# Control Algorithms

The control algorithms is discrete time software implementation. The control is implemented using Simulink models cross compiled onto the target microcontroller. There are two major portions to the control algorithm, the finite state machine and the PID controllers. An overview of the architecture is given below for a single leg:

# D:\MyDocs\Documents\GitHub\AgileRoboticControls\System Modelling\Control\Control - General.png

Figure : General control architecture for a single leg. There are a total of four PID feedback loops in the full system, one for each leg.

This architecture was selected to ensure a high system responsiveness. The state machine feedback loop involves heavy mathematical calculations to determine the position of each foot during runtime, so it runs much slower than the PID feedback loops. By creating an independent loop for each leg PID controller multiple feedback loops can be run for every system model loop, leading to high responsiveness. This master-slave architecture can be implemented through processor multithreading, and if that does not operate quickly enough additional slave microcontrollers will be purchased to run the PID loops. A detailed diagram of the real time control block diagram is given below:

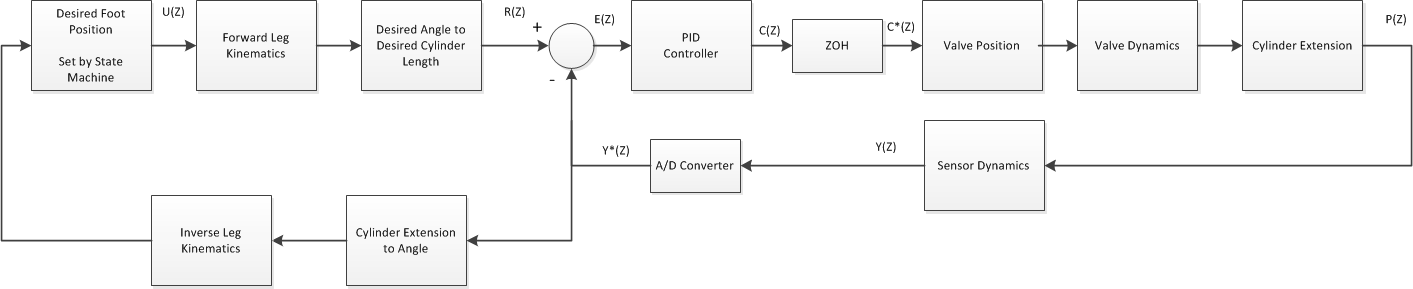


Figure : The master-slave architecture block diagram for a single leg. The left loop is the main system model loop and the right loop is the PID feedback controller. There are four right side loops in total, one for each leg.

The main intelligence of the robot will be implemented using a finite state machine. State machine transitions will be defined based on user input. Currently an initial state machine has been developed that has two states, stand-by and stop. Stand-by is a state in which the robot is determined to be functioning correctly and is awaiting user input. Stop is the state in which the robot completely stops ongoing motion and transitions into a stable position before powering down. Figure ***#*** shows a flowchart representation of the currently proposed state machine architecture:



Figure : Flowchart representation of the state machine architecture.

The state machine has a total of seven states. On system startup the state machine is set to *Initialize* which ensure the communication protocol is functioning and checks all subsystems for faults. When fault checking is completed the system transitions to the *Idle* state and waits for user input. Based on controller input the robot may enter the *Move Forward*, *Move Backward*, *Turn Right*, or *Turn Left* states and transition between them. The transitions between these states depends on the gait implementation of the robot. The above flowchart assumes stationary turns, but if the robot can only turn while walking forward or backward the transitions will be modified as required.

The final state is *Stop* which is transitioned to when a fault is detected or when input by the user. The stop state puts the robot into a stable default position before powering down each subsystem. The stop system will also be transitioned to if the microcontroller detects a power loss in the subsystems resulting from the emergency stop being activated.