# *A Novel Decentralised Software Process Approach For Realtime Navigation Of Service Robots*

*S. Veera Ragavan,*
*School of Engineering,*
*Monash University Sunway Campus*,
Bandar Sunway, Selangor, Malaysia.
Veera.ragavan@eng.monash.edu.my

*Velappa Ganapathy,*
*.School of Engineering,*
*Monash University Sunway Campus*,
Bandar Sunway, Selangor, Malaysia.
velappa.ganapathy@eng.monash.edu.my

*Abstract*— **We present a novel decentralised architecture for navigating and controlling a Service Robot based on control of processes rather than control of discrete actions. In a world of synchronous, tightly coupled multilayered (n-tired), hierarchical systems for Service Robot applications we propose an alternate architecture that is asynchronous, loosely coupled to uncoupled, process based, and safe-fail. The Modules and Components that have been developed and tested for this asynchronous control architecture are discussed and reported in this paper. The software engineering concepts introduced make implementing the control systems more flexible so that they can be dynamically reconfigured with ease and can be upgraded or adapted in a flexible manner. The resulting architecture is simple, and can support a wide range of trade-offs that can be manipulated easily at run-time.**

*Keywords*—**GPS, GIS, Service Robots, Robot Navigation, GSM, OTA (over the air), System Architecture and Design.**

## I. Introduction

Traditionally the control of generalised Service Robots would require well-defined activities tightly coupled across different hierarchies on single platform i.e. monolithic control architecture.

In spite of an explosion of technology and methods, the Service Robots are still not complex and in their early stages of development. Many researchers specialize in one or more areas/topics, which usually involve development of algorithms. However, in order to test the competence on a real robot, a complete system is needed involving a process based approach. Many of these are required to run in parallel and need to communicate both synchronously and asynchronously. It has to also accommodate changing application requirements, incorporate new technology, interoperate in heterogeneous environments, and maintain viability in changing environments. This puts a tremendous burden on the developer if he or she has to build everything from scratch and hence a delay in "Market ready" products.

We present a novel decentralised architecture for navigation and control of Service Robots based on control of processes rather than control of discrete actions. The current approach is a loosely coupled integration of different process technologies and computational mechanisms. It is our firm contention that a well designed software architectural framework is necessary to effectively leverage microcontrollers ( read Service robots) , wireless networks (read Telematics, distributed wireless networks) and process orchestration ( read service) to address problems of complexity, scale and reliability of networked Service Robots.

This paper discusses our domain, existing architectures, component and processes execution techniques and the approach we took to integrate these to form a distributed decentralised web enabled service that is robust and safe-fail. The resulting architecture is simple, and can support a wide range of trade-offs that can be manipulated easily at run-time.

## II. The Service Robot Domain

### *A. Layered Architecture and Hybrid approaches*

Early robotic systems for single functions were designed as control systems with a clear feedback model. A sensor generates feedback, which is compared to the expected feedback which is derived from a model of the system. Any deviation is used to update the control signal so as to minimize the error over time. As complexity grew and the robots needed to perform more than one function, the perception-action loop was extended to have a planning component. This was a natural linear extension beyond traditional control towards modern day Service Robots. This resulted in a hierarchical system having an elaborate model of the world, using sensors to update this model, and to draw conclusions based on the updated model. Obviously it does not perform very well in dynamic and unpredictable environments as the sensors and real world models are usually inadequate. That the actions are not a direct consequence of perception is perhaps the reason why it is also called the sense-plan-act paradigm

Reactive approaches are often capable of autonomously exploring new regions in the environment and, as there is no fixed plan, they are generally able to respond rapidly to any changes that may occur in the operating environment. Moreover, they are more tolerant to uncertainties in sensor measurements and the errors. Robots that were running reactive behavior based systems performed very well, also in changing environments. However, the purely reactive scheme is not capable of performing complex tasks. A software architecture based on purely reactive approach is usually monolithic and requires rewriting of control software for even small changes in the task, or environment.

On the other hand deliberative navigation methods generally assume that the obstacles in the environment in which a robot moves are known in terms of their physical location and dimensions. The navigation task is then to plan a path that is both collision free and satisfies certain optimization criteria. The classical deliberative approach to navigation is based entirely on planning and on explicit symbolic models of the world exhausts the computation resources all along the way [1]. Even more, it does not seem to operate successfully in a dynamic changing world. It has difficulties in dealing with sensors' errors as well. The models it uses are not realistic; it appears that the world is too complicated to be presented completely. Whenever an attempt to create a complete model that includes all the essential knowledge needed to deal with the uncertainties and surprises of the real world, the model became enormously big and the planning too expensive in time and computer resources.

### *B. Hybrid Approaches*

A hybrid approach, combining low-level reactive behaviors with higher level deliberation and reasoning, has since then been common among researchers [2, 3]. For a long time now the hybrid systems are usually modeled as having three layers as shown in Figure 1; one deliberative, one reactive and one middle layer [4] and this approach remains vastly unchallenged.

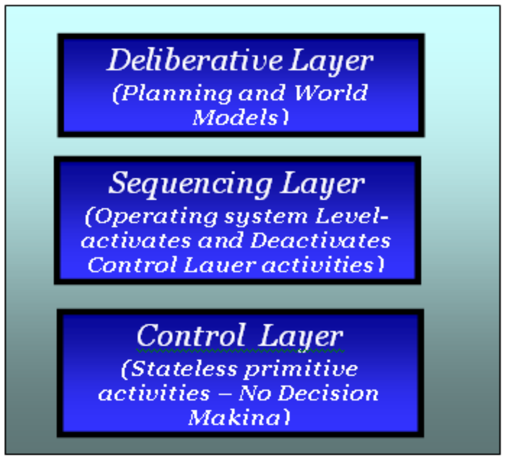

Deliberative Layer
(Planning and World Models)
Sequencing Layer
(Operating system Level- activates and Deactivates Control Lauer activities)
Control Layer
(Stateless primitive activities – No Decision Makina)

Figure 1. Three Layer ATLANTIS architecture [4]

The Lowest Layer or control layer is mainly reactive with no decision making and the process computations at this layer use the least amount of CPU time and tightly coupled to the Sensor-Actuator layer.

The Middle level or grey layer bridges the gap between the two layers [3] and is usually a Rule based layer or a finite state machine deciding which set of behaviours should be running. It also acts as a supervisory layer and catches failures of the reactive layer and then executes deliberative plans. The highest level of activities like planning, world modeling is done at this level and these are the areas that need significant computing resources. The advances in distributed computing techniques and communication infrastructure are leveraged in the proposed architecture to offer a decentralized control system.

## III. DECENTRALISED HYBRID SOFTWARE APPROACH

The hybrid deliberate/reactive approach has proven very successful, practical and robust in a large number of implementations, and it appears that there is general agreement among the research community that this is the best type of architecture for Service Robots. However, some types of modules are hard to force into any particular layer. In a general framework, it is imperative that no special architecture is preferred for enforcement and a good support for builders of the hybrid deliberate/reactive architecture is important so that the framework supports parallel execution of behaviours. This is precisely where this proposed architecture scores above the other architectures.

The major problems for robotics today lie, not in the hardware but on the software side. There are plenty of well functioning and robust algorithms developed by competent researchers readily available [5]. Each new implementation would provide significant gains in the performance and capabilities but it will be lost due to non portability and reuse issues.

While the lowest layer or reactive layer has to be embedded on the robot controller due to the obvious fact that this layer requires the highest response and lowest CPU time, the Middleware layer helps us to switch from the repository of allowable robot behaviours. What was essentially an AI Rule Based Behaviour Switching now graduates to a Location Based Behavior Switching in the current architecture.

In contrast to the “earlier” or traditional approaches to software reuse, which are built on the paradigm of a set of libraries containing many small building blocks, object-oriented frameworks allow the highest common abstraction level between a number of similar systems to be captured in terms of general concepts and structures. The result is a generic design that can be instantiated for each object system constructed [19].

The Object oriented framework [11, 12, and 13] is ideally suited for capturing the elements common to a family of related systems. In this sense, the framework is essentially a large design pattern capturing the essence of one specific kind of object system. The bulk of the system functionality is captured in the framework, which is maintained as a single entity. Each software system using framework is an instantiation of that framework [19].

In a distributed system or multiprocessor, middleware allocates system resources, giving requests to the operating systems on the individual processors to implement those decisions. One of the key differences between middleware and software libraries are that middleware manages resources dynamically. In a uniprocessor, the operating system manages the resources on the processor (for example, the CPU itself, the devices, etc.) and software libraries perform computational tasks based on those allocations.

Overview of some relevant software systems (implemented architectures) can be found in [20.] Examples of existing systems are AuRA, Task Control Architecture, Saphira, Teambots, Smartsoft, Mobility, Player/stage, MIRO [20], LICA [19], ORCA [18], BERRA [16], PeLote [15] and Loosely coupled Layered architecture for Robot Autonomy CLARAty.[14].

Middleware is software infrastructure that has been used to successfully integrate and manage software for complex distributed systems [8]. Middleware is generally constructed to provide communication between application software and processes in P2P, client-Server or Publish - Subscribe models. Most middleware addresses a particular domain such as web services, RTOS etc and define simple and uniform architectures for developing applications in the domain. Standard mechanisms for defining software interfaces and functionalities encourage the development of well-defined and reusable software. The Middleware concepts introduced make implementing the control systems more flexible so that they can be dynamically reconfigured with ease and can be upgraded or adapted in a flexible manner. An appropriate Middleware would allow software components to be integrated easily and provide standard functionalities such as support for robustness and fault tolerance, which can be easily reused in most applications.

We present a novel decentralized architecture as shown in Figure 2 for navigating and controlling a service robot based on control of processes rather than control of discrete actions.

By Process we mean a system element that is independent, and can be freely deployed and versioned. This approach loosely couples the various layers into process components that are well defined entities that can be replaced or made redundant without affecting the rest of the systems. It is shown here how they can be developed and tested separately and integrated later building on the Middleware Framework to provide a systematic approach to developing software that would be easy to integrate, manage, and reuse.

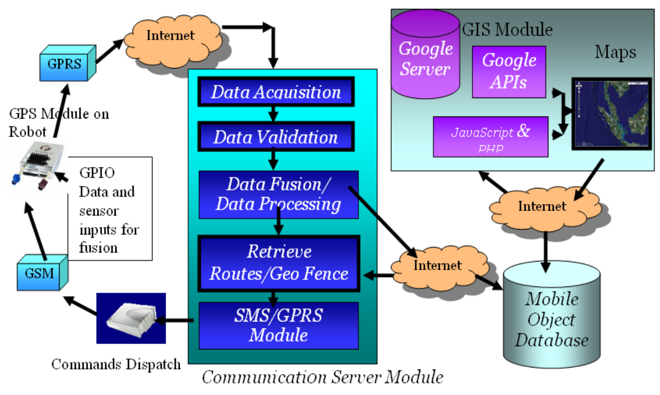


Figure 2. Decentralized System Architecture

## IV. Implementation, Integration and Experimental Test Setup.

The implementation of the distributed software engineering concepts introduced in the earlier section permits our Service Robot application to be dynamically reconfigured with ease and to be upgraded or adapted in a flexible manner using the P2P, Client Server and Producer-Consumer models.

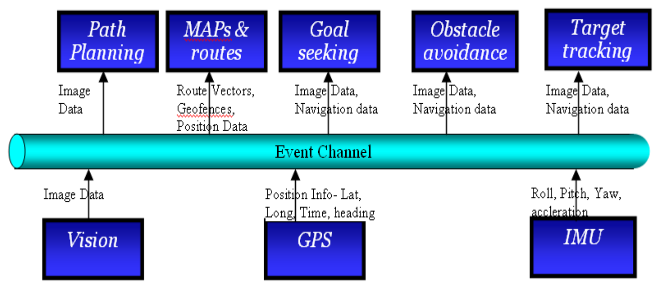

Path Planning
MAPs & routes
Goal seeking
Obstacle avoidance
Target tracking
Image Data
Route Vectors, Geofences, Position Data
Image Data, Navigation data
Image Data, Navigation data
Image Data, Navigation data
Event Channel
Image Data
Position Info- Lat, Long, Time, heading
Roll, Pitch, Yaw, acceration
Vision
GPS
IMU

Figure 3. Producer Consumer Middleware Model

The robot communicates to communication server (GPRS Data Acquisition Module, Data Validation and Command Dispatch Modules located in different physical locations through a secure web connection) in a P2P mode using GPRS / TCP/IP. We utilized Falcom StepIII Telematic modules [9] with Middleware (PFAL commands) to dynamically configure and process the sensor modules like GPS units, distance sensors and video camera. The communication between the Telematic terminal and the robot GPIO's (General Purpose Input and Outputs) is shown in the Figure 4.

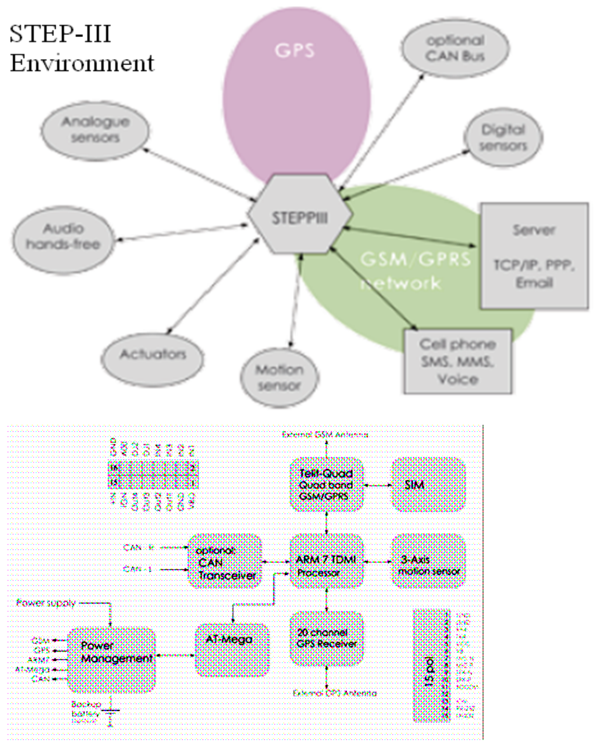


Figure 4. Telematic unit – Function Blocks in Falcom - Step III.

The Robot communicates with the sensors through the event channels in the Publish – Subscribe mode through the GPIO's. The control components are software modules that perform the tasks of path planning, goal seeking, obstacle avoidance target tracking and localisation. The sensor components consist of device drivers and hardware. The sensors used here are GPS, vision systems and IMU and there exists a loose coupling through the Middleware providing an abstract communication channel referred to in the Figure 3 as Event Channel. The sensors register with the event channel as publishers of data (e.g. camera as image data and GPS as position data) and Process components (e.g. Obstacle avoidance and target tracking) are subscribers. Subscribers get the data from whatever is available from the publishers and new publishers can be added at will.

## *A. GPRS Data Acquisition Module*

A heterogeneous asynchronous communication process spread over four process layers and two physical layers is at the core of this design process as shown in Figure 5. The Communication Server Module was developed based on Client/Server architecture to acquire the GPS data over a GPRS network using a TCP/IP connection. In regions of poor GSM coverage the module switches to SMS for command transfer. GPS devices running on GSM/GPRS SIM cards were configured as clients to stream positional information to a Test Communication Server which had to be located external to the university network due to static IP/Firewall restrictions. The units streamed data directly from GPS devices to an external communication server which performed the processes of data acquisition and validation functions before passing on the data for Data Fusion.

The balance process of the Communication Services such as data and alert processing, command and alert service responses and configuration despatches was spread over a remote system within the university campus through the web. Therefore, data was collected externally, and the client application was used to stream the bulk data to the Server for processing.

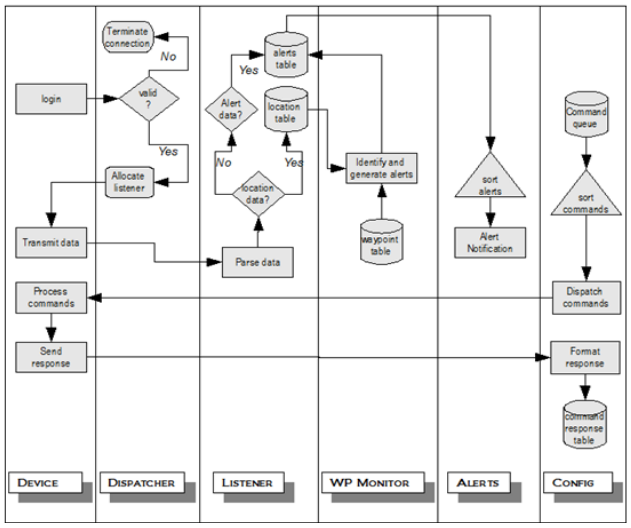
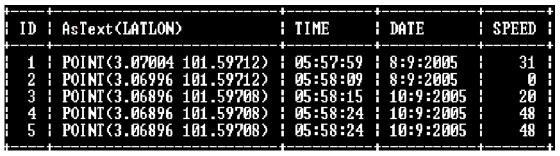


Figure 5. Communication Process flow

Data received at the server was validated prior to structuring based on the starting characters of the string ($GPRMC) and by checking whether the third element in the string represents character 'A', which implies the validity of the string. Once validated, each string was structured using Sensor Bridge components.



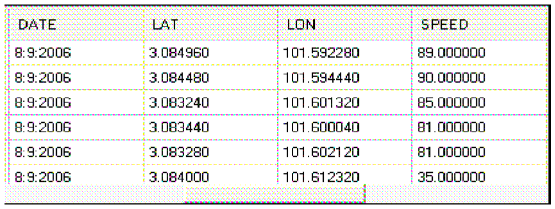
```
+----+---------------------------+-----------+------------+--------+
| ID | AsText(LATLON)            | TIME      | DATE       | SPEED  |
+----+---------------------------+-----------+------------+--------+
|  1 | POINT(3.07004 101.59712)  | 05:57:59  | 8:9:2005   |     31 |
|  2 | POINT(3.06996 101.59712)  | 05:58:09  | 8:9:2005   |      0 |
|  3 | POINT(3.06896 101.59708)  | 05:58:15  | 10:9:2005  |     20 |
|  4 | POINT(3.06896 101.59708)  | 05:58:24  | 10:9:2005  |     48 |
|  5 | POINT(3.06896 101.59708)  | 05:58:24  | 10:9:2005  |     48 |
+----+---------------------------+-----------+------------+--------+
```

Figure 6. Spatial Table containing GPS data

Structuring Data and Data fusion is done using Sensor Bridge. Sensor Bridge is a component suite developed for Visual Studio 2005. It can incorporate data from different types of sensors and actuators. It consists of a hierarchy of class structures designed to manipulate raw sensor data received. Using Sensor Bridge, data received from different GPS devices were structured into separate series tables created from Sensor Bridge Components. Thereafter, the useful information such as latitude, longitude, time, date and speed were extracted to be stored for further processing. The inertial measurement readings obtained through the GPIO of the GPS modules are also streamed along with the position info to provide for near real time location awareness.

## *B. Mobile Object Database and Database Management.*

A Mobile Object Database (MOD) system was developed using MySQL as the data repository to store the processed GPS data. MySQL is an open source database management system, noted mostly for its speed, reliability and flexibility. Furthermore, MySQL incorporates spatial extensions under the specification of the Open GIS Consortium (OGC), which is an organization that groups many other organizations that prescribes standards for GIS data processing. The Mobile Object Database for this system was developed adhering to the structure of the geometry types proposed by the OGC. Figure 6 depicts the structure of a spatial table created for a GPS device using geometry type 'POINT' to store latitude and longitude values.



| DATE | LAT | LON | SPEED |
|---|---|---|---|
| 8:9:2006 | 3.084960 | 101.592280 | 89.000000 |
| 8:9:2006 | 3.084480 | 101.594440 | 90.000000 |
| 8:9:2006 | 3.083240 | 101.601320 | 85.000000 |
| 8:9:2006 | 3.083440 | 101.600040 | 81.000000 |
| 8:9:2006 | 3.083280 | 101.602120 | 81.000000 |
| 8:9:2006 | 3.084000 | 101.612320 | 35.000000 |

Figure 7. Filtered and structured Position Information

A database connection between the MIS process and MySQL was established using a .NET connector component provided by MySQL. Data processed and structured into series tables using Sensor Bridge components were written into separate spatial tables to manage and store Geo-fences and Route patterns that have either been acquired through reactive navigation or through route patterns marked on the GIS system as shown in Figure 7, which can be re-transmitted to the Mobile robot through GPRS in order to navigate objects successively.

## *C. Transmission of Routes and Virtual Boundaries*

Geo-Fences are virtual boundaries the robot is supposed to take to reach the goal or keep away. Geo-Fences and the route vectors saved in the MOD are retrieved from the application to be transmitted back to the corresponding Mobile devices as shown in Figure 8. The Geo-Fences can also be created from the Maps like Google Earth and the boundary information can be sent to the Robot through the Telematic unit. PFAL (Device Middleware) commands were used to configure predefined routes and virtual boundaries in the GPS devices. For route configuration, a virtual boundary was created within a 30m radius for each waypoint in the route as required by the PFAL commands. Figure 8 depicts the configuration of a route to be transmitted to GPS devices.

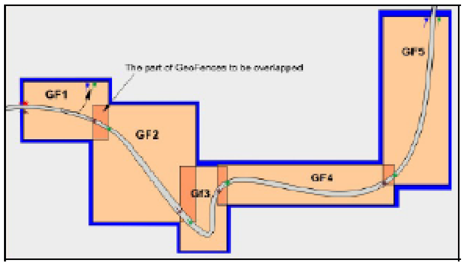
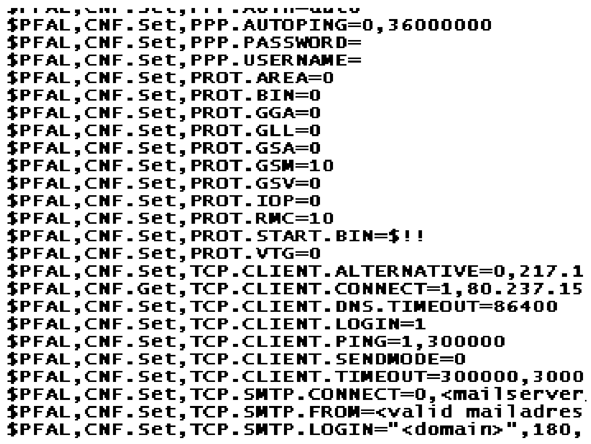


Figure 8. Route planning and Geo-fence configuration “Over The air”

### D. *Over The Air (OTA) configuration and GPIO Enabling and Disabling.*

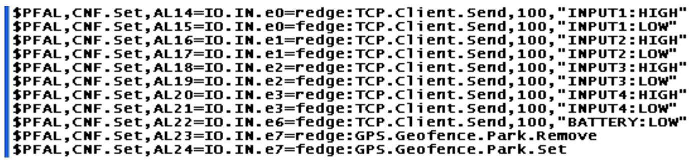
The device middleware and the GPRS link enable configuring and reconfiguring the devices as many times as required and this is effectively used to change the control program and actions remotely to use the robot for different services without having to make any major changes to the design, hardware or software. Sample configuration scripts sent over the air to the device is shown below in Figure 9.



```
$PFAL,CNF.Set,PPP.AUTOPING=0,36000000
$PFAL,CNF.Set,PPP.PASSWORD=
$PFAL,CNF.Set,PPP.USERNAME=
$PFAL,CNF.Set,PROT.AREA=0
$PFAL,CNF.Set,PROT.BIN=0
$PFAL,CNF.Set,PROT.GGA=0
$PFAL,CNF.Set,PROT.GLL=0
$PFAL,CNF.Set,PROT.GSA=0
$PFAL,CNF.Set,PROT.GSM=10
$PFAL,CNF.Set,PROT.GSV=0
$PFAL,CNF.Set,PROT.IOP=0
$PFAL,CNF.Set,PROT.RMC=10
$PFAL,CNF.Set,PROT.START.BIN=$!!
$PFAL,CNF.Set,PROT.VTG=0
$PFAL,CNF.Set,TCP.CLIENT.ALTERNATIVE=0,217.1
$PFAL,CNF.Get,TCP.CLIENT.CONNECT=1,80.237.15
$PFAL,CNF.Set,TCP.CLIENT.DNS.TIMEOUT=86400
$PFAL,CNF.Set,TCP.CLIENT.LOGIN=1
$PFAL,CNF.Set,TCP.CLIENT.PING=1,300000
$PFAL,CNF.Set,TCP.CLIENT.SENDMODE=0
$PFAL,CNF.Set,TCP.CLIENT.TIMEOUT=300000,3000
$PFAL,CNF.Set,TCP.SMTP.CONNECT=0,<mailserver
$PFAL,CNF.Set,TCP.SMTP.FROM=<valid mailadres
$PFAL,CNF.Set,TCP.SMTP.LOGIN="<domain>",180,
```

Figure 9. Middleware configuration scripts sent over GPRS and SMS

The GPIO’s data requests can be enabled or disabled “on the fly” through Over The Air Commands to optimise on the performance of the mobile device that is resource and energy starved. Video camera for instance which consumes huge amounts of battery power can be switched on ,off or at optimised rates at any time based on events or by the operator. Figure 10 shows the Middleware commands that are sent through GPRS or SMS to dynamically configure, enable or reconfigure the GPIO’s.



```
$PFAL,CNF.Set,AL14=IO.IN.e0=redge:TCP.Client.Send,100,"INPUT1:HIGH"
$PFAL,CNF.Set,AL15=IO.IN.e0=fedge:TCP.Client.Send,100,"INPUT1:LOW"
$PFAL,CNF.Set,AL16=IO.IN.e1=redge:TCP.Client.Send,100,"INPUT2:HIGH"
$PFAL,CNF.Set,AL17=IO.IN.e1=fedge:TCP.Client.Send,100,"INPUT2:LOW"
$PFAL,CNF.Set,AL18=IO.IN.e2=redge:TCP.Client.Send,100,"INPUT3:HIGH"
$PFAL,CNF.Set,AL19=IO.IN.e2=fedge:TCP.Client.Send,100,"INPUT3:LOW"
$PFAL,CNF.Set,AL20=IO.IN.e3=redge:TCP.Client.Send,100,"INPUT4:HIGH"
$PFAL,CNF.Set,AL21=IO.IN.e3=fedge:TCP.Client.Send,100,"INPUT4:LOW"
$PFAL,CNF.Set,AL22=IO.IN.e6=fedge:TCP.Client.Send,100,"BATTERY:LOW"
$PFAL,CNF.Set,AL23=IO.IN.e7=redge:GPS.Geofence.Park.Remove
$PFAL,CNF.Set,AL24=IO.IN.e7=fedge:GPS.Geofence.Park.Set
```

Figure 10. GPIO’s Enabling and Disabling Events “Over The Air”

### E. *Geographic Information System (GIS)*

A GIS system is a technique that manipulates, integrates and maps geographical information based on positional coordinates. (Latitude / Longitude). Many GIS software’s have been developed and integrated to precisely plot and display positional information, such as ArcGIS, MapPoint, Google Maps etc. Companion papers [6,7] describes the GIS system that was developed along this work as a separate Process to remotely track the robot on a web page embedding Google Maps APIs. Google APIs are freely available and accessible on the internet, and provide satellite maps as well as street maps. GIS systems incorporate several layers, each providing a different set of information to represent positional data. Google APIs provides the ability to embed many types of layers to enhance the quality of the data representation of the GIS system. This service as shown in Figure 11 is used to plan the path, Geo-fence specifications and waypoint location. Like wise options are available for the robot to be controlled and commanded over the web as a Tele-robot if need be.

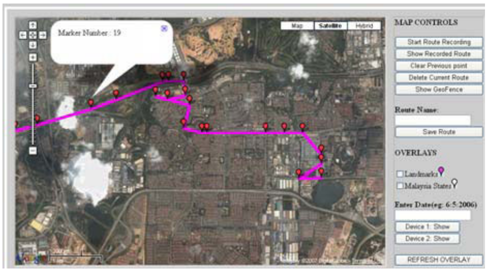


Figure 11. GIS system used for Planning.

## V. Results and Discussion

As a test example the Path Planning and Navigation System for Service Robots was successfully developed and deployed. It has also been established that Middleware can be used reliably to integrate disparate systems and processes and helps in smooth evolution of Complex Dynamical Systems.

The integration of the process modules was seamless. It has been established that dynamic heterogeneous systems can be evolved such as:- an embedded RTOS controller of the robot communicating through a GPRS mobile network, through a Windows based communication server that is a client/server application over TCP/IP, communicating the raw data acquired over a secure connection to be structured, processed, validated and fused at a remotely located server running on a different version of a Windows system across a Firewall, to be further stored in an Open Source Spatial MOD Database System running on MySQL, for further reporting and tracking of the robot movements in near Real Time over a Web based GIS Tracking system continuously and accurately on a Map and send the information stored or created such as a Geofence or Route all the way back to the robot. - all in a few hundreds of milliseconds. Lastly it should be noted that our hybrid approach has considerably evolved over time based on lessons learnt real-time and in distributed systems.

## VI. CONCLUSION

Modules and components have been developed for an asynchronous, loosely coupled to uncoupled, process based, and safe-fail system as discussed and reported in this paper. The processes have been successfully deployed across hybrid and heterogeneous platforms from dedicated RTOS processors on the robot to distributed and disparate server machines connected through the World Wide Web. It has also been established that middleware can be used reliably to integrate disparate systems and processes and helps in smooth evolution of Complex Dynamical Systems.

Having demonstrated how these strategies can be successfully implemented using the distributed networked software infrastructure such as Middleware, Webware and Hardware, a major challenge lies as future work in understanding how to make the most of it especially,

- understanding the tradeoffs between Knowledge representations that are process based reactive, deliberative or hybrid and
- how to reduce the risk by managing software related failures in network controlled systems.

## ACKNOWLEDGEMENTS

The authors wish to thank Monash University Sunway Campus for providing a seeding research grant to purchase GPS modules, sensors, software and equipment necessary for the conduct of these experiments.

## REFERENCES

[1] R. A. Brooks, Intelligence without Reasoning, IJCAI 91, Sydney, Australia, Aug.1991.

[2] Arkin, R. C. Integrating behavioral, perceptual, and world knowledge in reactive navigation, in `Robotics and Autonomous Systems, Vol. 6, pp. 105-22- year 1990

[3] Cattoni, Roldano (IRST - Istituto per la Ricerca Scientifica e Tecnologica); Di Caro, Gianni; Aste, Marco; Caprile, Bruno, Bridging the gap between planning and reactivity: a layered architecture for autonomous indoor navigation, IEEE International Conference on Intelligent Robots and Systems, v 2, 1994, p 878-885

[4] Erann Gat, Integrating reaction and planning in a heterogeneous asynchronous architecture for mobile robot navigation, ACM SIGART Bulletin, v.2 n.4, p.70-74, Aug. 1991

[5] Ibrahim, M.Y. (Sch. of Appl. Sci. &amp; Eng., Monash Univ., Clayton, Vic., Australia); Fernandes, A. Study on mobile robot navigation techniques, IEEE International Conference on Industrial Technology , 2004, pt. 1, p 230-6 Vol. 1

[6] A.U. Alahakone, S.V. Ragavan, "Geographic Information System for Path Planning and Navigation of Mobile Objects" Proceedings of Conference on Innovative Technolgoies in Intelligent Systems & Industrial Applications, Kuala Lumpur , Malaysia. 17th -19th November 2007.

[7] Veera Ragavan, V. Ganapathy, "A General Telematics Framework for Autonomous Service Robots ", IEEE on Conference Automation Science and Engineering, Scottsdale USA, Sept 2007.

[8] Service Continuity in Networked Control Using Etherware Baliga, G.; Graham, S.; Lui Sha; Kumar, P.R. Distributed Systems Online, IEEE Volume 5, Issue 9, Sept. 2004 Page(s): 2 - 2

[9] Falcom product website – http://www.falcom.de/fileadmin/downloads/documentation/STEPPIII/STEPPIII_flyer_v1.0.0_pre_web.pdf

[10] Heck, B.S.; Wills, L.M.; Vachtsevanos, G.J., Software technology for implementing reusable, distributed control systems, Control Systems Magazine, IEEE Volume 23, Issue 1, Feb. 2003 Page(s):21 - 35

[11] S. Bagchi and K. Kawamura, "An Architecture of a Distributed Object-Oriented Robotic System", Proc. IEEEIRSJ International Conference on Intelligent Robots and Systems (IROS'92), pp. 71 1-716.

[12] D. J. Miller and R. C. Lennox, "An Object-Oriented Environment for Robot System Architectures", Proceedings 1990 IEEE International Conference on Robotics & Automation, pp. 352-361, 1990.

[13] Helkne Chochon, "Object-oriented design of mobile robot control systems," 2nd ISER, Toulouse, France, June 1991, pp. 317-328.

[14] Urmson, C.; Simmons, R.; Nesnas, I., "A generic framework for robotic navigation" , Aerospace Conference, 2003. Proceedings. 2003 IEEE Volume 5, March 8-15, 2003 Page(s):5_2463 - 5_2470

[15] Kulich, M., Kout, J., Preucil, L., Mazl, R., Chudoba, J., Saarinen, J., Suomela, J., Halme, A., Driewer, F., Baier, H., Schilling, K., Ruangpayoongsak, N., Roth, H.: PeLoTe – a Heterogeneous Telematic System for Cooperative Search and Rescue Missions. Urban search and rescue: from Robocup to real world applications, in conjunction with the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) Sendai (Japan), September 28, 2004.

[16] Lindstrom, M., Oreback, A., & Christensen, H.I.: BERRA: research architecture for service robots, Robotics and Automation, 2000. Proceedings ICRA '00. IEEE International Conference on ,Volume: 4 , 24-28 April 2000 Pages:3278 - 3283 vol.4

[17] Kortenkamp, D., Bonasso, R. P. & Murphy, R., Artifcial Intelligence and Mobile Robots - Case Studies of Successful Robot Systems, AAAI Press / The MIT Press. eds (1998).

[18] Anders Oreback & Henrik I. Christensen: Evaluation of Architectures for Mobile Robotics, Autonomous Robots, Volume 14, Issue 1, January 2003, Pages 33 – 49

[19] Scott M. Lewandowski, "Frameworks for component-based client/server computing", Source ACM Computing Surveys (CSUR) archive Volume 30 , Issue 1 Pages: 3 – 27

[20] Stefan Enderle, Hans Utz, Stefan Sablatnög, S. Simon, G. K. Kraetzschmar, G. Palm, " Miró: Middleware for Autonomous Mobile Robots", IFAC Conference on Telematics Applications in Automation and Robotics – 2001.