# #3

Rushang V Karia

*This Lab report is meant to be descriptive of what I learnt when I did this project. For a more accurate representation please see the assignment manual OR please check the following file and line # < Target.h #255 > for the structure of the program*

## TARGET IMPLEMENTATION DETAILS

The co-ordinate transforms were split as required. There are two threads. The main thread which constitutes the controller and is synchronous on a timer.

The second thread is the mouse thread.

To change the initial angle/Link settings please change untitled_data.c and look for the appropriate constant that you want to change.

UDP Receive should be  > 0 in target. It can be 0.0000001 also. All that is need is the host to send data first. After that communication begins.

## WHY MATLAB APPEARS TO STOP DEAD

It is because of the initial block at the UDP. After that it should be fine since it does not do any particularly time consuming applications. Also MATLAB appears to be single-threaded to Windows for some reason. Be ready to call Ctrl-Alt-Delete for help!!

## A NOTE ON THE INITIAL GRAPH.

The initial graph makes the robot come from ITS start position to our start position that is (0,0). This happens because the target udp receives the ROBOT START POSITION first and passes it along to the Tx Ty decoder who then packs it and sends it to the HOST. So the host first draws [0.429,0]. The host also computes the (dx) values for xyz=[0,0,0] that we sent. So it sends this to the target which then makes the host traverse to [0,0] gradually since it is a control law computation.

Once the host reaches (0,0) both the host and target and now in sync and there should be a 1-1 correspondence. By correspondence we mean that host_x=K(target_X). Again the argument is control law computation.

## WHY SHOULD THERE BE TWO THREADS??

The mouse should be an autonomous device. It is like driving a car. Even if you do not hold the steering wheel the car still moves. There should be no need for you to hold the steering wheel at all times.

Similarly just because the robot acts on events at specific time instant does not mean that we cannot provide input. The mouse generates continuous data and therefore should be asynchronous.

# #3

# Rushang V Karia

## SOME NOTES ON THE MOUSE IMPLEMENTATION

**I use/dev/input/mouse*…if you are sure that there is only one mouse feel free to use /dev/input/mice**

/dev/input/mouse* uses the PS2 mouse protocol.

**Movement Data Packet**

The standard PS/2 mouse sends movement/button information to the host using the following 3-byte packet:

| | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| **Byte 1** | Y overflow | X overflow | Y sign bit | X sign bit | Always 1 | Middle Btn | Right Btn | Left Btn |
| **Byte 2** | X movement | | | | | | | |
| **Byte 3** | Y movement | | | | | | | |

The range of values that can be expressed is -255 to +255. So assuming you are fast enough, your robot might receive false data since I have NOT incorportated. *{Referenced from Internet}*

## TWEAKS THAT CAN BE MADE TO THE TARGET

1. You can decrease the time period so that the robot graph appears more smooth in plotting. This is obvious since the rate transition buffer is read only at those intervals.
2. The target outputs X=0, Y=0 as the initial values. However the target is not set to that initial value. As a results the robot is made to first come to 0,0. This can be changed by adding

## HIDDEN DANGER

1. Matlab does not specify the size of the rate transition buffer. So it is possible that many events might be missed. For the current assignment, it specifies a size of 2 but that does not make sense. [target.h #74] Maybe I am wrong.
   While using Simulink for real projects this is very application specific and must be known.
2. Matlab uses faster builds by default for generation. Please ensure that this option is set to optimized builds for embedded applications.

# #3

# Rushang V Karia

**POSSIBLE ERRORS DURING IMPLEMENTATION { Some were faced by me!!}**

1. Please make sure that the blocking time in the target in NOT infinite!!!
2. There is a very rare error of port corruption:: There is a slight chance that the target thread does not exit. So it keeps on generating events on the same port no. So if you decide to restart the program you will see output even if you never started your target application. The only workaround is to kill the offending process. A restart of the target will fix that or you can manually netstat and kill the process.
   <u>Hint: There is a very high chance that this is the problem if you do not see the robot graph move!!</u>
3. During building S-functions sometimes they will spurt out errors like sfunxy flag. These are harmless and press run again to make them go away
4. Please give data to the robot. It might go mad if you don't provide input. This is due to the precision of floating point and since the computation is changing it by some exponential or other function {Didn't look that closely}. But yes after a few values the difference in precision is sufficient enough to cause huge changes in the values.
5. Make sure that if setting receive IP to '0.0.0.0' no other "hacker" is trying to poll the port.

**WORKING OF THE TARGET { This might be a little inaccurate }**

The program starts by calling target_initialize() which creates all threads for the s-functions and sets up the entire environment. It preserves the order based on the priority you gave and pthread_create() is called in that order. So in effect it guarantees that processes of different priority are STARTED in order.

The other steps are usually implementation dependent.

In this case, once everything starts, the host computes the initial values and sends to host who is blocked. After that the whole procedure kickstarts. The mouse places all events asynchronously into the buffer. The target picks up packets at clock events and redo's the computation and sends again.

Recursive loop.

**A NOTE ON CODE GENERATION**

Code generation is a very powerful tool to use. However looking at the files *_data.h and other files, its use in an embedded application can be questionable. However this is merely on observation. A more complete test and profile suite is absolutely necessary to quantify any claims made.

*A greater effort must be made towards code generation for embedded applications and a community be put together so that a much vaster library can be made.*