Abstract

Medical nanobots not only repair cells and tissues but also multiple nanobots can help cure various types of diseases such as cancers, infection or to remove infected cells/tissues. To automate Medical Nanobot we need program to detect it and work on it; and there may be the need for manual work to move Medical Nanobot and perform operations.

A very basic software attempt is made for Medical Nanobot using C/C++ later same methodology can be used for advanced programming of Medical Nanobot. In this paper flow diagram of medical nanobot for disease detection, removal of infected cells, tissues, repairing the cells, tissues and continuous monitoring is made including various pseudo code is demonstrated such as setting up, driving nanobots for manual and automatic, auto pan/tilt, nano gyro sensors for disease detection, camera configuration, nano servo mechanism, handling interrupts and synchronization of nanobot using C++.

Keywords

Nanobot; Nano Robot; Medical Nanobot; Nano Machines; Cell Repair; Tissue Repair; Algorithm; C++

Introduction

Medical uses of nano devices incorporate plaque evacuation and heart repair. They should relocate to a foreordained site and stay in that area to finish the task.

Multiple medical nanobots can be used collectively for medical applications to map the human body, to regulate the cardio-vascular system, for insulin regulation, for targeted drug delivery, for diagnosis of cellular pathologies and for destroying tumor cells [1].

Another necessity of nano machine is that it works autonomously, free of outside control. Physical, electrical, and compound responses can deliver a reaction yet the presentation of these stimuli victimizes the device of the benefit of autonomous task and they additionally can create a reaction in the nanobots encompassing condition.

An innovative theory in the utilization of these nano devices to battle disease that includes utilizing silicon nanomachines with a thin covering of gold and light in the close infrared range.

Light in the 700-1000 nanometer range will go through tissue with insignificant ingestion. At the point when this close infrared light strikes this specific sort of nanomedibot, the device gets hot because of the wavering of the metal's electrons in response to the light. Utilizing MRI to definitely put the nanomedibots in the dangerous district, the light makes the devices warmth to 131 degrees Fahrenheit which wrecks the destructive cells yet doesn't harm encompassing tissues.

Likewise with respect to disease treatment, ribonucleic corrosive obstruction is a technique that assaults tumors on a hereditary level. Nanobots weighed down with meddling RNA that deactivates the protein creation of the growth and murders the danger would connect themselves to the tumor and convey the deadly hereditary material.In addition of expelling plaque from blood vessel dividers; they could likewise be utilized to discover arteriolar weakness.

Nanobots may likewise be utilized to distinguish particular chemicals or poisons and could give early cautioning of organ disappointment or tissue dismissal. Additionally they can be used to take biometric estimations, they might be utilized to screen the general soundness of a person.

These nano devices may discover application in an assortment of mechanical applications. Research is continuous into utilizing them in the oil business.

In addition, current research is examining their application in nano photonics to create light more effectively. PC circuits might be delivered by these small devices. They could make circuits on a very smaller scale than current drawing systems and would take into consideration to manufacture of extremely small processors and chips [2-4].

One of the major advantage of nanobots it can be considered as a way of delivering differentiated stem cells to various positions in the body. Stem cell research has been a...
huge increase in regenerative medicine. Nanobots help to enhance its impact on medicine in the near future by providing an effective way of delivering them [5][Figure 1][6].

Flow Diagram of Medical Nanobot

A. Creating nanobot base class

Pseudocode for simple nanobot base class Example I

Below is the example code to create a nanobot's base class [7].

```cpp
#include "Libraryfile.h" // Library File for Nanobot
#include "Nanobot.h" // Includes all functions, recursion loops for Nanobot
static const UINT32 LEFT_Nanobot_PORT = 1; // Left Navigation for Nanobot
static const UINT32 RIGHT_Nanobot_PORT = 2; // Right Navigation for Nanobot
static const UINT32 JOYSTICK_PORT = 1; // Manual movement if automation is failed

void Initialize(void) {
    CreateRobotDrive(LEFT_Nanobot_PORT, RIGHT_Nanobot_PORT);
    SetWatchdogExpiration(0.1); // Watchdog timer
}

void Autonomous(void) {
    SetWatchdogEnabled(false);
    Drive(0.5, 0.0);
    Wait(2.0);
    Drive(0.0, 0.0);
}

void OperatorControl(void) {
    SetWatchdogEnabled(true);
    while (IsOperatorControl()) {
        WatchdogFeed();
        ArcadeDrive(JOYSTICK_PORT);
    }
}
```

B. Manual and Autonomous Pseudocodes for Nanobots

An attempt is made for basic manual execution of program for a Medical Nanobot using C/C++ language.

Basic program of nanobot using C

Below is the pseudocode in C program that demonstrates driving the Nanobot for 2 seconds forward in Autonomous and in arcade mode for the Operator Control.

```c
#include <Libraryfile.h> // This loads the Nanobot script, allowing you to use specific functions below
#include <Nanobot.h> // Includes all functions, recursion loops for Nanobot
Nanobot myNanobot; // create Nanobot object to control a Nanobot
int pos = 0; // variable to store the Nanobot position

void setup() // required in all Nanobot Software code
{
    myNanobot.attach(9); // attaches the Nanobot on pin 9 to the Nanobot object
}

void loop() // required in all Nanobot Software code
{
    for (pos = 0; pos < 180; pos += 1) // variable ‘pos’ goes from 0 degrees to 180 degrees in steps of 1 degree
    {
        myNanobot.write(pos); // tell Nanobot to go to
    }
}
```

In this code Watchdog time class has been used to make sure that the nanobot will stop operating if program does something unexpected or crashes. something unexpected or crashes
position in variable ‘pos’
14. delay(15); // waits 15ms for the Nanobot to reach the position
15. }
16. for (pos = 180; pos >= 1; pos -= 1) // variable ‘pos’ goes from 180 degrees to 0 degrees
17. { myNanobot.write(pos); // tell Nanobot to go to position in variable ‘pos’
18. delay(20); // waits 20ms at each degree
19. }
20. }

C. Nano Gyro Sensors for Nanobots
Gyro sensors are the angular velocity applied to a vibrating element, the accuracy with which angular velocity is measured differs significantly depending on element material and structural differences. Various characteristics of gyro sensor include but not limited to scale factor, temperature-frequency coefficient, compact size, shock resistance, stability, and noise characteristics etc. [9]. In this program robot drives in a straight line using gyro sensor combination with nanobotDrive class. The NanobotDrive.Drive method takes the speed and turn rate as arguments; where both vary combination with nanobotDrive class. The NanobotDrive.Drive

Pseudocode for simple camera initialization
1. if (StartCameraTask() == -1) {
2.   dprintf(LOG_ERROR, “Failed to spawn camera task; Error code %s”, GetErrorText(GetLastError()));
3. }

Pseudocode for camera configuration
1. int frameRate = 15; // valid values 0 - 30
2. int compression = 0; // valid values 0 – 100
3. ImageSize resolution = k160x120; // k160x120, k320x240, k640x480
4. ImageRotation rot = ROT_180; // ROT_0, ROT_90, ROT_180, ROT_270
5. StartCameraTask(frameRate, compression, resolution, rot); // Nano camera configuration for nanobot

Image acquisition through camera
1. double timestamp; // timestamp of image returned
2. Image * cameraImage = frcCreateImage(IMAQ_IMAGE_HSL);
3. if (!cameraImage) {
4.   printf(“error: %s”, GetErrorText(GetLastError()));
5. }
6. if (!GetImage(cameraImage, & timestamp)) {
7.   printf(“error: %s”, GetErrorText(GetLastError()));
8. }
9. // Image acquisition through nano camera for nanobot

E. Vision and Image Processing for Nanobots
Nanobot’s vision system has to make distinction between objects and in most of all cases it has to track. It is used to automate the process and object detection [11].

Pseudocode for Color Tracking for Nanobot Example I
1. TrackingThreshold tdata = GetTrackingData(BlUE, FLUORESCENT);
2. ParticleAnalysisReport par;
3. if (FindColor(IMAQ_HSL, & tdata.hue, & tdata.saturation, & tdata.luminance, & par) {
4.   printf(“color found at x = %i, y = %i “, // finding color for nanobot
5.   par.center_mass_x_normalized, par.center_mass_y_normalized);
6.   printf(“color as percent of image: %d “,
7.   par.pdfToImagePercent);
8. }

Pseudocode for Using Specified ranges Example II
1. Range hue, sat, lum;
2. hue.minValue = 140; // Hue
3. hue.maxValue = 155;
4. sat.minValue = 100; // Saturation
5. sat.maxValue = 255;
6. lum.minValue = 40; // Luminance
7. lum.maxValue = 255;
8. FindColor(IMAQ_HSL, & hue, & sat, & lum, & par); // Specifying range for nanobot
Pseudocode for Initializaiton of nano camera Example IV
1. while (IsAutonomous()) {
2.   if (FindColor(IMAQ_HSL, & greenHue, & greenSat, & greenLum, & par) && par.particleToImagePercent > MIN_PARTICLE_TO_IMAGE_PERCENT) {
3.     nanobot - > Drive(1.0, (float) par.center_mass_x_normalized);
4.   } else nanobot - > Drive(0.0, 0.0);
5.   Wait(0.05);
6. }
7. nanobot - > Drive(0.0, 0.0); // Automating nanobot

F. Nano Servo Mechanism for Nanobots
Nano Servomechanism is also required for nanobots for rotation.

Pseudocode for Nano servo mechanism Example-I
1. Servo servo(3); // create a servo on PWM port 3 on the first module
2. float servoRange = servo.GetMaxAngle() - servo.GetMinAngle();
3. for (float angle = servo.GetMinAngle(); angle <= servoRange / 10.0) {
4.   servo.SetAngle(angle); // set servo to angle
5.   Wait(1.0); // wait 1 second
6. }
7. // Nano servo mechanism for nanobot II

Pseudocode for Nano servo mechanism Example-II
1. #include "Nano_BaeUtilities.h" // Library files for functions for nano servo mechanism
2. paniInit(); // optional parameters can adjust pan speed for nanobot
3. bool targetFound = false;
4. while (!targetFound) {
5.   panForTarget(servos, 0.0); // Start from 1 to +1 // code to identify target for nanobot
6. }
7. // Nano servo mechanism for nanobot II

F. Nano Solenoid for Nanobots
Solenoids are used as an actuator. Here solenoid can be used to remotely control if automation fails or any other malfunction. It can be used as steering of nanobot [12].
Figure 2: UML Diagram of Architecture Description

12. digIn.EnableInterrupts(); // count 5 times
13. while (counter.Get() < 5) {
14.   Wait(1.0);
15.   digOut.Set(1);
16.   Wait(1.0);
17.   digOut.Set(0);
18. }
19. if (interruptCounter == 5 && counter.Get() == 5) printf("Test passed!\n"); // free resources
20. digIn.DisableInterrupts();
21. digIn.CancelInterrupts();
22. }
23. END_TEST(TestInterruptHandler) // Interrupt Handler for Nanobot

Appendix

Example UML logic for finite state machine for nanobot
1. namespace FSM;
2. class MgaObject {
3.   String name;
4.   String position;
5. }
6. class Transition {
7.   is A MgaObject; * transition--1..*StateMachine stateMachine;
8.   1..*transition--1..*AssociationStateState associationStateState;
9. }
10. class State {
11.   is A MgaObject;
12.   0..1 dstTransition--1..*AssociationStateState associationStateState dst;
13.   1..*srcTransition--1..*AssociationStateState associationStateState src;
14. }
15. class StateMachine {
16.   is A MgaObject; * stateMachine--1..*RootFolder rootFolder;
17. }
18. class RootFolder {
19.   String name;
20.   0..1 - > * RootFolder rootFolders;
21. }
22. class AssociationStateState {};
23. namespace PrimitiveTypes;
24. class String {};
25. class Integer {};

Example UML logic Architectural description design for Nanobot
Figure 2 is the Example of UML logic Architectural description design for Nanobot.

Example UML logic code architectural description for nanobot
1. namespace ArchitecturalDescription;
2. class Architecture {
3.   0..1 - > 1..*ArchitectureDescription describedBy;
4. }
5. class ArchitectureDescription {
6.   0..1 selectedBy--1..*ArchitectureView selects;
7.   0..1 - > 1..*ArchitectureView organizedBy;
8. }
9. class ArchitectureView {
Example UML diagram for access control design for nanobot

Figure 3 is the Example of UML diagram for access control design for medical nanobot.

Example UML logic for finite state machine for nanobot

1. // UML class diagram in Umple representing a system for managing access to facilities
2. namespace access_control; //Ref_Facility_Type
3. class FacilityType {
4.    code;  
5.    description {
6.      Menu, Record, Screen
7.    }
8.    key {
9.      code
10.    }
11.  }
12. } //Functional_Area
13. class FunctionalArea {
14.    String code;
15.    0..1 parent-- * FunctionalArea child;
16.    description {
17.      Operations, Control
18.    }
19.    key {
20.      code
21.    }
22. } //Facility_Functional_Area
23. association { * FunctionalArea-- * Facility; }  
24. class Facility {
25.    Integer id; * - > 0..1 FacilityType;
26.    Integer access_count;
27.    name;
28.    description;
29.    other_details;
30.    key {
31.      id
32.    }
33. }  
34. class Role {
35.    code;
36.    role_description {
37.      Dba, ProjectMgr
38.    }
39.    key {
40.      code
41.    }
Examples of Pseudocodes in C++ is presented, UML logic for Medical nanobot is for demonstration and illustrations only and is not tested. Original code may vary and differ based on operations, model and requirements of multiple functionalities and operations.

**Results**

Basic Algorithm for Medical Nanobot with the flow chart specified in Figure 4, UML logic for Finite State Machine, UML logic Architecture Description for Nanobot, UML logic for Access control design and example of C++ is presented.

Simplified pseudocode for nanobot is demonstrated such as setting up, driving nanobots for manual and automatic, auto pan/tilt, nano gyro sensors for disease detection, camera configuration, nano servo mechanism, handling interrupts and synchronization of nanobot using C++.

**Conclusion**

Based on the above results, basic algorithm for Medical Nanobot can constructed using C++ as shown in the Figure 2.

Examples of UML for Finite State Machine, Access Control design, Access control logic, Architectural description can be used for designing software for Medical Nanobot using C++ or any other programming language.

**Discussion**

For construction of Nanobots biodegradable poly(lactide-co-glycolide) (PLGA), an FDA approved polymer, can be used to formulate the nanoparticles to form a nanobot [13].

Advances in delivering therapy, reduction of analytical tools, enhanced computational and memory capabilities and developments in inaccessible communications will be integrated allowing for the development of such nanobots [14].

In order to evaluate the effectiveness of technique, known in literature and simulation results showed the effectiveness technique in terms of achievement, that is the destruction of the cancerous cells, and velocity of destruction [15].

**Acknowledgment**

Author would like to thank Prof. Navarun Gupta, Prof. Hassan Bajwa, Prof. Linfeng Zhang and Prof. Hmurcik for their academic support. Author also thanks anonymous reviewers for their comments.
References

9. Seiko E. Gyro sensors - How they work and what's ahead. 2018