Miniproject

Marius Lervik and Endre Leithe

October 2017

Part 1

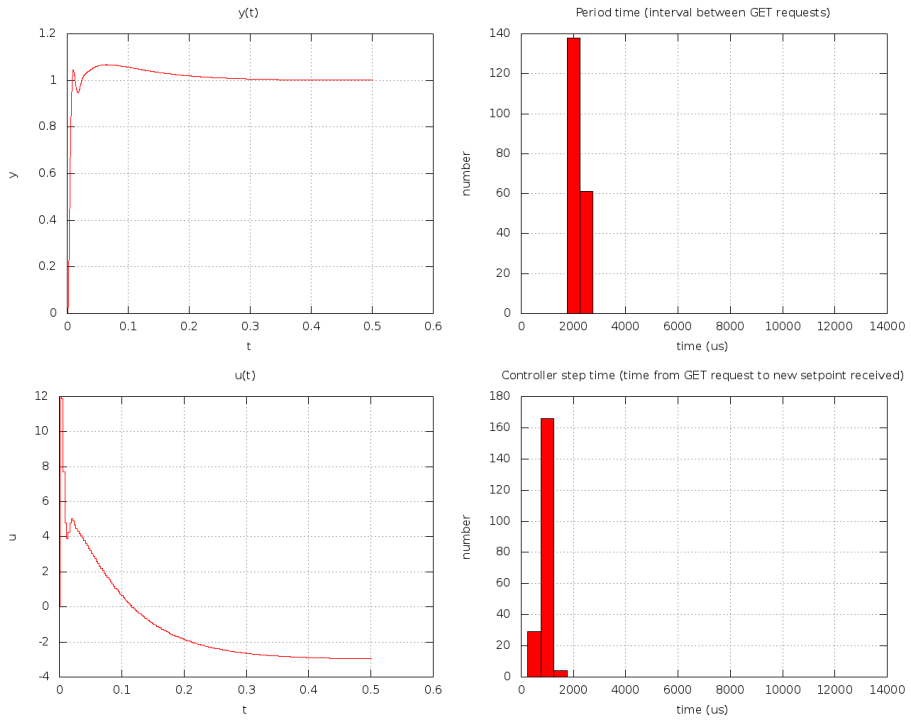

Figure 1: Graphs of simulation result

The results from part1 is shown in fig. 1. The code is in part1.c. In order to find the controller period we looked at the results from the simulation. The graph for controller set time shows us that we can choose period at 2000us and we will still be able to meet the deadline. To play it safe we choose a period of 2500 us. This task was explained in detail in the assignment text and the code was not complex. We decided to make a function for calculating u. Here we found a nice use for the static type for keeping track of the integral. Because the server also sends at SIGNAl messages, we had to ignore these messages or else we got some weird results when calculating u. We fixed this by using the goto statement to receive the next message. We were satisfied with the response of the server simulation.

Part 2

The results from part2 is shown in fig. 2. The code is in part2.c. We used the same method as in part1 to find the controller period. We chose a period of

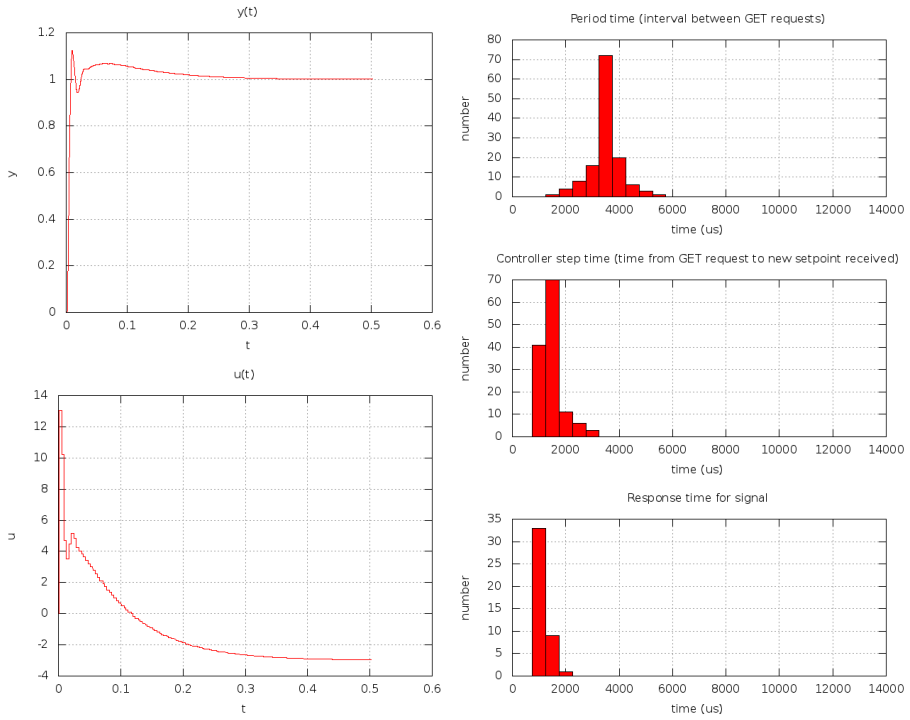

Figure 2: Graphs of simulation result

3800us. We used three threads in this task as stated by the assignment text. For synchronization between the threads we used semaphores and one mutex. The thread that received upd messages used a global variable to store the value of y. After storing y it signaled the semaphore for y_received. This is fine because we only receive a y after sending a GET message, so there will never be any race conditions. The thread that implements the PI-controller is very similar to the main function in part1. When receiving a signal message the udp_listener thread signaled the semaphore for signal_received. The respond thread will then send a ACK message. The mutex was used to make sure only one thread called udp send at a time. The extra threads and overhead effected the controller step time and period time. We had to choose a larger period compared to part1. This was expected. As a result of this the PID controller performed a bit worse than in part1.

From the graph in fig. 2 we see that the response time for signal is quite high. This is because we have chosen the period such that the PI_controller thread will not be idle for most of the time. If we wanted to lower the signal response time we could have increased the controller period such that the PI_controller thread would spend more time sleeping. Then the response thread would get to run more and we would get lower signal response times. However the assignment

text did not specify the importance of a low signal response time.