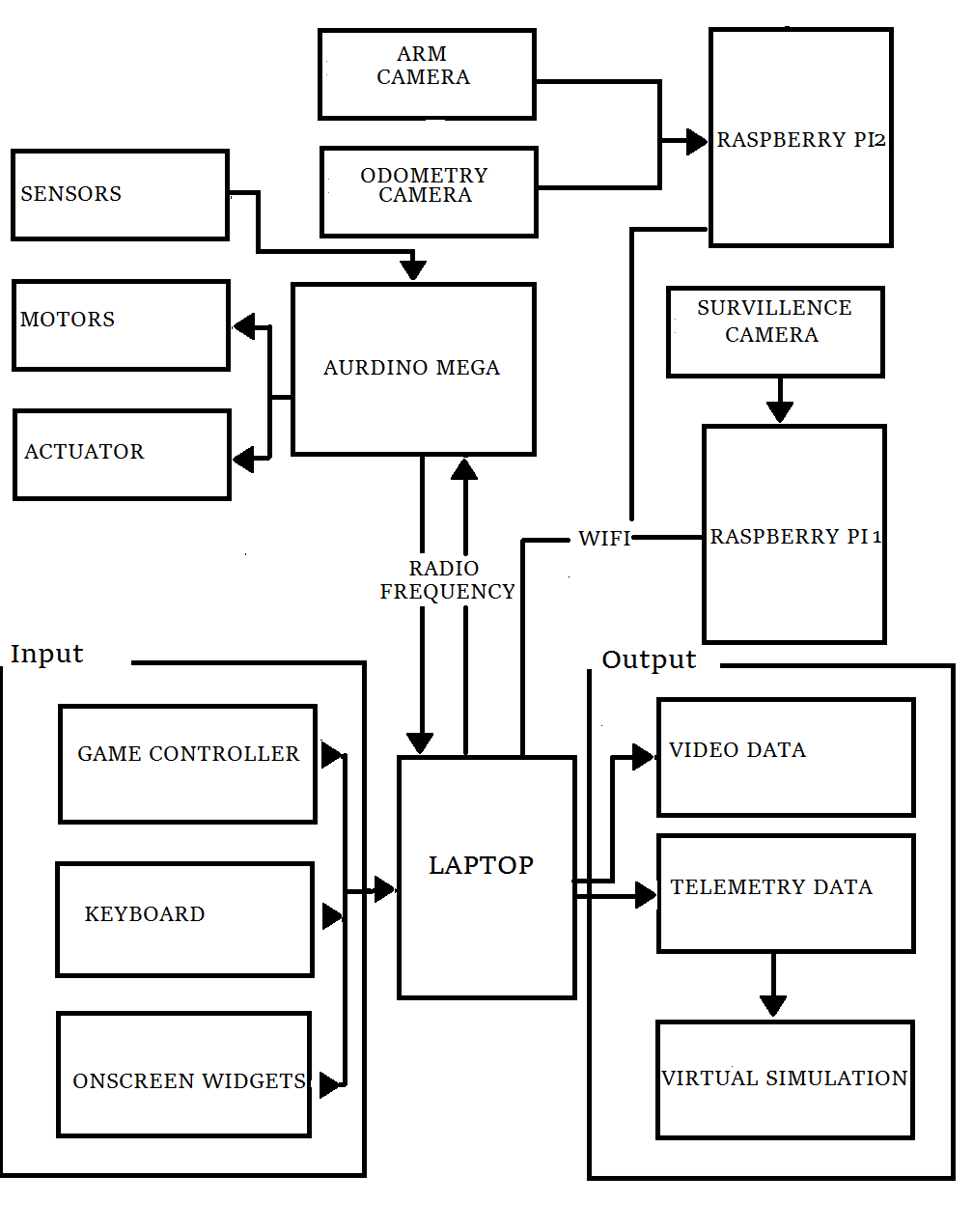
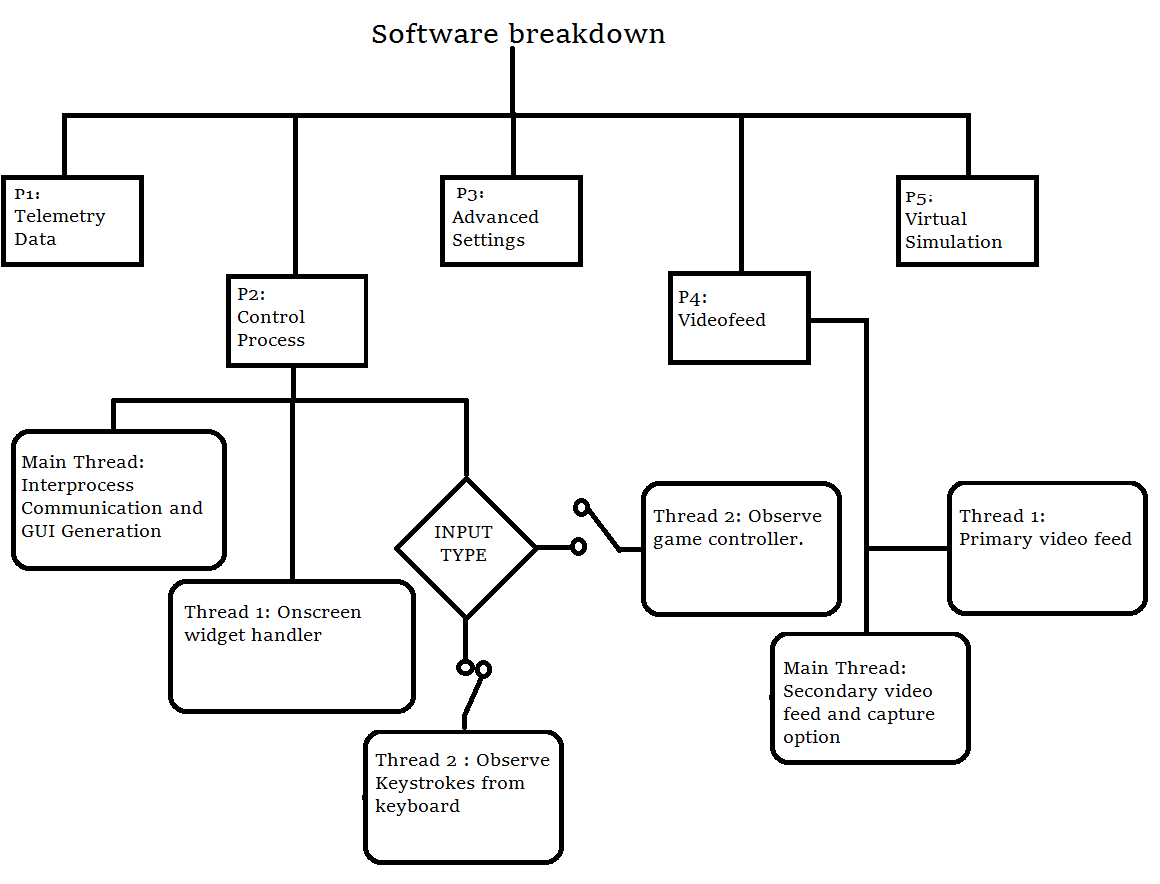
**Architechural diagram:**

* The software is designed keeping in mind it effectively utillies the various modularity concepts and parllel processing power that mordern computers utillies today.
* The objective while designing the software was that the software can give very effective and flexible control over the rover.
* The software takes the input to control the rover from the keyboard, game controller or the onscreen widgets. The provision to choose the input device is provided in the graphical user interface itself.
* The commands given by the operator through the software will be sent across the onboard arduino mega by a 300mhz rf module in form serial data.
* The software will be receiving the video feed from the onboard raspberry pi attached to a standard survillence camera via wifi.
* The visual odometry data from the second raspberry pi will also be sent using wifi. The provision will be made that the same wifi band can be utilised for the video feed and visiual odometry data. This will be done by selective usage of the band between the raspberry pi s i.e. the wifi band will be allocated according to the need for eg. the visual odomery data will be sent using wifi during the navigation task as the video feed will not be required during this time.
* The various sensor data will be received by the onboard arduino and will be sent to the Ground control Station using the 300mhz RF module . The software will receive this sensor data in form serial data.
* The serial data recived in form of a string will be processed either to be displayed systematically on the Graphical User Interface(GUI) or will be send across to other processes to construct various virtual simulation or to record feedback for motors and actuators.

**Software breakdown**

****To implement parallel and systematic execution of commands, the various functionalities required to control and moniter the rover are divided into several processes. The larger processes are further brocken down into threads.

* **Process 1( Telemetry Data):** This process is responsible to receive the data from the arduino mega via RF which is recieved in form of a string. The string is further broken down to into list of floating point values. These values are futher communicated to other processes for more processing.
* **Process 2( Control Process):** The process is assigned the task of sending the various commands delivered by the operator via his/her choice of input device (Keyboard/Game Controller/Onscreen widgets).The provison to make the choice of input device is given in the GUI itself.
  + **Thread 1(Onscreen widgets):** This thread will observe the changes made in the onscreen widgets and accordingly will send the commands to the rover. The thread will also update the widget on the basis of the commands given by the other input devices.
  + **Thread 2(Keystrokes and controller):** The commands given by the keyboard or controller on the basis of choice will be sent to the on-board controller via RF using serial library in python.
  + **Main Thread(Interprocess communication):** This thread will be utilised to do the inter process communication.
* **Process 3(Advanced Settings):** This process will help to determine the least count of every command i.e. for eg. it will decide that how long will the motor run on each command .
* **Process 4( Video Feed):** The process will receive the video feed and will make provision to display the video feed in the GUI.
  + **Thread 1( Primary video feed):** This thread will be used to load the video data from the survillence camera.
  + **Main Thread( Secondary video feed and capture command**): This thread will load the video feeed from the secondary camera and also provide provision to capture images using onscreen button.
* **Process 5(Virtual Simulation):** The process will receive the required telemetry data from **process 1** and using the required data it shall create the various virtual simulations using OpenGL library in python2.

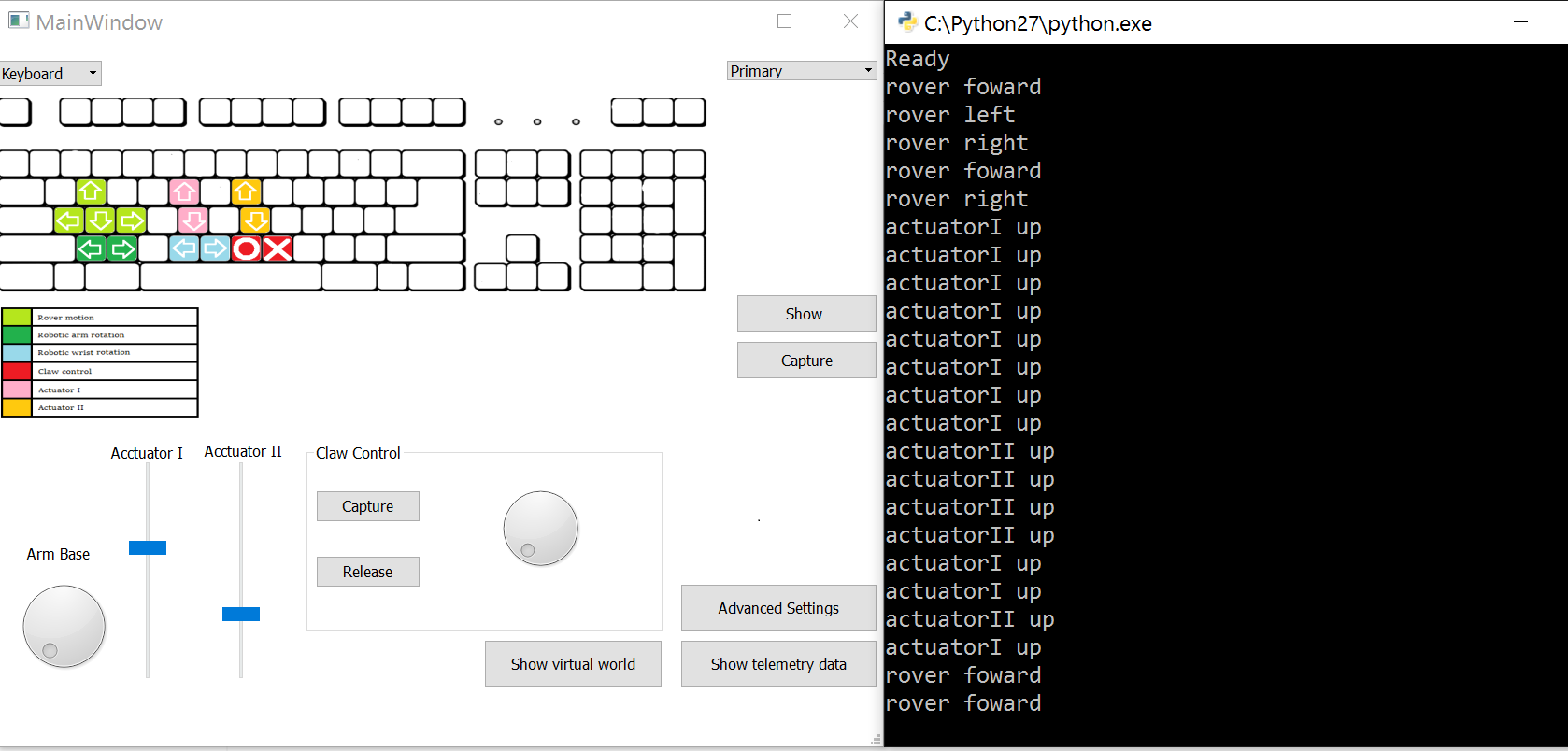
**FEATURES OF THE GROUND CONTROL SOFTWARE:**

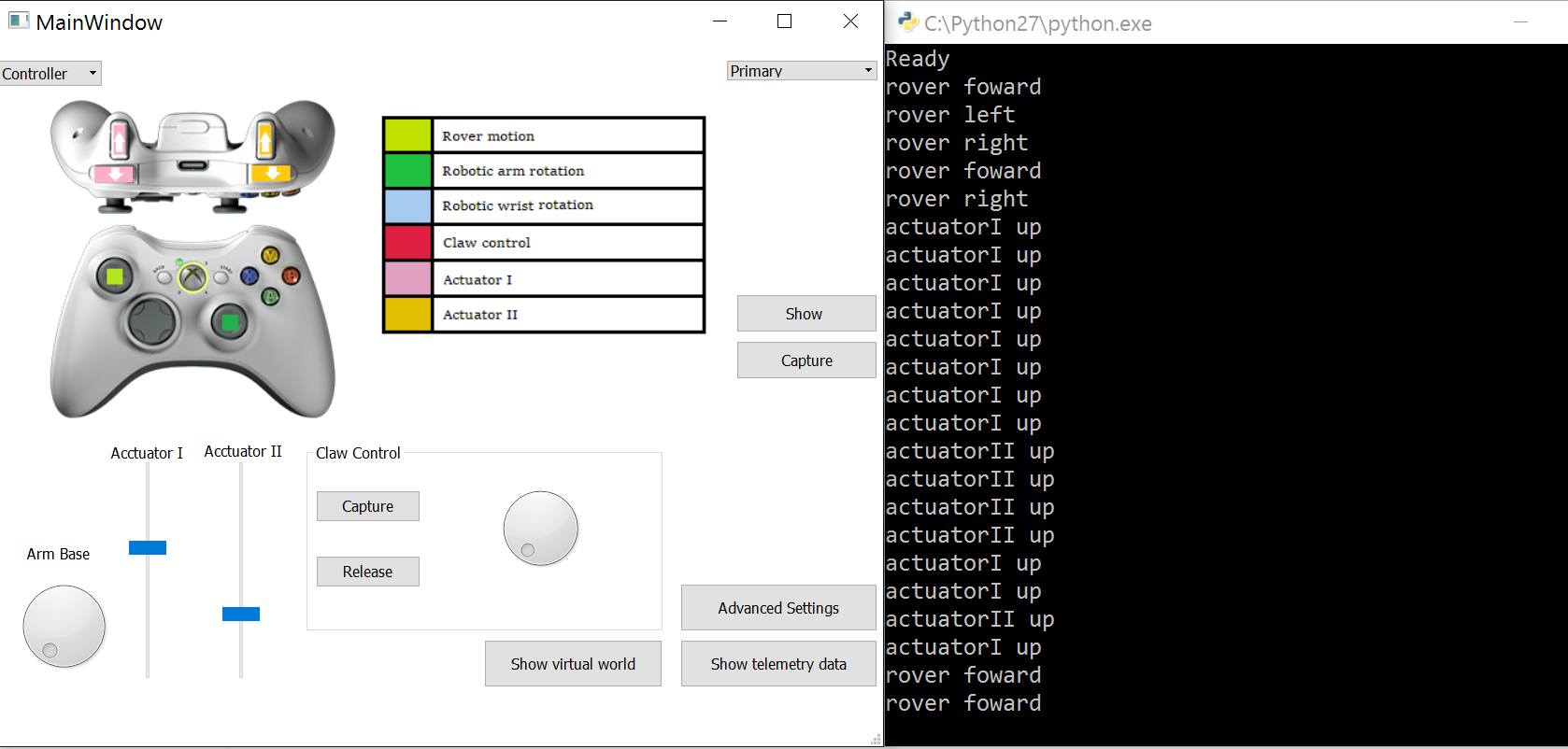
**Contol system:**

* This subsystem is responsilble for the movent of the rover that is to control the motors and also to control the robotic arm along with the movement of the claw and scoop and drill assembly.
* The system will allow you to control the speed of the rover by manipulating the least count values which can be altered in the advanced settings process.
* The effective control of the actutors and the base worm wheel motor gives the arm six degree of freedom.
* Every command is accomapnied by a calculated delay to avoid the need to put inplace anykind of flushing algorithm, thus enhancing the control over the rover.
* This is the most important process as the other process can only start when this process is in the system.
* The various hardware associated with this system are-

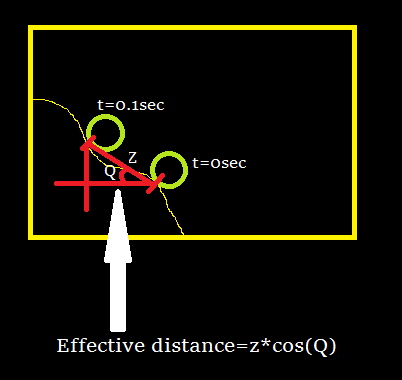
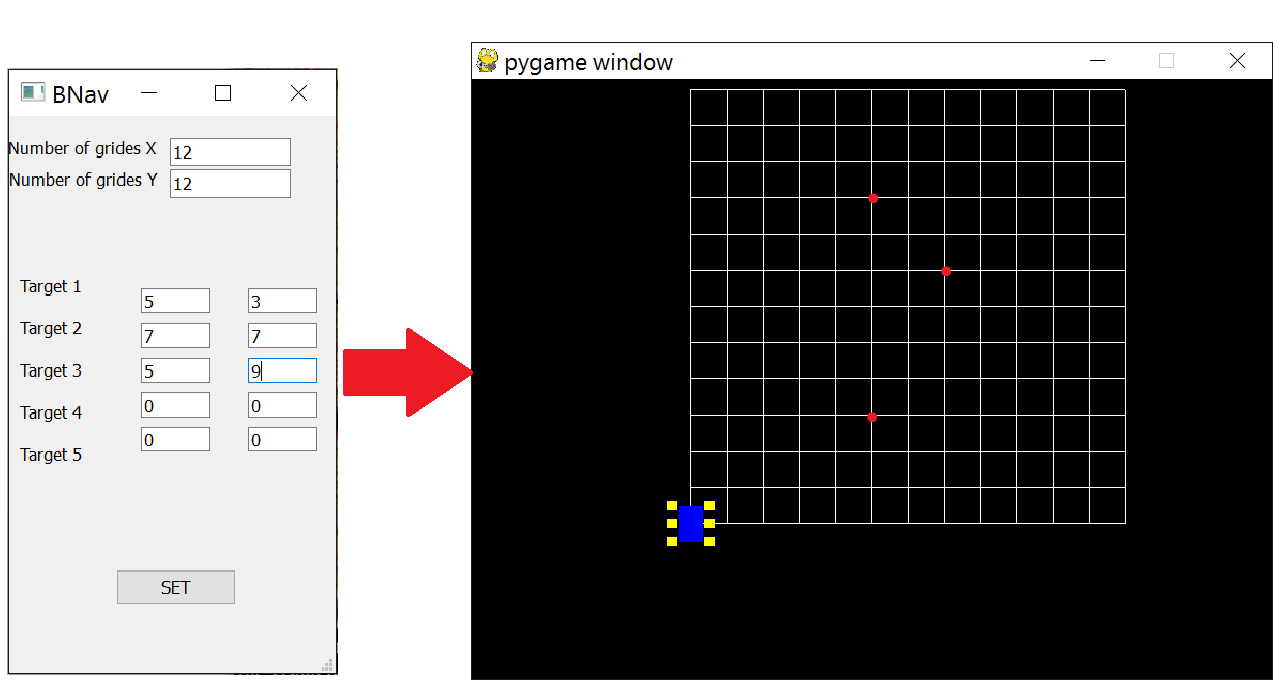
|  |  |  |  |
| --- | --- | --- | --- |
| SR NO | NAME | IMAGE | SPECIFICATION |
| 1 | LENOVO Z50 | https://images-na.ssl-images-amazon.com/images/I/513er1R8rXL._SL1500_.jpg | * WINDOWS 10 + ubuntu 15.01 * Intel Core i7 (4th Gen) 4510U / 2 GHz * 3.1 Ghz * 1TB Storage * 8GB RAM |
| 2 | Frontech JIL-1731 | http://n4.sdlcdn.com/imgs/a/2/h/Frontech-Red-Gaming-Pad-or-SDL526462419-1-ece0e.jpg | * 12 buttons * 2 analog stick * feedback |
| 3 | Intex multimedia keyboard | http://ecx.images-amazon.com/images/I/710FkxRMAqL._SL1500_.jpg | * Standard 102 keys * 9 multimedia keys * Interface: USB 2.0 |
| 4 | SMA Antennas | DAA043SA100N 433MHz 3dBi Right Angled SMA Antennas - Black + Golden (2 PCS) | * 433mhz * 3dbi * Omnidirectional * Impedence : 50 ohms |
| 5 | 3dr Radio telemetry kit | http://www.geeetech.com/images/v/Geeetech_3DRRADIO_3.jpg | * 433mhz * Receiver sensitivity to -121dBm * Transmit power up to 20dBm (100mW) * Air data rates up to 250kbps |
| 6 | Zebronics ZEB-NC1000 | Zebronics ZEB-NC1000 Notebook Cooling Pad | * USB powered * Fan Size: 180mm |

* The following is the GUI used in the control system:-

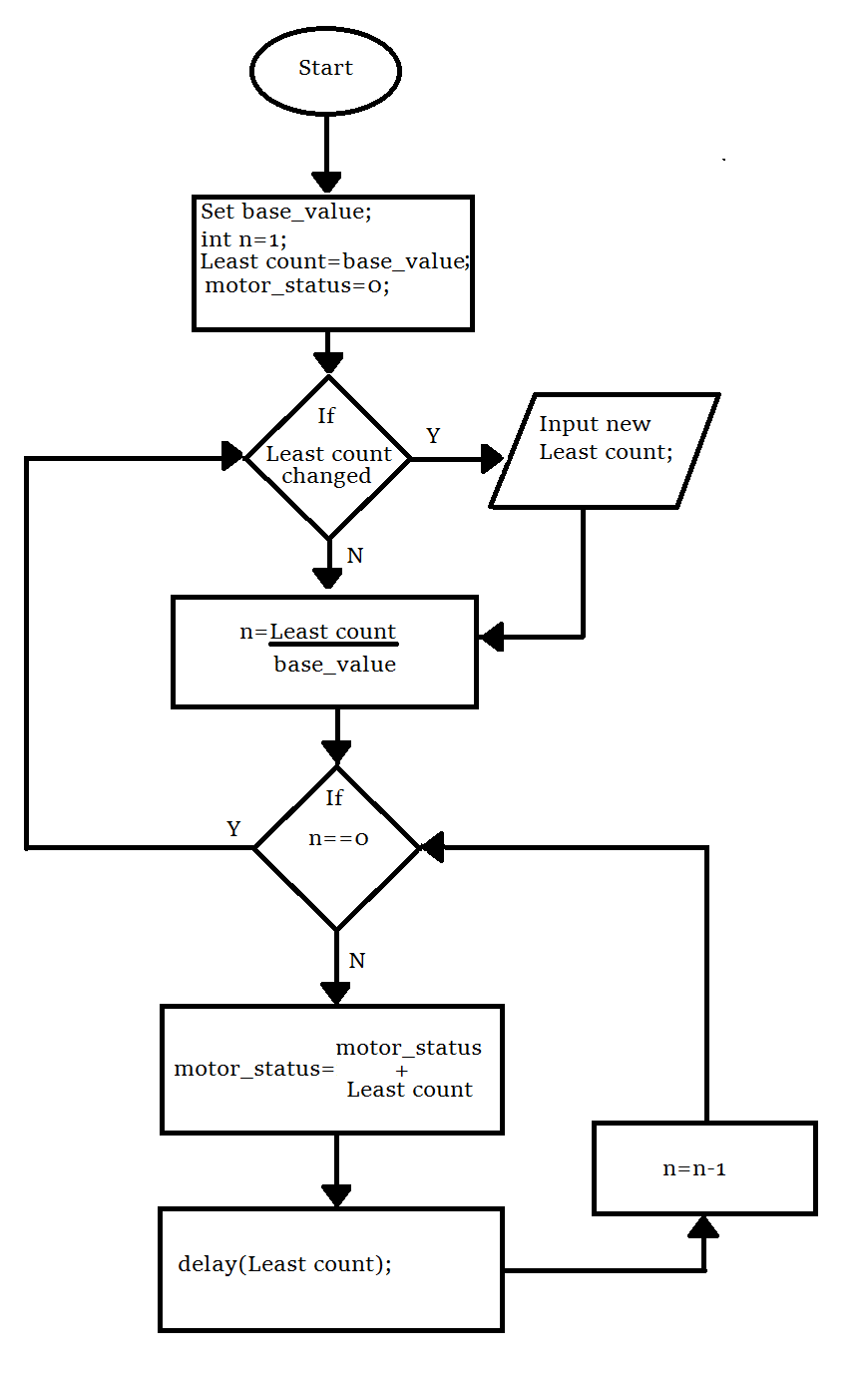


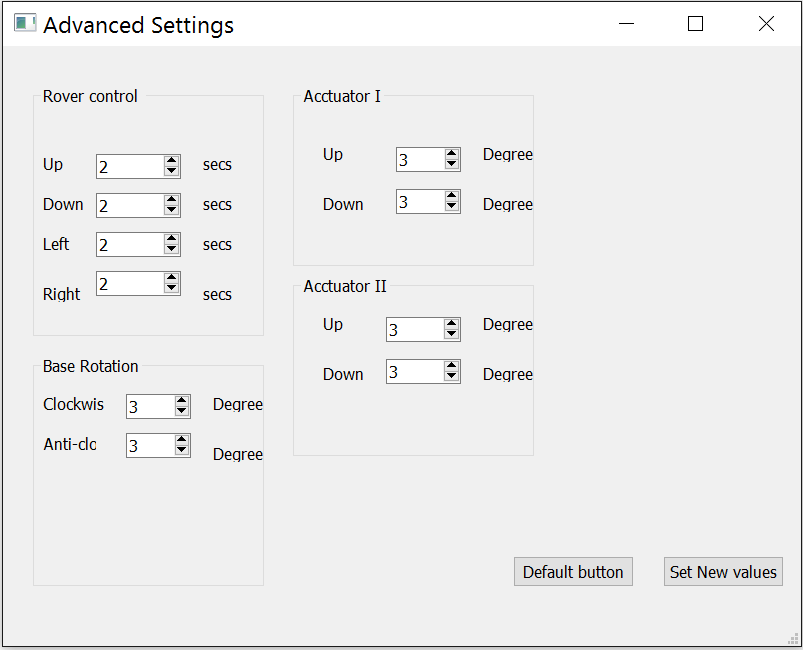


**Blind Navigation System:**

* In order to complete the navigation task the rover control system will have to contain a provision which allows us to navigate the field without using any kind of visiual data. For this reason we came up with the Blind Navigation.
* Our Rover uses visual odometry, along with Data from Inertial measurement unit which gives the gyro-meter and accelerometer data to generate a virtual simulation.
* This Virtual Simulation of the Navigation enables pilot of the rover to navigate the rover without the use of any kind of visual data.
* Visual odometry is technique which uses successive images shot from a camera attached to the vehicle to determines the speed of the vehicle and the path
* The other mars rover use various camera to prepare the simulation, whereas our rover uses just one camera and 4 IMU to draw a virtual navigation. Like other mars rover our rover may also be used to compute the safest path, i.e. to avoid obstacles and rough terrain.
* The images illustrate the algorithm used in visual odometery.
* The following is the blind navigation interface we have provided in GUI of the rover software:

**Advance setting System:**

* ****In order to effectively control the rover it is mandatory to have a provision where the LEAST COUNT of every command can be set .
* Adding the ablity to control the least count can ultimatly allow you to control the speed of the rover by giving a calculated delay between successive commands.
* The following diagram illustrates algorithm used to moderate the speed.
* According to second loop there will be certain number that will be printed onto the serial monitor.
* The on-board aurdino is programed in such a way that each time a character is encountered a motor will rotate for base\_value of time. So if n charecters are encountered on the serial monitor the motor will run for nx(base\_value) time.
* The delay is provided so that extra character are not printed on the serial monitor during the execution of previous command. Hence, this code will minimise the lag between the command given and the required action.
* The following is the provision provided within the GUI of the software to make the changes in the least command:

****