EE/CE 6302 Microprocessor System
Final Project
Report

N-BODY PROBLEM

Guided by
Professor Roozbeh Jafari

Designed by
Mohit Mahesh Bansal MMB140030
Samir Senapati Polobody SXP144330

By signing this form, I certify that any work submitted is original and belongs to our group unless adequately referenced. I certify that I have not made available this work to any other students in class.

12/06/2014

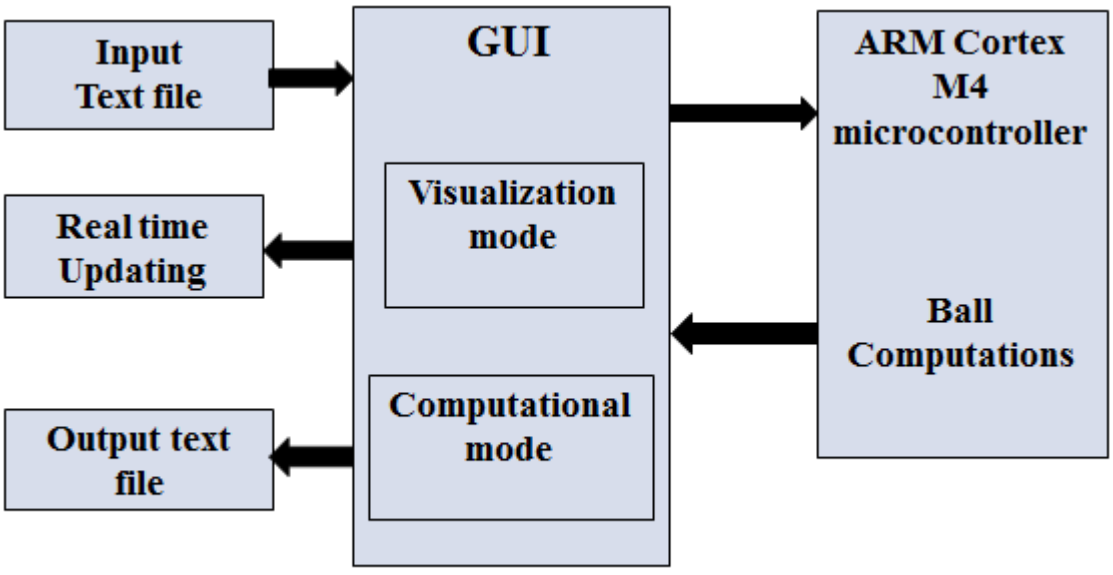
Project Objectives:

Our main aim of project was to use the Launchpad to simulate and iteratively compute solutions for the "N-body problem", i.e., simulate the motion of a given set of planetary objects as governed by the effects of the forces of gravity on each other and simulate it within square area bounded by non-rigid boundaries by enabling communication between launchpad and PC.

MOTIVAT
ION:

1. Understanding of
   - Arm Cortex M4 – Tiva C Launchpad
   - PC Serial Communication using any programming language
2. Real time computation and communication between a microprocessor and host

In project, we implemented two different modes namely Computation mode and Visualization mode which takes care of which operation to perform either to do the computation of input balls' x and y position or visualize the actual position of planets on the GUI.

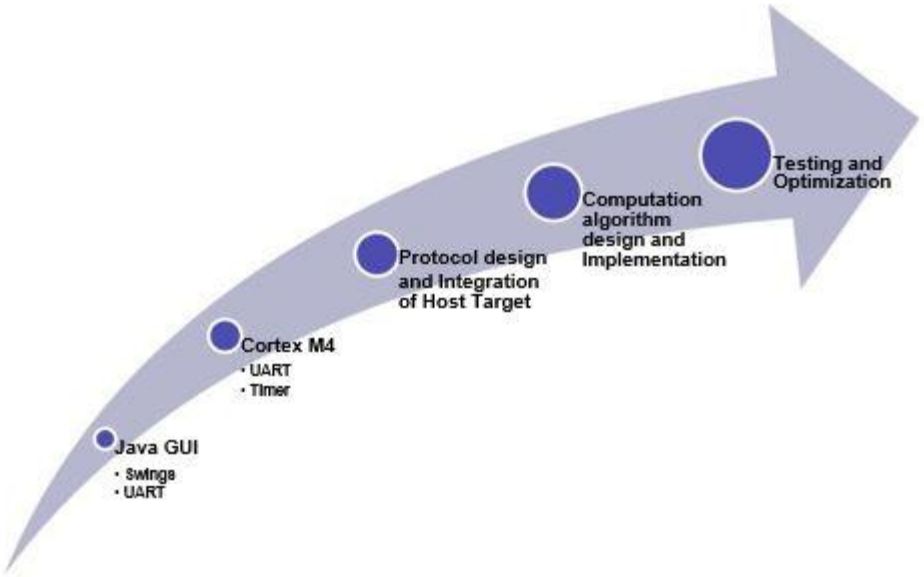


Input file consist of set of bodies (at least 2 bodies) with the following properties for each of them: Mass (in kg), Initial position (x,y) in meters, Initial velocity components (x,y) in m/s.

Specification:

A square area inside which balls simulation is shown is having rigid boundaries hence any ball which hits the boundary then instead of exiting the system it rotates on the boundaries.

Calculation of ball speed and position is done in Launchpad having Arm Cortex M4 which is sending this information to GUI via serial communication (RS - 232).



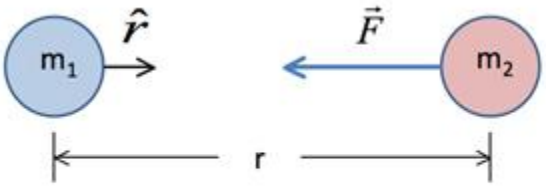
Technical Approach:

The project in basically divided into five parts:

- Developing the GUI using JAVA.
- Develop the protocol for the communication between Launchpad and GUI.
- Design and develop the algorithm for the computation of the forces between N-Body. Which means calculating the forces on a body due to N other bodies present in the universe. This was calculated using Newton's gravitational law.
- Integration of Java GUI and Microcontroller.
- Testing and optimization of the project developed.

Computational Algorithm:-

The computation of the N-body problem is governed based on the gravitational force.
The gravitational force is:-



$$\vec{F} = -\frac{Gm_1m_2}{r^2}\hat{r}$$

The approach that we took is that we calculated the value of acceleration in X_AXIS and Y_AXIS on the body considered due to the different bodies present in the universe. The value of this acceleration is then added for X_AXIS and Y_AXIS, then the result of ACCELARATION_X_AXIS AND ACCELARATION_Y_AXIS are converted into

velocity on X_AXIS AND Y_AXIS. The resultant distance in X_AXIS and Y_AXIS is then calculated by dividing it with the time step.

The formula for calculation of position is as below:-

$$\ddot{\mathbf{y}}_i = G \sum_{j=1, j\neq i}^{N} \frac{m_j(\mathbf{y}_j - \mathbf{y}_i)}{|\mathbf{y}_j - \mathbf{y}_i|^3},$$

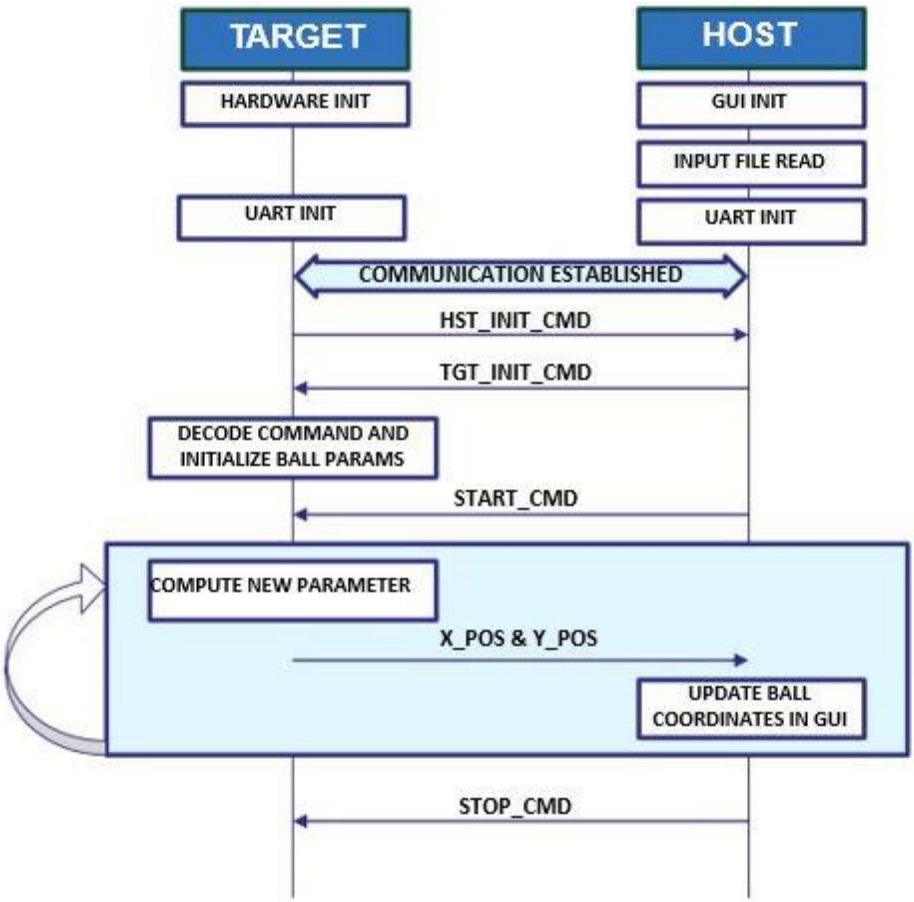
The algorithm employed in the computation is as below:-
snippet of the code-

```
for (p1 = 0; p1 < no_of_bodies; p1++)
{
    pax = pay =0.0;
          for (p2 = 0; p2 < ip.no_of_bodies; p2++)          //determine acceleration
               {
                    if (p1 != p2)  {
                                   DX_DISTANCE = bp[p2].x_pos - bp[p1].x_pos;
                                   DY_DISTANCE = bp[p2].y_pos - bp[p1].y_pos;
                                   ACTUAL_DISTANCE = sqrt(dx*dx + dy*dy);
                                   ACCELARATION = g * bp[p2].mass / (D*D);
                                   ACCEL_X_AXIS += DX_DISTANCE * A / D;
                                   ACCEL_Y_AXIS += DY_DISTANCE * A / D;
                                       }
                                     }
               VELOCITY_X_AXIS += ACCEL_X_AXIS * period1;        // VELOCITY
               VELOCITY_Y_AXIS += ACCEL_Y_AXIS * period1;

               DISTANCE_X_AXIS += VELOCITY_X_AXIS * period1; // Distance
               DISTANCE_Y_AXIS += VELOCITY_Y_AXIS * period1;

}
```

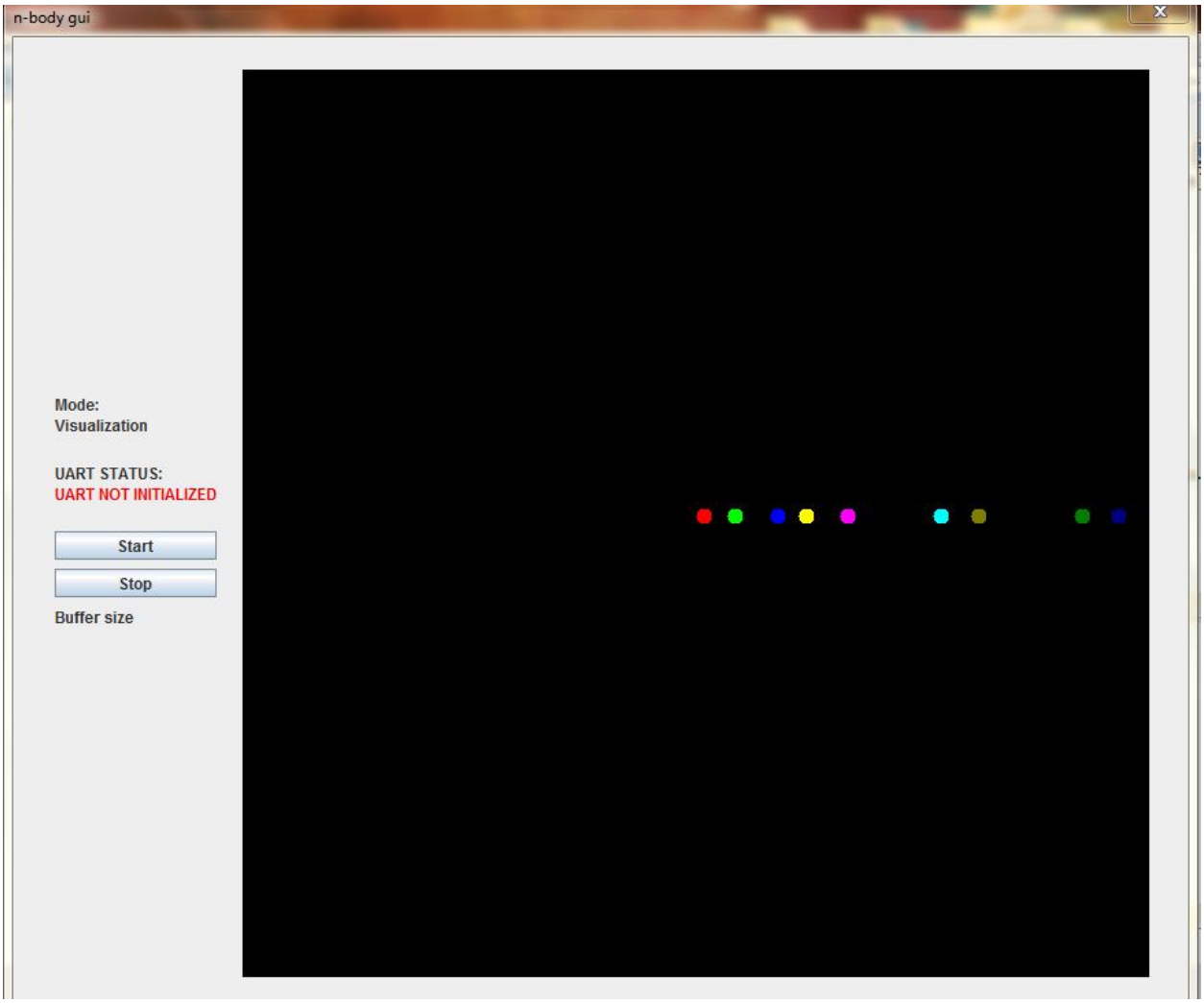
Main tasks to be done in Target are configuration of UART, Timer and computation algorithm implementation. Whereas in GUI, serial communication, use of java swing for different objects like balls and lines; real time update of balls position based on collision or free fall on screen, and providing user friendly interface with different facility like start, stop, resume and quit option for visualization in real time.

- ***Configuration of Timer and UART:***

  - We used two timers, one for generating delay time period of 40 ms(25 frames in one second) which can be used to calculate all ball positions and speed each 40ms.
  - Using Timer 0A, ball position and speed calculation is done each 40 ms. After this x and y positions are calculated and sent to JAVA GUI where the actual position of planets are converted to the positions on the GUI.
  - For communicating in real time between Host and Target we selected **UART protocol** mainly because of its simplicity and features like Pin multiplexing, auto flow control.



The GUI developed in JAVA is :-



Lab Question:-

1. What problems have you encountered in this lab?

Answer:- We faced several problems during this project as listed below,

- Finding a Serial Communication DLL
- Java JRE compatibility with Serial Communication DLL
- Making the design scalable for dynamic number of balls

2. What is the maximum number of bodies you can simulate?

Answer :- We were able to simulate up to 30 balls at a decent speed, while we were able to simulate for 50 bodies without much concern. Computation mode for 9 bodies for 100years took about 1minute and 45 seconds.

3. How do you optimize your algorithm to shorten the calculation time?

Answer :- Following steps we did to improve calculation time:

- Improving UART communication:
  By applying some changes in computation mode for sending data, we could improve upon calculation time. As only after calculating final position for balls only we are sending details to GUI for displaying it in text file.
- Less number of Branch:
  Employing the good coding practices can help in reducing the computation time. Like the main drawback of our project that we discovered at later stage is that we employed the computation logic in ISR which should have been in the main program.
- Ball computation algorithm
  The algorithm for the computation should use as minimum variable and as simple as possible. This will help in reducing the computation time drastically.

4. How do you maintain precision of the numbers for accuracy?

Answer :- The precision of the results in governed by the Euler's law. By employing the as small value of time step as possible. We can obtain the best possible results. As stated above our computation formula use differential time component for distance calculation, so the differential component of time should be as minimum as possible.

5. What is the Average percentage CPU utilization?

Answer :- The graph for the utilization according to the number of bodies is as below :-