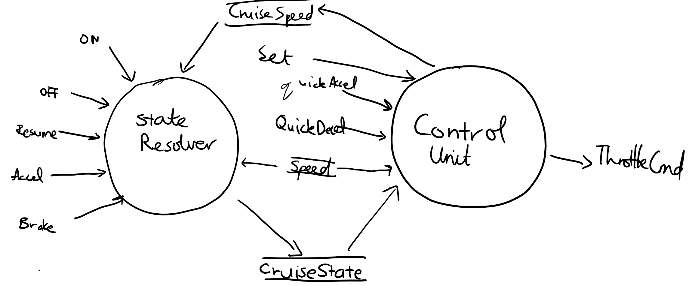
In Computer Systems 723 we have been tasked with the creation of a cruise control system in order to gain hands-on experience in designing embedded software using a model-based approach. We will first start the assignment by creating a functional model that adheres to the specifications we have been given as per the assignment brief. Next, we will implement our model using the synchronous language Esterel. Our goal is to have created an embedded system which is executable and reactive, which adheres to all the given specifications.

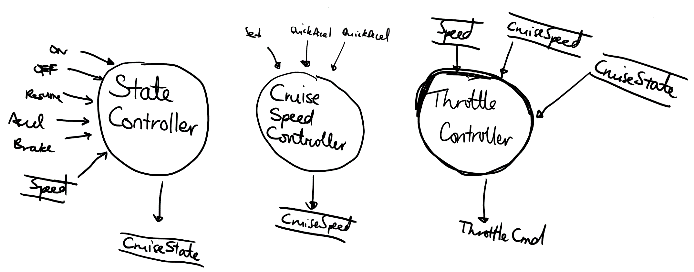
# Diagrams

First, we had to create functional specification diagrams. The diagrams would help us get a better understanding and handle of the system that we were tasked to create as well as help us when it came to the creation of our system. Below are images of our model during development.   
First, we started with a simple model of a Cruise control system which was inputted simple inputs such as ON, OFF, Accel, Brake etc. with outputs such as Cruise State and Cruise Speed.

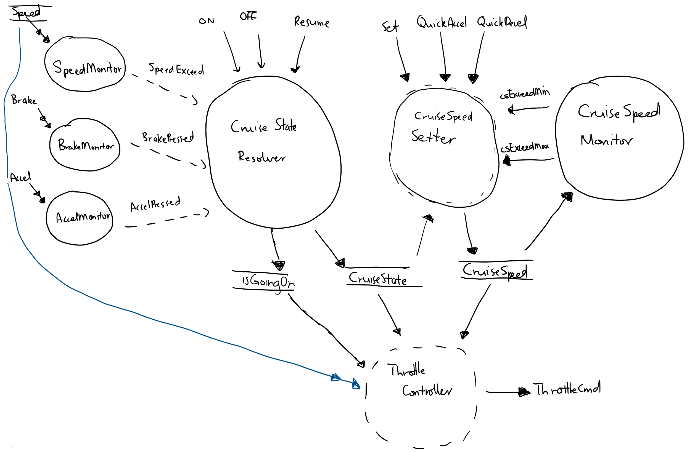


A model such as this was incomplete and simple in our eyes and we felt we needed to break up the signals into more specific sub-modules so that implementation and understanding would be easier. This prompted us to split the previous Cruise Control module into two separate modules called State Resolver and Control Unit. With this refinement we could now easily distinguish between the two main areas of this system; the states and control.

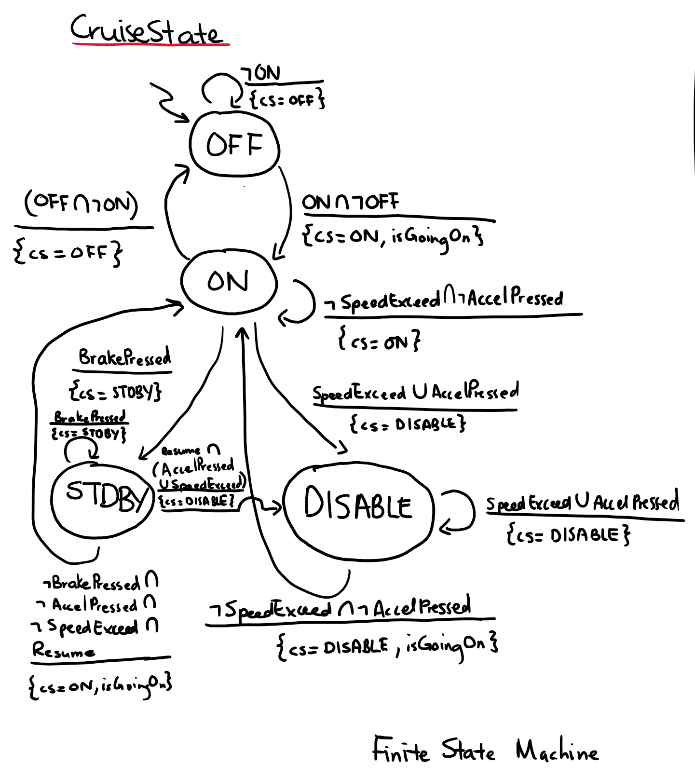


Next, we decided that we could even further separate the Control Unit module into the Cruise Speed Setter and the Throttle Controller for good object-oriented design.   


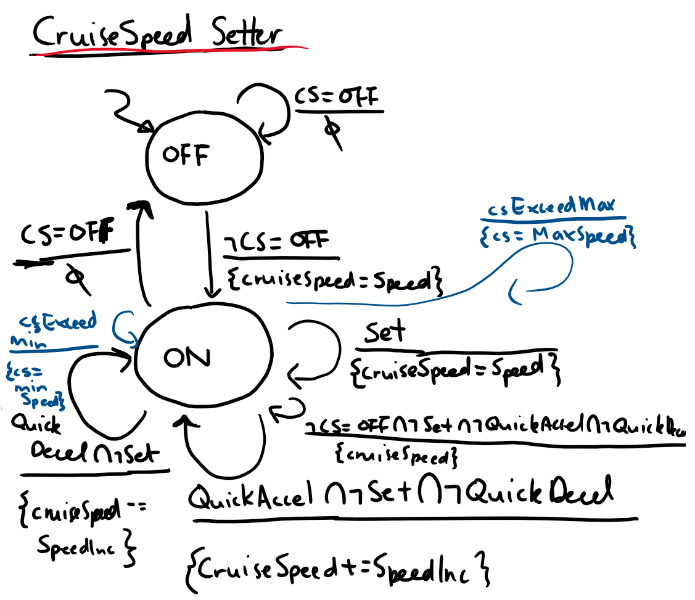
Lastly, we now needed to include monitors to monitor certain inputs from either the user or other modules in our design which could then pass the correct signals to the appropriate modules so they can continue correct functionality. We added an Accel Monitor, Brake Monitor, Speed Monitor and Cruise Speed Monitor. The final version of our specification model is shown below:



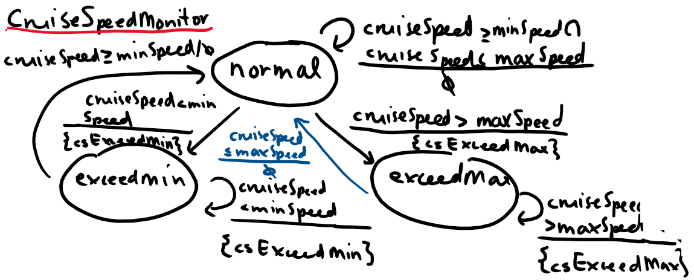
After verifying that our current model fit all the requirements that were stated in the project brief we then started on the creation of the Finite State Machines for each module.



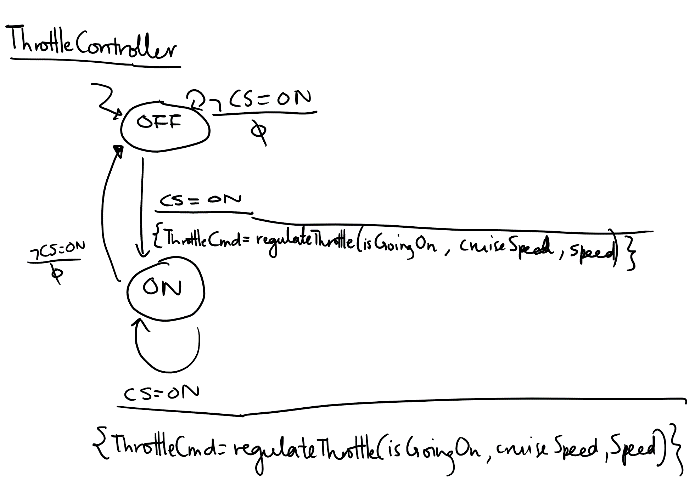
The Cruise State Resolver is one of the most complex modules in the system, and decides what state the cruise controller is currently in. It takes multiple inputs ranging from the current car speeds to binary inputs like OFF or ON and returns the current state the car should be in.



The Cruise Speed setter module is designed to set the speed at which the cruise control system will run the car at when the cruise controller is active.

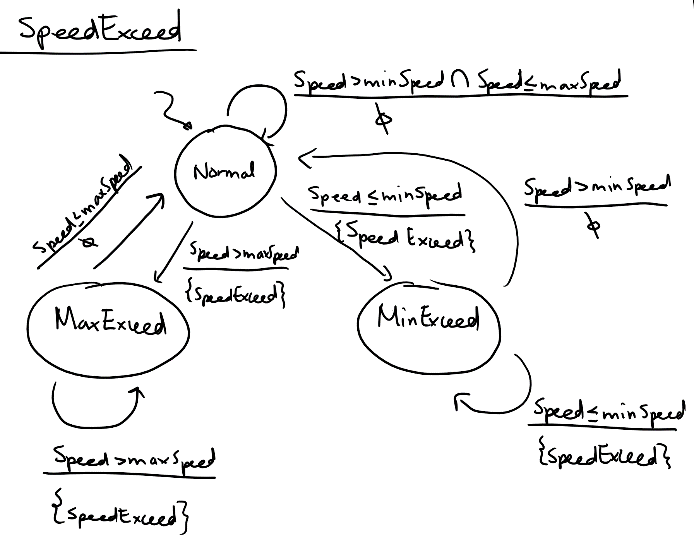


The Cruise Speed Monitor takes in the current cruise speed and determines if the speed exceeds either the minimum or maximum value, at which it will output a signal notifying the Cruise Speed Setter so.

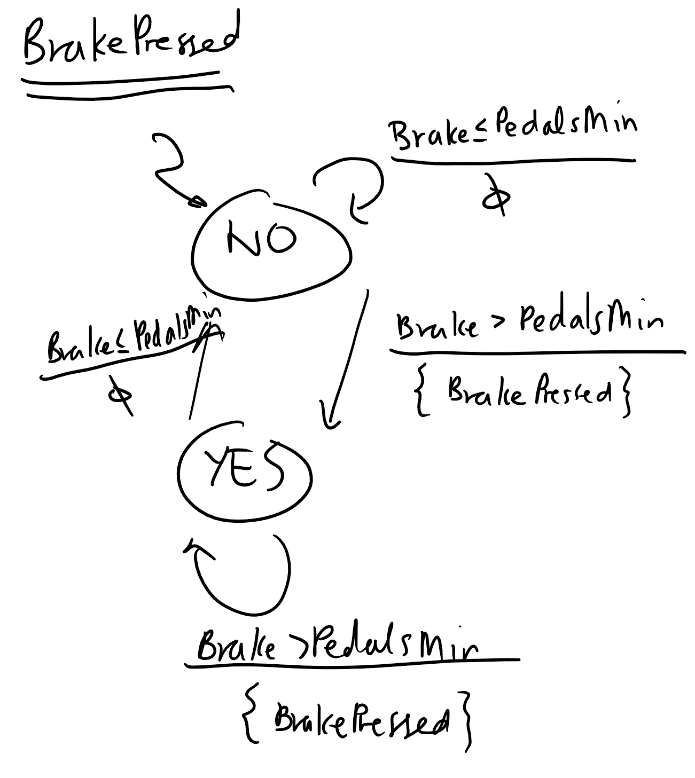


The Throttle Controller considers the current cruise state, cruise speed and speed to calculate what to regulate the throttle of the car at.

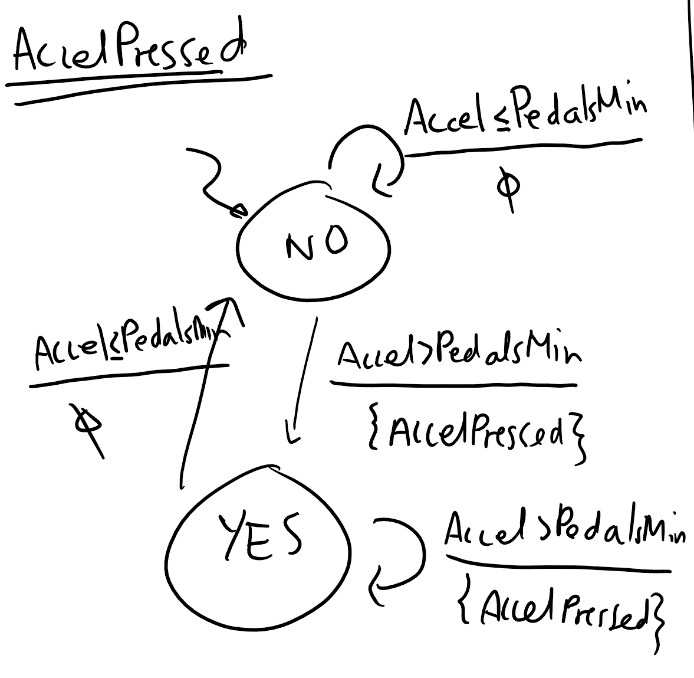
The Finite State Machines for each of the monitors are much simpler than the other Finite State Machines as they mainly focus on one input and act accordingly as opposed to the other modules which are inputted multiple signals and sometimes output a considerable number of signals.



The Speed Monitor is given the current speed of the car and calculates whether the speed exceeds the given min or max limits for car speed.



The Brake Monitor takes the binary brake input and converts it into an output called BrakePressed to signify if the main car’s brake has been pressed.



Finally, the Accel Monitor uses the accelerator binary input to calculate if the car’s accelerator pedal is being pushed or not.

# Implementation

Once we had finalized our designs it was then time to start the implementation. Although we had never used Esterel for programming before we understood how the language worked and decided that the main issue we would face would be the initial creation of the diagram using the given specifications and the Esterel syntax, as we found that some of the specifications given to us were difficult for us to implement as we didn’t know how the syntax for those specifications worked. Thankfully with the help of the lecture slides, TAs and provided reading, we could understand the language more, allowing us to be able to successfully implement our design in Esterel. We started with the conversion of the given specifications of the cruise control system being converted individually into pseudocode and eventually Esterel code alongside the creation of the functional model and Finite State Machines. Once the individual modules had been created and finalized, they needed to be integrated with one-another. The integration was the most difficult part, as it would create the final product for this assignment. Once the integration was complete the system would then need to be verified to ensure it met all the provided specifications in the brief.

In the end our system was able to meet all conditions that were provided to us through verifying it by using the vector in and out files provided by the assignment brief.

# Conclusions

In conclusion, our created cruise control system

# References

# Appendix A