

SOFTWARE FRAMEWORK FOR MULTI AGENT SYSTEM DEVELOPMENT IN EMBEDDED SYSTEMS

Gusthignna Wadu Sasitha Suvipul Keshan De Silva

(149152B)

Degree of Master of Science in Artificial Intelligence

Department of Computational Mathematics

University of Moratuwa

Sri Lanka

February 2017

SOFTWARE FRAMEWORK FOR MULTI AGENT SYSTEM DEVELOPMENT IN EMBEDDED SYSTEMS

Gusthignna Wadu Sasitha Suvipul Keshan De Silva

(149152B)

Thesis submitted in partial fulfilment of the requirements for the
degree of Masters of Science in Artificial Intelligence

Department of Computational Mathematics

University of Moratuwa

Sri Lanka

February 2017

Declaration

I declare that this dissertation does not incorporate, without acknowledgment, any material previously submitted for a Degree or a Diploma in any University and to the best of my knowledge and belief, it does not contain any material previously published or written by another person or myself except where due reference is made in the text. I also hereby give consent for my dissertation, if accepted, to be made available for photocopying and for interlibrary loans, and for the title and summary to be made available to outside organization.

Name of the Student

G.W.S.S. Keshan De Silva

Signature of Student

Date:

Supervised by

Professor Asoka S Karunananda

Signature of Supervisor(s)

Date:

Acknowledgements

I would like to express my gratitude to my supervisor Professor Asoka S Karunananda for the useful comments, remarks and support given through the learning process of this thesis. Also, I like to thank the panel of lectures who guided me throughout the master's program. Furthermore I would like to thank family and friends, who have supported me throughout the program to archive the goals.

Finally I would like to thank *Federico Musto* - CEO of Arduino, for his valuable advices on make the final product delivery to the global access and *Benny Estes* - Product Manager at myDevices, for his valuable guidance and inspiration.

Abstract

Multi agent systems and embedded systems both are consider as major fields in the recent research history. Both have shown the potential of developing intelligence solutions to the modern world needs. Several standards including FIPA and ACL followed by the developers and researchers when model the multi agent systems. JADE like frameworks play a major role in the field where user can design and develop multi agent system using prebuild components in the framework. Even though there are number of multi agent solutions which are run on top of the embedded systems there is no common framework to facilitate the multi agent system developments in the embedded environments. Intention is to introduce a common framework which is designed and developed to facilitate major multi agent behaviors to the researches in this field to make their work easy.

In order to archive this goal completely new software framework was design, developed and evaluated for the major embedded hardware platforms available in the market. This frame work can provide communication platform for the different type of communication channels and multi agent behaviors to the host software system in embedded platforms. Since framework is fully compatible with the FIPA standard and communication using standard communication language ACL users can easily use the framework. Solution was design in such a way that users can implement their own modules and plug in easily. In additional to that this modular design allows users to port the framework to new hardware platforms with minimal changes.

Evaluation of the framework was conducted by using several approaches. Completely new embedded solution was design and developed for the field of home garden watering in order to evaluate the features of the framework. Where different agent with different communication capabilities; demonstrate the usefulness of the framework. In addition to that several memory matrixes and performance matrixes were generated in order to evaluate the performance of the framework. Memory analysis shows that framework will consume 26.7% of the flash memory on average and can operate on minimum of 1KB static random access memory.

Contents

Chapter 1 Introduction.....	1
1.1 Prolegomena.....	1
1.2 Background and Motivation.....	2
1.3 Problem in Brief.....	2
1.4 Aims and Objectives	3
1.5 Proposed Solution	3
1.6 Outline of the Thesis	4
1.7 Summary	5
Chapter 2 MAS in Embedded Systems.....	6
2.1 Introduction	6
2.2 Starting of a new era with the JADE.....	6
2.3 MAS in Embedded Systems.....	8
2.4 Discussion	10
2.5 Summary	11
Chapter 3 Theoretical foundation.....	12
3.1 Introduction	12
3.2 Embedded Systems / Platforms.....	12
3.3 Multi Agent Systems.....	18
3.3.1 Agent Communication.....	19
3.3.2 Agent Coordination	24
3.3.3 Agent Negotiation.....	25
3.4 Summary	25
Chapter 4 Approach	26
4.1 Introduction	26
4.2 Hypothesis.....	26
4.3 Input and Output.....	26
4.4 Process.....	27
4.5 Potential Users of the System.....	27
4.6 Features	28

4.7	Summary	28
Chapter 5 Design.....		29
5.1	Introduction	29
5.2	Top Level Design.....	29
5.2.1	Master-Slave Configuration	29
5.2.2	Peer-to-Peer Configuration.....	30
5.3	Framework Components	31
5.3.1	Platform Core	31
5.3.2	Framework Core.....	31
5.3.3	Agent Container.....	31
5.3.4	Message Dispatcher.....	32
5.3.5	Communication Mapper.....	32
5.3.6	Behavioral Engine	33
5.4	Summary	34
Chapter 6 Implementation.....		35
6.1	Introduction	35
6.2	Component Implementation.....	35
6.3	Platform Core.....	35
6.4	Framework Core.....	36
6.5	Agent Container	36
6.6	Message Dispatcher.....	38
6.7	Communication Mapper.....	39
6.8	Behavioral Engine	40
6.9	Summary	41
Chapter 7 Evaluation.....		42
7.1	Introduction	42
7.2	Multi-Agent based Home Garden Monitoring System	42
7.2.1	Hardware module design	43
7.2.2	Define process and behaviors	45
7.2.3	Evaluation results discussion and conclusion.....	46
7.3	Performance Matrices	47

7.4	Memory Analysis	49
7.5	Summary	50
Chapter 8 Conclusion and Further work		51
8.1	Introduction	51
8.2	Conclusion.....	51
8.3	Limitations and Further Works	52
8.4	Summary	52
References		53
Appendices.....		55
Appendices A : Arduino Hardware Platform		55
A.1	Introduction	55
A.2	Hardware Specifications	55
Appendices B : Hardware Modules		58
B.1	Introduction	58
B.2	RF Module.....	58
B.3	RTC Module.....	59
B.4	Wi-Fi Module.....	59
B.5	Bluetooth Module.....	60
B.6	DHT Module	60
B.7	OLED Display Module	61
B.8	Ethernet Shield	61
Appendices C : Multi-Agent based Home Garden Monitoring System.....		62
C.1	Introduction	62
C.2	Hardware Module Specifications	62
C.3	Module Connection Details.....	63
Appendices D : Sample Codes		64
D.1	Introduction	64
D.2	Agent Initialization.....	64
D.3	ACL Message.....	64
D.4	CFP Process Implementation	65

List of Figures

Figure 2.1 : Embedded Agent Communication with JADE	10
Figure 3.1 : Microcontroller Packages.....	13
Figure 3.2 : PWM - Duty Cycle.....	15
Figure 3.3 : MAX232 interfacing with microcontroller	16
Figure 3.4 : Radio Frequency Receiver and Transmitter modules	17
Figure 5.1 : Framework Components	30
Figure 6.1 : Indirect Agent Configuration	37
Figure 7.1 : Component Diagram of Home Garden Monitor System.....	43
Figure 7.2 : Plant Agent Configuration.....	44
Figure 7.3 : Plant Agent	44
Figure 7.4 : Water Tank Agent Configuration.....	44
Figure 7.5 : Water Tank Agent	44
Figure 7.6 : Resource Agent Configuration.....	45

List of Tables

Table 3.1 : ACL Performatives	20
Table 3.2 : ACL Parameter List	22
Table 7.1 : Task Description Table	47
Table 7.2 : Framework Memory Consumption.....	50
Table A.1 : Arduino Hardware Specifications.....	55
Table B.1 : RF Module Specifications.....	58
Table B.2 : Wi-Fi Module Connection Details	59
Table B.3 : DHT Module Specifications	60
Table B.4 : OLED Module Features	61
Table B.5 : Arduino Ethernet Shield.....	61

Introduction

1.1 Prolegomena

Multi agent systems (also known as MAS) is one of the major and historical filed in the artificial intelligence. Where all the individual units are considered as agents and final goal will be archived by the effective communications among those agents. Since the agent based solutions are not algorithmic; there are very close to the human behaviors in some situations. Because of these features researchers has put some extra effort in this filed, in order to provide more intelligent solutions to the world. Since most of the system behaviors can map to the agent based solutions industry also intent to provide their solutions based on the multi agent technology. As a result of that there are number of major researches has be conducted in different application domains.

These researches are tried to provide intelligent solutions to software systems which mimic the real world problems/scenarios. JADE[1] like software frameworks has been developed to facilitate the development of agent based software products. Since these frame works provide most of the agent behaviors out of the box; developers can concentrate on the business logic without worrying about the agent behavior implementation. In addition to that JADE is fully compatible with the FIPA[2] standards. As a result of that there are number of multi agent based systems (in both academics and industry) were developed using the JADE framework.

Embedded systems which describe the both hardware and software solutions are embedded in to a single product, so that it can be used effectively. Modern world software solutions are more likely to be implemented in the mobile platforms; solution developers are practice to think about both software and hardware solutions. As a result of that most of the software solutions which are designed now-a-days run on dedicated hardware platforms, in contrast to the personal computers. There are some hardware platforms like Arduino[3] which makes the development to

the embedded systems easy. So that researchers as well as academics try to provide their software solutions to the limited resources hardware platforms which make those solutions more portable and connective.

1.2 Background and Motivation

Embedded solution development is one of the fields which emerge during the last five-six years. Rapid increase in the hardware platforms and the availability of large number of programming languages and interfaces to use with the hardware platforms enables researchers and developers to developed more embedded solutions than ever before.

Developing multi agent systems in embedded platforms become one of the major interest areas for the researchers and the hobbies alike. There are number of researches has been conducted in this area recently; including Power Controlling [4][5], IoT[6], Smart Houses[7] and many more. These vary from controlling a tiny autonomous robot, home security/control system to the complete industrial autonomous systems. Where well establish concept like multi agent systems; combined with the newly introduce and more powerful hardware platforms.

With this rapid development in hardware platforms (and IoT platforms) and the increase in interest of developing multi agent based systems; leads to need of a common software framework for multi-agent system development in embedded systems.

1.3 Problem in Brief

Even though there are number of researches has been done in this filed; there is no common software framework available to develop multi agent solutions in the embedded platforms. As a result of that; most of the researchers are try to build their system/solution from the scratch. This is a time consuming and repetitive task. In addition to that this approach will not cover or provide all the features that multi agent framework should cater. Since particular developer is only interest about his own research goals; he will only develop the features that need to archive his research goals.

1.4 Aims and Objectives

Main goal or the aim of this research will be developing a complete multi-agent software framework which facilitates the end to end development of the multi agent systems in the embedded environment. Major objectives of the research can be summarizing as follows;

- Identify the popular hardware platforms and develop a complete software framework for those identified platforms to make the multi agent based development easy.
- Provide support for the major multi agent behaviors, protocols and communication standards.
- Make the software framework available to the general public with the complete documentation, under the license of free and open source.

1.5 Proposed Solution

In order to archive the objectives define in the previous section a completely new software framework was proposed as a solution. This contains number of software modules to facilitate different type of multi-agent features and functionalities. Each software module will be responsible for specific task within the multi-agent system. For an example communication between all the agents will be handling by a one framework module while the coordination/negotiation related task will be handling by a separate framework module. This modularized solution allows framework to work more effectively in different type of hardware platforms as well as easy to evolve with the time.

In order to work with different type of hardware platforms with different type of memory levels, low memory consumption framework is proposed. So that memory footprint of the proposed solution will be minimal. As a result of that proposed solution - framework will able to perform in hardware platforms with minimal flash memory and/or SRAM is available and keep more memory available to the end users to perform their custom tasks.

In addition to that solution is design in a way that each module of the framework can be customized. As a result of that framework user can modify the software modules in the framework according to their target hardware platform.

Since the software framework will be available to the developers and researchers as a product of free and open source then can further modify, customize or improve the software modules based on the hardware platform they are working on.

1.6 Outline of the Thesis

Following sections of the document will discuss this solution with more comprehensive manner. Other sections of the document are organized as follows.

Chapter 02 (MAS in embedded systems): Discuss about the related works which has been done by the other people in are of interest. This includes both multi agent development and the embedded system development.

Chapter 03 (Theoretical foundation): Provide an in-depth of theoretical information about both multi agent systems and the embedded system platforms. Different communication methods and hardware modules which facilitate those will also discuss under the theoretical foundation.

Chapter 04 (Approach): This chapter defines the hypothesis of the research and discusses how the technology is used to solve the problem with references to the users, input, output and process.

Chapter 05 (Design) Chapter 06 (Implementation): These two chapters will discuss the analysis, design of the software framework and implementation of the framework in detail manner. Design section includes all design diagrams with the functional description of the each software module. Implementation chapter present the implementation of the software framework module by module with the help of algorithms, flowcharts and other code segments as needed.

Chapter 07 (Evaluation): This chapter provides how the evaluation was done for the software framework in order to verify that it meets the objectives. This includes the designing the evaluation method, generating evaluation matrices and other statistical analysis conducted in order to verify the outcome of the research.

Chapter 08 (Conclusion and Future Work): Through-out this chapter; predefine objectives and the results from the evaluation will be compare and contrast make the concussion about the outcomes of the research. In addition to that this chapter mentions several addition things that can be considered as a future works.

1.7 Summary

This chapter provides an overall view of the entire research by mean of Background and Motivation of the research, problem which is going to be address during the research, proposed solution for the identified problem and the main objectives of the research. In addition to that structure of the thesis also discuss in this chapter.

MAS in Embedded Systems

2.1 Introduction

There are number of researches has been conducted in the field of MAS in Embedded Systems. This chapter provides a comprehensive analysis of those researches. According to the literature survey, most of these studies try to provide smart solutions to the real world problems by implementing multi agent systems in the various hardware platforms available at the time of study.

2.2 Starting of a new era with the JADE

Until the *Fabio Bellifemine* at all[1] publish their finding in 1999, there was no common software platform to develop multi agent systems in computer systems. In this era several researches were working towards to standardize the multi agent system development. Even though there are existing solutions / build tools to develop MAS such as MOLE[8] there were some drawbacks in these systems.

According to the authors MOLE is the first mobile agent system which is developed in Java. This provides stable environments for the development and usage of mobile agents in the area of distributed applications. At the time MOLE initially released (in 1995) it has state of the art communication concepts and well organized secure architecture for the mobile agent system development. It has some great features compare to the contesters of that time, such as reduction of communication cost achieved by bringing two entities that (heavily) interact with each other to the same location and providing support for the asynchronous communication mechanisms by using asynchronous message queues, asynchronous processing of requests and event based programming.

The goal of Fabio and the group is to simplify development while ensuring standard compliance through a comprehensive set of system services and agents. Ultimately they manage to develop the JADE; which can be considered as a multi agent development framework. According to the authors JADE can be considered an agent middle-ware that implements an agent platform and a development environment. This was capable to deals with all those aspects that are not peculiar of the agent internals and that are independent of the applications, such as message transport, encoding and parsing, or agent life-cycle.

This development was considered as one of the major development in the implementation of multi agent systems in the computer systems. Most of the researches related to the multi agent systems conducted in each year, uses the JADE framework to boost up their prototyping as well as implementation process. JADE consists of software modules which hide the complexity of different communication channels between agents and the containers, as well as different types of agent behaviors which ranges from simple cyclic behaviors to more complex compound behaviors. In addition to that JADE fully compatible with the FIPA[2] standard which is one of the major standard still use to make the agent communication.

From that's onwards most of the Multi agent related developments were done using the JADE frame work. As a result of that all the multi agent based system developments were came in to a common standard and researchers able to concentrate more on their own field of interest rather can waste time on developing the MAS frameworks from scratch.

Qingquan Sun et al[7] propose a completely new smart house and house automation system which is based on multi agent technology. According to the authors implementation of their proposed solution was based on the JADE framework. During the process, belief, desire and intention (BDI) agent behavior model and a regulation policy-based multi-agent collaboration mechanism were implemented using the features/capabilities provided by the framework. In addition to that a set of metrics for multi-agent systems performance evaluation was generated using the JADE framework.

2.3 MAS in Embedded Systems

In recent history there is a special interest in research world to implement the MAS in embedded systems. Following section highlight some of the researches done in the field of MAS in embedded systems.

A.Carrasco et al[4] have done a research to improve the quality of Supervisory Control and Data Acquisition (SCADA) systems for automated surveillance. This system contains surveillance camera(s), multimedia stations, multimedia SCADA devices, SCADA bridge and supervision station. They model the surveillance system as a multi agent system and implement the solution in an embedded environment. When modelling the system both real work functionality of surveillance and features of the existing systems was taken in to the consideration by authors.

SquidBee[9] was used as the hardware platform which is developed by Libelium. Apart from coding in SquidBee is free and open source, they use this platform based on several features such as compatibility with the Arduino, in-built Xbee wireless communication module to enable communication through WiFi and availability of enough digital and analog pins. Based on this hardware platform, multi agent system was successfully implemented by the group.

Since there is no common way to implement the MAS solution; they used a separate node which is implemented using a SBC (Single Board Computer) and it will act as a heart of the all communications done by the agents. This SBC module consist with 233 MHz AMD Geode CPU with 64 MB of SDRAM. In addition to that there are number of general purpose IO (GPIO) pins with support for the I2C bus interface which makes it easier to implement the supervisor agent in that platform. In order to communicate between agents they use WiFi as a communication channel. According to the authors both SquidBee and type of SBC they used support for the WiFi communication and having in-built WiFi communication module which makes the implementation process easier.

Pilot test for this solution was done in collaboration with the Technology of Materials research group of the University of Cadiz (Spain) with different hardware platforms with limited resources. This evaluation process demonstrates that their solution was capable to perform the multi-agent task in different platforms with different resource levels.

‘Micro-grids’ which is known as eco-friendly power system; uses renewable power sources such as solar and wind power. Hak-Man Kim et al [5] have conducted a research to autonomous the control of micro-grids based using multi agent based solution. According to the authors, for efficient and economical micro-grid operation, a human operator is required as in other power systems. But it is difficult because there are some restrictions related to operation costs and privacy issues. In order to overcome this issues and restrictions, autonomous operation for micro-grids is proposed.

In this proposed solution four types of agents (including managing agents) will communicated with each other’s and control the grid in the optimum way so that grid will operate in the optimum manner. To build the multi-agent system, the functionality of agents, interactions among agents, and an effective agent protocol have been designed. This protocol was designed base on the Contract Net Protocol (CNP)[5] which is a high-level protocol for communication and control in a distributed systems.

Even though each agent can be implemented in the embedded system; they use the Agent-based Architecture of Distributed Information Processing Systems (ADIPS)[10] framework to implement the system in distributed PC environment. Finally the intelligent multi-agent system for micro-grid operation based on the proposed scheme was tested to show the functionality and feasibility on a distributed environment.

Internet of Things which is an emerging trend in the research world, which connect embedded systems via the internet. Where each IoT enabled device has its own processing power and storage space (which will be comparable low) and communication module to establish the connection to the Internet. Since each connected device is willing to communicate with other connected devices, share data/command/information among other connected devices and plane-execute some goals collectively; IoT can be considered as an extension of the classical multi-agent systems in embedded platforms. Since there is no central device or unit to control the individual devices, connected devices in the IoT will more or less work as agents in the multi-agent system. Inherent nature of the IoT make it’s a proper playground for Multi agent systems. So that there are number of researches [6] has been done in the field of IoT which are based on the multi agent technology.

Madakam et al.[6] done a comprehensive review on number of research proposed solutions which uses IoT enabled solutions to build systems that provide replacement or extensions to the real-world systems. Domains of this research vary from the simple home automation to managing power grids which are distributed all around the world. But in common all of the solutions were implemented as multi-agent systems so that each device will act as an agent and execute and evaluate the actions by communicating with necessary agents.

2.4 Discussion

Even though there are number of research done in the field of multi agent systems in embedded systems; there were no common framework for develop the MAS in ES. Some researches implement software solutions only the features that they are interest according to the field of study; were it is impossible to reuse by the others. Most of the researchers implement their solutions using the JADE framework based on the PC hardware platforms and use the calculated results in the embedded platforms.

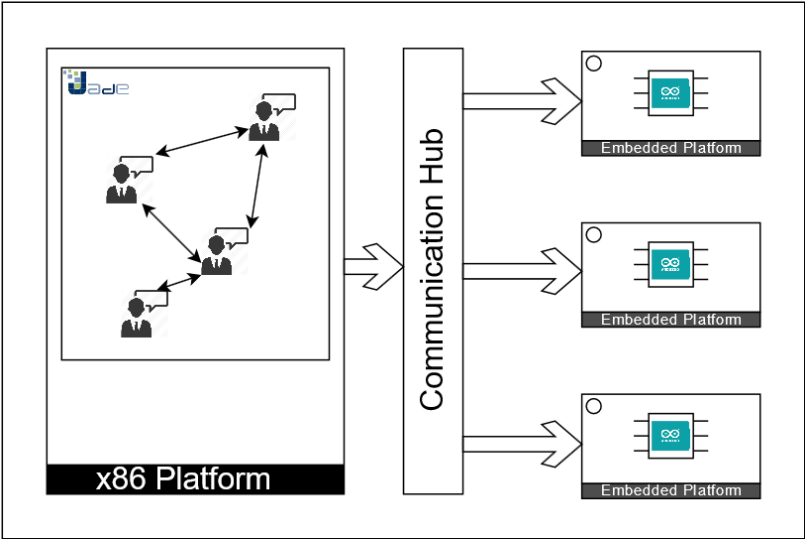


Figure 2.1 : Embedded Agent Communication with JADE

According to the current state of the field, it is clear that the need of a complete software framework for the multi agent development in the embedded systems is high priority and having such a framework will energize the development in the field of multi-agent systems in embedded environments.

2.5 Summary

This chapter analyses the others work done in the field of multi agent systems in the embedded systems. Also highlight the JADE; which is the one major framework in the MAS development in PC platforms. Next chapter is focus on the technological details related to both multi agent systems and the embedded systems.

Theoretical foundation

3.1 Introduction

This chapter consists of detail explanation about the theoretical foundation about the technologies which are used for the proposed solution. Which basically categorize in to two sections Embedded Systems and Multi agent Systems.

Embedded System section explains the different type of platforms available to implement the systems and different type of communication methods / modules available. Next section discusses the major theoretical information about multi agent systems and its behaviors.

3.2 Embedded Systems / Platforms

There are number of embedded systems - hardware platforms are available with various features. Those systems have their own hardware implementation (hardware platform) and software interface. Based on the requirement and the solution; embedded system designers will select the most appropriate platform for their products/prototypes or evaluation models. This section will discuss about some major platforms and there capabilities.

Even though most of the embedded system development platforms are providing almost same functionalities there are some differences between those platforms, when we consider about their capabilities. Those differences can be results from the hardware capabilities as well as software capabilities of those platforms. Number of inputs/outputs that platform can handle, types of inputs/output it can handle, programed memory, user memory, clock speed, interrupts, etc... are some of the features which differentiate one hardware platform form another. Before analyze the platform feature it is required to discuss the major terminologies and technologies which are bind with the embedded system development domain.

3.3.1 Microcontroller

Microcontroller[11] can be considered as a SoC (System on Chip) containing a processor core, memory and programmable input/output peripherals. Memory is in types of Flash, ROM and even small amount of RAM. These are used for the embedded applications, in contrast to the microprocessors which are used in personal computers. Some microcontrollers use 4-bit words and operate at lower clock rate around 4kHz, for low power consumption. Power consumption while sleeping may be lower as few Nano watts. On the other hand there are some microcontrollers perform critical roles, where they may need to act like a signal processors, with higher clock speeds and power consumptions. There are several dozen microcontrollers and vendors available such as *PIC by Microchip Technologies*, *ATmega by Atmel*, *TMS by Texas Instruments*, *Parallax Propeller*.

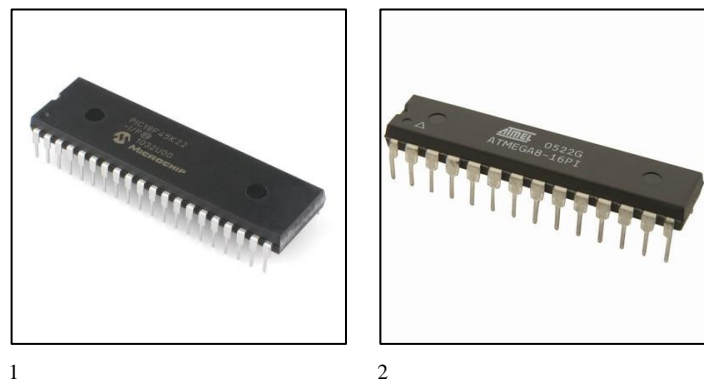


Figure 3.1 : Microcontroller Packages

3.3.2 Digital Input / Output

If a signal is either one (HIGH, 5v, 3.3v, etc.) or zero (LOW, 0v, -5v, etc.) then it is considered as a digital signal. If a hardware platform can accept these types of input signals then it is considered as digital input. In same manner if a system is capable of producing a HIGH or LOW output then it is considered as a digital output. In hardware platforms there are separate PINs which can only accepts digital input and separate set of pins which can output digital outputs. In

¹ PIC microcontroller : <http://www.alpha-crucis.com/2049-2539-thickbox/picaxe-40x2-microcontroller-40-pin.jpg>

² Atmega microcontroller : <https://wolfpaulus.com/wp-content/uploads/2012/11/atmega8.jpeg>

contrast some platforms have PINs which can operate as either digital input pin or as a digital output pin based on the software configurations.

3.3.3 Analog Input / Output

In contrast to the digital input/output; analog input/output is capable to handle the signals where signal value is between HIGH (5v, 3.3v, etc.) and LOW (0v, -5v, etc.) with predefined resolution. Because of that analog input PINs can accept and voltage value between HIGH and LOW and map the value in to 8bit or 16bit value. (Ex: analog input which can vary from 5v to 0v can be map to an 8bit value 0 to 255) In same manner some platforms has analog output pins which can produce analog voltage based on the software definition. This feature of the platform will be inherited from the microcontroller capabilities it has or based on the other peripherals which support the platform.

3.3.4 PWM Output

PWM (Pulse-width modulation)[12] is a technique which is used to generate analog like output from the digital output PINs. This technology is used; because some embedded platform does not have analog output PINs but have number of digital output pins. So that those PINs can be used generate analog outputs.

In this technique analog result is generated by digital means. A digital controller is used to generate square wave (a signal which is on and off) and this on-off pattern can simulate the voltages in between on and off by changing the portion of the time; signal spend in the ON status versus the time signal spend in the OFF status. The duration of ON time is known as pulse width. Different output level can be archived by changing the pulse width. If the pulse width time is high then the output voltage value will be high and vice versa.

In following figure green line represent the time period which is inverse of the PWM frequency. This frequency will be deferred from one hardware platform to another platform. Duty cycle which represent the time portion that signal is ON within a given time period. Based on the value which is used for the duty cycle output voltage of the PWM output will be varying.

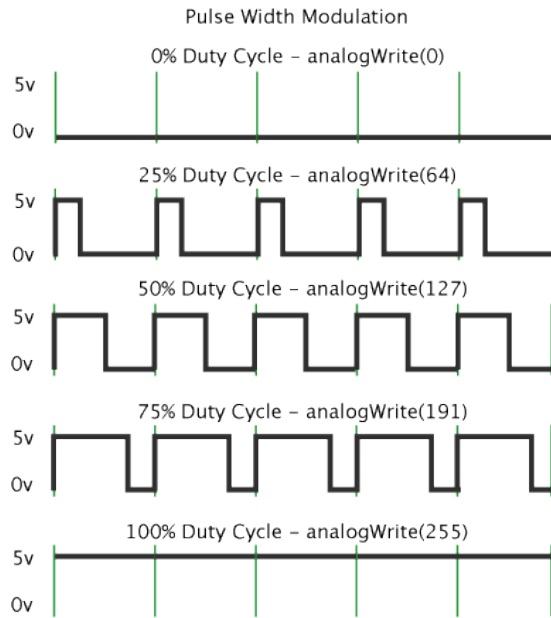


Figure 3.2 : PWM - Duty Cycle

3.3.5 Interrupts

This is the one of main/powerful feature or a concept that microcontroller has. The interrupts can be defined as a notification to the controller, by hardware or software[13]. Once this notification is received, microcontroller will immediately stop its normal routing and responds to that interrupt by executing an Interrupt Service Routing (ISR) or Interrupt Handler. After the execution of ISR routing microcontroller return back to the instruction it has jumped from. Interrupts are used in system design to eliminate the use of polling; which has higher power consumption and block all other instructions. Since controller is only response when the interrupt is occurred it is enabled the power saving for the controller and the device which uses the controller.

3.3.6 Communication Methods

Microcontrollers and other peripheral devices which can connect to the microcontroller allow the communication between microcontroller and other devices such as sensors, computers, or even other microcontrollers. This communication can be either wired or wireless. Most of the microcontrollers enabled that communication by providing separate PINs. (Rx and Tx PINs)

(a) *Serial Communication*

UART (Universal Asynchronous Receiver Transmitter) or simply serial communication is one of the basic communication interface which available in the most of the embedded system development platforms. This interface will provide a cost effective, simple and reliable communication between controllers. RS-232 standard defines the voltage levels for the UART. In this standard logic one is define as a negative (-3v to -25v) voltage while logic zero is a positive voltage (+3v to +25v). Since this voltage levels are different from the standard microcontroller operational voltage levels it is required to have a separate voltage level converters such as MAX232. In some development platforms such as Arduino; this converters are in-build. So that the can directly use by the developer without having separate components.

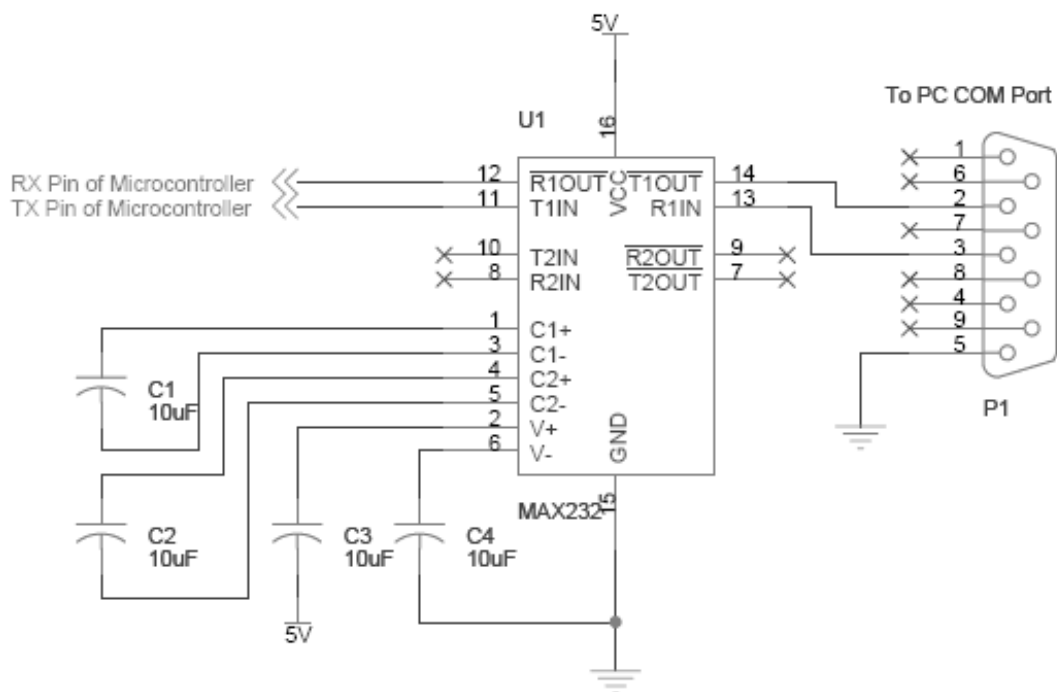


Figure 3.3 : MAX232 interfacing with microcontroller

³ Image Source : <http://www.8051projects.net/serial-communication/max232.png>

(b) *RF Communication*

Radio Frequency (RF) is a rate of oscillation in range of 3 kHz to 300 GHz; which can be used to establish a communication link between two end points. There are separate modules available which can connect to the microcontrollers and used as a communication devices. These modules are known as RF modules. There are three types of modules are available

- (i) *Transmitter Module*: This module can transmit the data as a radio frequency. (modulating) The transmission frequency and the power will be determined by the capabilities of this module.
- (ii) *Receiver Module*: This module will receive radio signals and demodulate those signals in to digital signal. Most of the receiver module can only be operate in predefine frequency only. So that it will only receive the signals which are send by its matching transmitting module.
- (iii) *Transceiver Module*: This module is design to perform both transmitter module and receiver functionalities.



Figure 3.4 : Radio Frequency Receiver and Transmitter modules

3.3 Multi Agent Systems

The term agent system can be expressed in different manner. One major way of defining agent is; a computer system which is capable of taking independent actions on behalf of its user by Wooldridge[21]. Russell and Norving [22] define; agent is anything that perceives its environment through sensors and act upon the environment through actuators. One major consideration about agent is; agent will interact with the environments. Because of that environment which agent get interact with also take in to the consideration when design / developing the agents.

Environments can be categorized to following categories according to their characteristics. Fully observable environment is an environment where sensors of the agent will detect all aspect of the environment vs. partially observable environments. Environment is partially observable because of the incorrect sensors or due to the noise in the environment. If the next state of the environment is completely determine by the current state of the environment and the action which is taken by the agent, then the environment is categorize as deterministic environment otherwise stochastic.

There are environments where agent's experience is divided into independent atomic episode, each consists of agent perceiving and performing a single action known as episodic environments. In contrast sequential environments current decision could affect future decisions. In addition to that environment can change while the agent deliberates, known as dynamic environments. But in the same cases environments will not update the surrounding if agent action is in-progress. Finally an environment can be either discrete or continues base on the relationship between time and the states in the environment, and percept and action.

Multi agent systems are design and developed to mimic the real world systems. Most of the real world systems are dynamic, complex and interconnected. A system is considered as complex if the system contains number of components interact with each other's; and each component has its own preferences or objectives and global properties are emerged from the interaction between those components.

In summary, multi agent system can be defined as a loosely coupled network of software agents that interact to solve problems that are beyond the individual capacities or knowledge of each

problem solver. There are two major types of Multi agent systems based on how each agent interacts with other agents.

1. Cooperative Multi-Agent Systems (*CMAS*) - Consist of set of agent where those agents will work cooperatively to attain a common goal.
2. Self-Interested Multi-Agent Systems (*SMAS*) - In this environment agent are works in a competitive manner to archive their individual goals.

There are three essential features that each and every multi agent system has; known as *communication, coordination and negotiation*. Following section of this chapter will discuss each feature in detail manner.

3.3.1 Agent Communication

Regardless of the environment or agent type (Whether they working in cooperative system or competitive system) agent communication plays a major role. Since agents in a multi agent system need to pass information, requests, confirmation, etc. to other agent they need a well define communication model/standard. Agent Communication Language is used to archive this goal.

3.3.1.1 Agent Communication Language - (ACL)

Agent Communication Language[15] or ACL is based on the famous theory in psychology know as speech-act theory. This theory is an attempt to formalize the way humans uses the language to archive the everyday tasks. For example make a request, giving orders, make a promise, etc. ACL is also follows the same theory and establish some standards for the agent communications. This can be use from some form of primitive communications to the elaborated standards. Because of this standard it make easier to communicate between agents coming from various environments and sources.

3.3.1.2 ACL Message Structure

A FIPA ACL message contains a set of one or more message parameters. The number of parameters required to generate a ACL message is vary (and depend) on the situation. The only parameter that is mandatory in the ACL message is the *performative* parameter; which is used to define the communication type. There are set of predefine values for the performative; following table summarizes the major types(in alphabetic order) of the performatives available in the standard[16].

Performative	Description
Accept Proposal	The action of accepting a previously submitted (typically through a propose act) proposal to perform an action. The agent sending the acceptance informs the receiver that it intends that (at some point in the future) the receiving agent will perform the action, once the given precondition is, or becomes, true
Agree	The action of agreeing to perform some action, possibly in the future. The agent sending the agreement informs the receiver that it does intend to perform the action, but not until the given precondition is true.
Cancel	The action of one agent informing another agent that the first agent no longer has the intention that the second agent performs some action. Cancel is simply used to let an agent know that another agent no longer has a particular intention.
Call for Proposal	The action of calling for proposals to perform a given action. CFP is a general-purpose action to initiate a negotiation process by making a call for proposals to perform the given action. In normal usage, the agent responding to a CFP should answer with a proposition giving the value of the parameter in the original precondition expression
Confirm	The sender informs the receiver that a given proposition is true, where the receiver is known to be uncertain about the proposition.
Disconfirm	The sender informs the receiver that a given proposition is false, where the receiver is known to believe, or believe it likely that, the proposition

	is true.
Failure	The action of telling another agent that an action was attempted but the attempt failed
Inform	The sender informs the receiver that a given proposition is true.
Not Understood	The sender of the not-understood communicative act received a communicative act that it did not understand. There may be several reasons for this: the agent may not have been designed to process a certain act or class of acts, or it may have been expecting a different message.
Propose	Propose is a general-purpose act to make a proposal or respond to an existing proposal during a negotiation process by proposing to perform a given action subject to certain conditions being true.
Reject Proposal	Reject-proposal is a general-purpose rejection to a previously submitted proposal. The agent sending the rejection informs the receiver that it has no intention that the recipient performs the given action under the given preconditions.
Request	The sender is requesting the receiver to perform some action. The content of the message is a description of the action to be performed, in some language the receiver understands. The action can be any action the receiver is capable of performing.
Subscribe	The subscribe act the agent receiving the subscribe will inform the sender of the value of the reference and will continue to send further informs if the object denoted by the description changes.

Table 3.1 : ACL Performatives

However it is expected that most ACL messages will also contains the sender, receiver and content parameters. Apart from that there are set of optional parameters available in the ACL message standards. Following list of parameters can be identifying in the ACL message structure according to the abstract parameter message payload identified in the -FIPA00001[15].

Parameter	Category	Description
performative	Type of communicative acts	Denotes the type of the communicative act of the ACL message
sender	Participant in communication	Denotes the identity of the sender of the message, that is, the name of the agent of the communicative act.
receiver		Denotes the identity of the intended recipients of the message.
reply-to		This parameter indicates that subsequent messages in this conversation thread are to be directed to the agent named in the reply-to parameter, instead of to the agent named in the sender parameter.
content	Content of message	Denotes the content of the message; equivalently denotes the object of the action. The meaning of the content of any ACL message is intended to be interpreted by the receiver of the message. This is particularly relevant for instance when referring to referential expressions, whose interpretation might be different for the sender and the receiver.
language	Description of Content	Denotes the language in which the content parameter is expressed.
encoding		Denotes the specific encoding of the content language expression.
ontology		Denotes the ontology(s) used to give a meaning to the symbols in the content expression

protocol	Control of conversation	Denotes the interaction protocol that the sending agent is employing with this ACL message.
conversation-id		Introduces an expression (a conversation identifier) which is used to identify the ongoing sequence of communicative acts that together form a conversation.
reply-with		Introduces an expression that will be used by the responding agent to identify this message.
in-reply-to		Denotes an expression that references an earlier action to which this message is a reply.
reply-by		Denotes a time and/or date expression which indicates the latest time by which the sending agent would like to receive a reply.

Table 3.2 : ACL Parameter List

3.3.2 Agent Coordination

Agents in a multi agent system need to be coordinative (or collaborative) in order to prevent chaos, satisfy global constraints, explore distinctive expertise and synchronize individual agent behaviors. Agents will use both organizational coordination and contracting as techniques to archive the coordination within the multi agent system.

3.3.2.1 Organizational coordination

In this approach agents are placed into certain organizational (hierarchical manner) structure along with certain communication patterns (can be either master-slave or peer to peer relationship). During the multi agent planning number of agents will attempt to construct a global problem resolution plan while individual plans are to avoid scenarios that would conflict with the plans of others. In this process agent need to follow the pre-defined organizational hierarchy and communicate according to the define communication patter. As a result of that this coordination does not scale well and may involve bottlenecks and central points of failure.

3.3.2.2 Contracting

In this approach agents will establish contract relationships with other agents using Contract-Net Protocol (CNP). In this process a single agent can be act as a manager and contractor. This process can explain as below.

An agent will advertise the job to other agents. (Job can be any operation task, information request, etc.) Other agents will submit their bids. Bid value indicate that how well those agents are suited for the given job. The manager (manager may be the agent who advertise the job) uses certain criterion to select one agent out of the all bidders. The selected agent will be the contractor after signing the contract. After words contractor (agent) may subcontract its tasks to other agents as well.

Even this approach is sophisticated and highly scalable; the trust can be a major issue within the contracting and bidding.

3.3.3 Agent Negotiation

Agent negotiation plays major role in the multi agent systems. This will facilitate the agents bargain between other agents to archive their goals.

More specifically; contracting agents will leads to bargaining (negotiating) between the manager and the bidders. Then that communication process will become a negotiation process. For example, an agent can use a strategy of constraint relaxation to submit bids that become increasingly more attractive to the manager.

Negotiation process can either be cooperative or competitive based on the situation. In either case agent can be success or fail in the negotiation process.

3.4 Summary

This chapter provides an in-depth description of the technological background and theoretical background of both multi agent systems and embedded systems. Next chapter explains the Approach of the research with key components of the approach.

Approach

4.1 Introduction

This chapter contains the approach of designing the common framework for the multi agent systems in embedded platforms to solve the problem which is discussed during the previous chapters. Input, output, process, users and the features are discussed in a detailed manner within this chapter.

Following solution was proposed based on the available technologies and methodologies discussed in the previous chapter.

4.2 Hypothesis

Availability of a common framework for multi agent systems in embedded platforms will improve the development process of multi agent systems in embedded environments.

4.3 Input and Output

One or more multi agent software components which were developed using the common framework will be the *input* to the framework and input to the system it-self. Since all the communication channels were routed through the framework all the incoming messages from those systems can also be considered as the *input* to the system.

Framework will manipulate the given input (messages) in order to produce appropriate multi agent system features based on the inputs to the framework. These features will be transferred back to the host software component (which is run on top of the framework), so that it can be considered as the *output* of the system.

Since the proposed solution is a framework; user program which is developed based on the framework can also be considered as both input and output of the system. Where user program

will use appropriate software components to interact with the framework and framework will trigger back the user program code segments as output.

4.4 Process

Based on the different type of inputs from the different type of hardware platforms, produce the required output for the host system (based on the hardware platform) can be consider as the process of the proposed framework. Host software system (Multi-agent system) will be considered as an external input for this process while multi-agent based behaviors and actions will be produce by the process. Framework will contain several software modules to work with different type of hardware platforms as well as different type of inputs generated by the host systems.

In addition to that this process allows framework to be users to extend the functionality of the framework for the different type of platforms as needed.

This process allows users to easily implement major functionalities of a multi-agent system such as identify the available agents, initiate the communication between agents, request agent details from the facilitator, coordination and negotiation among the agents, etc. The designing of the framework is discussed in the design chapter in a detailed manner.

Potential Users - discuss in the next section can use the functionalities provided by the framework to develop the multi-agent system based on their requirements.

4.5 Potential Users of the System

Mainly there are two types of users can be identified as potential users for the system.

- Firstly *people who have special interest in developing multi agent systems* for the embedded platforms. They can use the framework to develop there solutions on top of the selected platform. Since framework provides all the required software components to implement the multi-agent features; developers can concentrate more on the actual system features; rather than implementing the basic features from scratch.

- Secondly, *people doing research in the field of multi agent technologies* and embedded systems can use this framework to speed-up their solution implementation as well as the evaluation. In this approach those users can rapidly prototype their designs / solution and evaluate them in order to achieve their research goals, without spending time to implement the multi-agent communication models, behaviors, etc.

4.6 Features

The common software framework for multi agent development in embedded systems has set of valuable features which makes this framework more useful to the potential users discussed in the previous section. Those features can be summarized as;

- Framework is compatible with most of the major embedded platforms.
- Easy to use when developing multi agent solutions
- Compatible with ACL messages and FIPA standards
- Inbuilt communication module for almost all communication modules
- Support for major agent behaviors out of the box
- Small program memory footprint
- Can be modified/customized and adapt to the entirely new hardware platform

4.7 Summary

This chapter discusses about the approach to design a common framework for multi agent system development in embedded platforms; followed by the complete description of the input / output of the system. Afterwards potential users of the system and the major features of the system (Framework) are discussed in detail manner.

Design

5.1 Introduction

Framework is compliance with FIPA specifications and support for the major communication modules/channels that different hardware platform provides. It provides the capabilities to the users to implement their own multi agent system on top of the framework without considering much about the underline hardware platform.

The common software framework is designed as a collection of software components which can be easily integrate together in order to provide the MAS features. User can select appropriate components based on the hardware platform, communication mechanism and multi agent features they prefer.

5.2 Top Level Design

Framework was designed in such a way that it can be used in two different configurations.

1. Master-Slave Configuration
2. Peer-to-Peer Configuration

5.2.1 Master-Slave Configuration

In this configuration one end point was considered as a master and all communication channels will be routed through this master. But this node can also be considered as an agent. So that any multi agent related behaviors can also be implemented in this node also. All the other nodes can communicate with the master to fulfill their requirement. This node can also be considered as a directory facilitator which is and major part in the JADE framework as well.

If one hardware platform/module has more capabilities (processing power, memory etc.) then that can be nominated as master. One major advantage of this configuration is that other nodes will have fewer loads compared to the master node.

5.2.2 Peer-to-Peer Configuration

This configuration considers each hardware node equally and route the communication accordingly. One major advantage of this configuration is that, distributed/decentralized behavior of the multi agents reflect properly.

Regardless of the configuration, framework consists of software components. Following sections will discuss design details for each component individually.

- Platform Core
- Framework Core
- Agent Container
- Message Dispatcher / Generator
- Communication Mapper
- Behavioral Engine

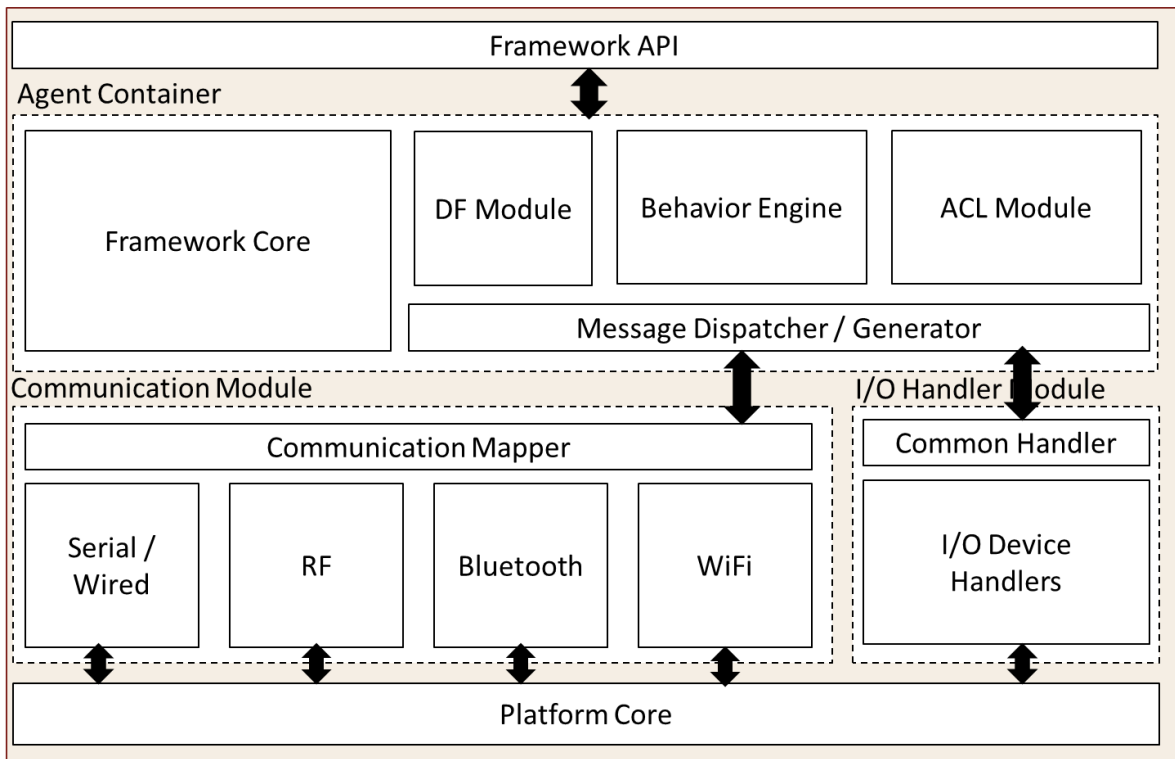


Figure 5.1 : Framework Components

5.3 Framework Components

This section will discuss the in depth design detail of the each software component which is defined in the previous section.

5.3.1 Platform Core

This software component is the lowest level of the component suit. It communicates with the underline hardware and facilitates the required functionalities for the other components. Since this module communicates with the hardware platform and used platform specific functions such as interrupts, timers, etc.; implementation of the module is different from hardware platform to platform.

This software component allows rest of the framework to generic and platform independent. In addition to that the software framework can be ported for any new hardware platforms just by implementing this component for that hardware platform. Rest of the components can be used with minimum modifications.

5.3.2 Framework Core

Basic functionality of the multi agent systems are provided by this software component. This follows the required specifications and provides the MAS functionalities. In addition to that this facilitates the message communication routing for the other components.

Also this component act as a base component of the Behavioral Engine; which contains necessary software segments for agent behaviors such as cooperative, negotiation, emerging, etc.

5.3.3 Agent Container

This software component is almost identical to the Agent Container implementation of the JADE except minor areas. Main idea of the agent contain is to provide a place to store the agents (including essential details about the agents) and make the communication mechanism unknown to the agents.

Based on the memory availability of the underline hardware platform the container size is determined by the framework itself. Eventually the number of agents that can be located within a node will be determined by the size of this container. Addition to that Framework user can define

the memory size to be used for the agent container so that framework will not use whole memory available in the hardware platform.

5.3.4 Message Dispatcher

Message dispatcher was design as a sub software component which is build inside the agent container. This facilitates the communication features for the agents it-self. All inbound and outbound messages for the agent were handled by the message dispatcher in the particular container. Message dispatcher will use a local cache/agent table to identify the exact agent to be communicated. Since the communication methods are different from one hardware platform to another this module will used the functionality provided by the communication mapper component.

This module is also responsible for generating ACL compatible messages in order to make the agent communication standardized. Also this module will interpret receiving ACL messages from other agent and trigger the appropriate software modules to perform the required tasks. Framework will support for the following ACL performative types:

- *REQUEST*
- *INFORM*
- *QUERY_IF*
- *CFP*
- *PROPOSE*
- *ACCEPT_PROPOSAL*
- *REJECT_PROPOSAL*

In addition to that Message dispatcher module playing a major role in communication routing phase; where it will reroute the messages to the receiver when the agent act as a link between other two agents(indirect agents). This functionality will be discussed under the implementation chapter.

5.3.5 Communication Mapper

Since each embedded hardware platform has its own communication mechanisms and additional communication hardware modules, it is required to have a common mapper to communicate with all these modules. This component has a common mapper to map the communication channels

and separate component for each communication method. So that framework can be configured in any hardware platform with required communication mappers according to the available communication modules. Following communication methods were supported by the framework:

- Wired Serial Communication
- Bluetooth Serial Communication
- Wi-Fi Communication
- RF (Radio Frequency) Communication

In addition to that if the user needs additional communication methods which doesn't support by the framework natively; they can implement a separate communication mapper for that communicate method and add to the framework.

5.3.6 Behavioral Engine

Behavioral engine software module was design in order to provide major agent behaviors to the framework. So that users can use those behaviors out of the box. Some behaviors are associated with communication streams; So that this module is linked to the communication module as well. In addition to the some agent behaviors are executed based on the timing information. (Ex: Ticker Behaviors) As a result of that this module will directly communicate with the platform core module to get the timing information.

Framework supports following major behaviors;

- *SimpleBehaviour* - Basic type of behaviour which execute once when the behaviour is added to the agent.
- *CyclicBehaviour* - This behaviour stays active as long as its agent is alive and will be called repeatedly after every event.
 - o *TickerBehaviour* - a cyclic behaviour which periodically executes some user-defined piece of code
- *OneShotBehaviour* - This executes ONCE and dies.... Not really that useful since the one shot may be triggered at the wrong time.
 - o *WakerBehaviour* - This executes some user code once at a specified time
 - o *ReceiverBehaviour* - This triggers when a given type of message is received

This module was design in such a way that framework user can add any number of behaviors to an agent (from pre-define set of behaviors) with different parameters. For example single agent can have two Ticker behaviors; one with 1 second cycle time and another with 10 seconds of cycle time.

5.4 Summary

This chapter discusses about the complete design details about the software framework for multi agent. Starting from the top level architecture and the configuration of the system; Followed by the list of major components in the design and in depth design details of the each component. Required materials of the designing these components and integration was covered in this chapter.

Implementation

6.1 Introduction

This section will discuss about the implementation of the Multi agent framework for the embedded systems. Since the framework should support for major embedded system development platforms / microcontrollers the implementation process was repeated for those platforms. Each software component is designed once and implement for several platforms with several languages.

Each framework component is implemented individually and integrated in to the framework. Each individual component provides unique features/capabilities which are required by the multi agent system operations. Next section of this chapter will provide the details of each component implementation and integration with the existing components. Each component can work individually there can test and evaluate individually. Testing evaluation will be discussed in the separate section.

6.2 Component Implementation

6.3 Platform Core

Platform core is the module which actually interact with the given hardware platform and execute the platform specific code segments. This module was built as the lowest level layer of the framework. Framework users will not have access to any of the functionalities provided by this module. Rather framework itself uses these functionalities from other top level modules. Following functionalities were implemented in the platform core.

- Platform specific digital I/O pin access to control the status LED; which represent some actions taken by the framework itself.
- Serial communication establishment; Framework core module will execute this routing when it need to establish a serial connection between two agents. In this scenario framework core module just act as a wrapper to the actual implementation.

- Single Wire communication establishment; this functionality will be use by the framework core when the radio frequency connection need to be established between the agents.
- Timer / Time based functionality; Platform core module will access the platform specific timers to maintain the internal timers which are used by other modules, including behavioral engine. If the hardware platform contains a RTC (Real Time Clock) then this module will communicate with the RTC to get the current time which is also used by the other framework modules. (Please referee the Appendix B - Hardware Modules for the RTC configuration)

6.4 Framework Core

This module is the base module for the all other modules including agent container, message dispatcher and behavioral engine. This module is responsible for establish the actual communication channel when the agent container requested. Required information (such as serial ID or Rx Tx pins) will be pass to this module by the agent container whenever it requires to establish a new connection. In that case framework core module will collaborate with the platform code module and establish the connected as needed. More information about the connection types and parameters are discussed in the agent container module section in the same chapter.

Interesting functionality provided by this module is the time based event triggering. Module will use platform specific timers to identify the elapse times and maintain a timer; so that it can be use by the behavioral engine to execute the cyclic behaviors.

In addition to these features framework core module has debugging system in build; in a way that it will debug all required information for debugging the framework at the runtime. This debugging feature can be enabling and disable by the framework user. Once its enables it will log the information to any pre-define stream; can be either serial output or a display panel.

6.5 Agent Container

Agent container was implemented in such a way that; it can contain the agent with all other required in formations such as communication types (how to communicate with other agents.) Since single agent can communicate with more than one connected agent these connection mediums are implemented separately for each agent.

Implementation of the agent container module stores the agent information (such as agent identifier) and the connected agent information (such as connected agent identifier and connection type) in program memory. So that framework user can define once and use whenever needs those information. For the indirect agents only the virtual connections will be created.

For example; In the following configuration (Figure 6.1) for agent M, agent S1 and agent S2 are directly connected agents while agent S3 is an indirectly connected agent.

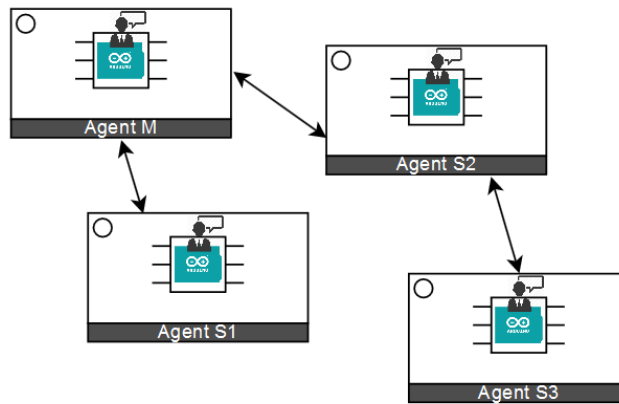


Figure 6.1 : Indirect Agent Configuration

Agent registration process was implemented in following manner. User can define the connected agents with appropriate details and register as direct connected agent or indirectly connected agent. These required information will vary based on the communication type. For example, if the connected agent is connected through the Serial communication, user needs to define the Serial line number only. But in contrast, if the communication channel is radio frequency (RF) based, then it must be registered with the appropriate TX and RX pin values. More information about the agent registration will be discussed in the evaluation chapter where the framework usage is described in a step-by-step manner.

Communication channel initialization is also implemented within the agent container module. So that whenever user registers an agent, the framework will initialize a new communication channel to communicate with that agent. Following sequence diagram explains the complete program execution sequence for the agent creation and registration.

6.6 Message Dispatcher

Message dispatcher was implemented on top of the agent container. So that agent can send and receive messages using this component. Implementation of this module contains a mapping data structure with the agent identifier and the connection type. Connection type is composite data type which contains the primary connection type (Serial, RF, etc.) as well as the additional attributes such as serial connection ID, receiver pin (RX), transmitter pin(TX) etc. Whenever message dispatcher get a send request from the agent container; it will go through the map and identify the target agent connection details. Afterwards it will send the given message to that connection.

In addition to that primary functionality; message dispatcher is responsible for the convert raw messages in to ACL messages and vice-versa. There are two common routings were implemented to archive this task. One will generate raw message content when it received the ACL message. String separators will be added to identify the each ACL component at the receivers end. Sample raw message will be in the following format - String(ACL.performative) + # + ACL.topic + # + ACL.content. In addition to that sender and receiver information will be appended to the raw message just before it will send to the communication mapper. From the receivers end message dispatcher will split this message and generate the appropriate ACL message and pass it to the agent container module.

In order to send and receive the messages through communication line; this module will seamlessly integrate with the communication mapper. So that message dispatcher can handle sending and receiving process in a common way without considering the actual communication medium. Following sequence diagram shows how the agent container module, message dispatcher module and communication mapper module work together to send/receive ACL messages in between agents.

Message dispatcher is also responsible for forwarding messages to the indirect agents. For example agent A is connected to agent B and agent B is connected to C then agent A is indirectly connected with the agent C. In such scenarios messages send by the agent A to the agent C will be identified as indirect messages by the messages dispatcher in the agent A. In this case message dispatcher will send the message to the agent B with the receiver attribute set to C. In

agent B message dispatcher was implemented in such a way that it will identify the message which is not addressed to him and forward it to the agent C. In this implementation it is guaranteed that intermediate agent will not modify any of the attributes in the ACL message. This was implemented by introducing additional routing call forward which is same as send routing except appending the sender, receiver information to the ACL message.

6.7 Communication Mapper

Communication mapper component was implemented to provide support for transmit and receive messages from/to end points. This component will identify the source of transmission media and send the message to the appropriate mappers. So that complexity will be hide from the agents. Communication mapper module contains APIs for send/receive messages with parameters to select appropriate communication media. Since all types of communications are handle by single common API user does not have to write different code for different communication methods such as serial communication, RF communication.

In addition to that communication network structure in define in this module as a separate tree like data structure. As a result of that each node knows how to route the communication / message to the desired destination. Since most of the hardware platforms supports only for peer to peer communication this network implementation is required to communicate with nodes which are not directly connected; but through a separate common node. In order to identify the each node in the network this node module uses a unique id which is assigned by the framework user to each hardware platform/end point. In addition to that framework user must have to define other connected nodes to that node when the node is initializing. (Most probably in initialization of the framework)

Serial Communication Mapper

This sub section of the communication module was implemented in order to wrap the serial communication interface and provide common API for the communication mapper. This module will initialize the serial communication and setup the TX and RX PINs in the microcontroller in order to perform the communication. Since serial communication is highly depending on the rate of operation; it was considered as a parameter. That allows frame work users to modify that parameter accordingly. Framework user has to define the TX and RX PINs that they are willing to use and the bit rate before start the communication.

RF Communication Mapper

Since radio frequency communication has some additional pre-requirements such as special formatted data, sync patterns, equal balance 0 and 1 bits, etc. Because of that standard serial communication will not be compatible with the RF modules/RF communication. So that separate software component was implemented in-order to handle those addition requirements. Since this software module hides the complexity of the communication from users framework users can establish a RF connection with minimum code with minimum parameters (ex: TX and RX PINs and bit rate)

6.8 Behavioral Engine

Behavioral engine was implemented in order to support for the major agent behaviors including both cyclic behaviors and one-shot behaviors. Following section will discuss each behavior, its implementation and the interaction with the other modules.

SimpleBehaviour

This is the simplest form of agent behavior; where the given call back will be execute at the same time the behavior is registered and only once. This behavior was implemented by the framework as a direct call back at the time of behavior registration.

TickerBehaviour

This is a cyclic behavior; means that this stays active as long as its agent is alive and will be called repeatedly after every cycle time. Ticker behavior was implemented with two parameters. Delay parameter; which will accept the time delay between each execution (cycle time) in seconds and call back routing parameter which will call by the framework for each cycle. In order to implement this behavior, behavioral engine will use the framework code module timer functionalities. When the timer expired (compared to the given delay parameter) framework core module will inform that to the behavioral engine; so that behavioral engine can execute the user given routing according to the time delay.

WakerBehaviour

This behavior is somewhat similar to the cyclic behavior; but this will not execute periodically. Rather this behavior will execute at once. Where framework user can define the exact time to

execute the behavior; then the framework will execute the call back routing given by the user. This behavior was implemented using two parameters; time parameter and call back routing parameter. Same as cyclic behavior implementation; behavioral engine depends on the framework core module timer functionality to identify if the current time is match with the given time (parameter). If so it will execute the behavioral routing.

ReceiverBehaviour

Whenever agent receives a message; agent's receiver behavior will be executed. This behavior is completely depends on the messaged receiving process rather than timing based as previous behaviors. If agent register this behavior with call back routing; framework will execute that routing each and every time that agent receives a message. In addition to that received ACL message will be passing to the call back routing as well. In order to implement this behavior; behavioral engine communicate with the message dispatcher module to identify the message receive event. On a message receive; message dispatcher module will inform the event to the behavioral engine with the received ACL message. After words behavioral engine identify the registered receiver behaviors for the agent and execute their call back routings with the ACL message.

6.9 Summary

This chapter discuss about the complete implementation process of the multi agent framework for the embedded system, including each component implementation and interaction between components in order to archive the design target. In addition to that several sequential diagrams were discussed in order to understand the component workflows for different scenarios.

Evaluation

7.1 Introduction

This chapter contains the evaluation process which is used to evaluate (and test) the multi agent framework which is discussed during this documentation. Since the main outcome of the research is a software framework for the multi agent systems in embedded platforms; unique approach was used to evaluate the system. Evaluation process consists of two major sub-processes.

One will be an embedded system which is specially design to behave as multi-agents. It was design in a way that it will contains different hardware types and communication methods between each agent. Previously discussed framework was used to implement the multi-agent features and evaluate the behavior of the system. Section 7.2 (Multi-Agent based Home Garden Monitoring System) discuss this approach in detail manner.

Second process is to measure some performance matrices of the framework in order to prove that it can be used with various hardware platforms. In addition to that several memory analyses were done to identify the memory foot-print of the framework in different platforms. Section 7.3 (Performance Matrices) contains more information about this evaluation process.

7.2 Multi-Agent based Home Garden Monitoring System

In order to evaluate and perform end to end testing of the multi-agent framework, prototype of sophisticated autonomous home garden monitoring system was developed. The concept of home garden monitoring system was highly inspired by the work done by Isern David at all[17] and Balbi Stefano at all[18]

System contains several hardware modules (represent different hardware platforms) capable of communicating with other module using different communication methods. Each hardware module is logically represent an agent and able to perform several agent behaviors when archiving there goals.

7.2.1 Hardware module design

Experimental design consists of three types of agents (hardware modules) known as ‘*Plant Agents*’, ‘*Water Tank Agents*’ and ‘*Resource Agents*’. Plant agents are directly getting the sensory inputs from the plant(s) and have actuators to control the water supply for the plant(s). Water tank agents are connected to the water tanks and get the sensory inputs from the water tanks. While resource agents are connected to the cloud, capable to extract data from different sources such as weather forecasting service, plant detail databases, etc. Following diagram shows the overall design of the system with the agent communication links.

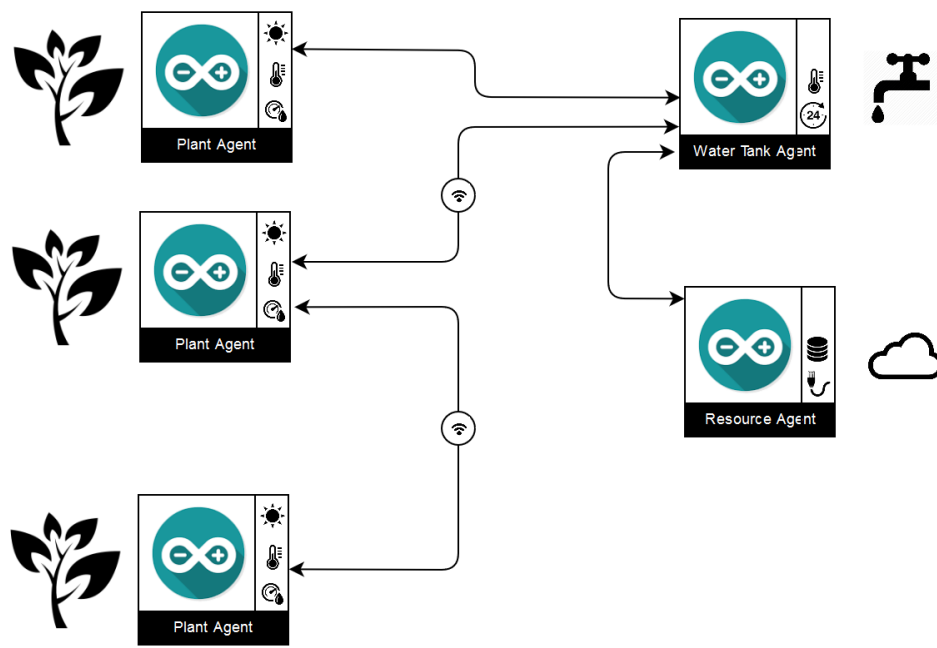


Figure 7.1 : Component Diagram of Home Garden Monitor System

Plant agent module is powered by the Arduino Nano (ATmega328) and contains set of sensors to measure the atmospheric temperature, humidity, soil temperature, soil water level and light condition. Main actuator of the agent is a motor which will eventually controller the water line to the plant. In addition to sensors it contains real time clock to keep the date time information and OLED screen to display basic information and status of the agent.

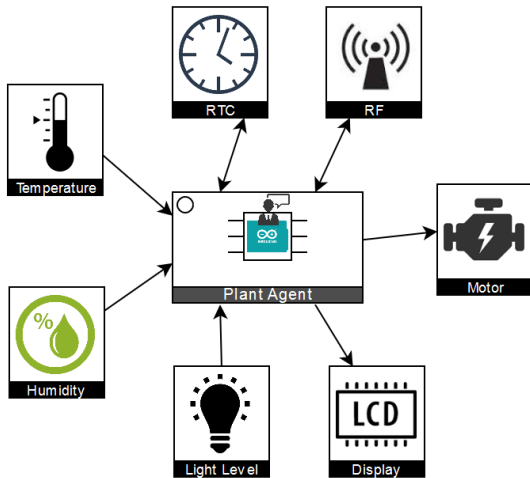


Figure 7.2 : Plant Agent Configuration

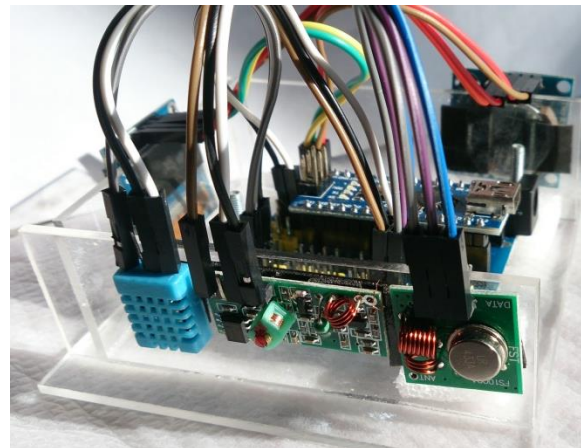


Figure 7.3 : Plant Agent

Water tank agent is powered by an Arduino Mega (ATmega1280) and contains LCD screen to display the information about the system. In addition to that sensors for the water temperature and water level are also integrated in to the water tank agent. In addition to that real time clock module was integrated in to the module to make the date/time awareness. Also water tank agent contains several communication modules (vary from wired to Wi-Fi) in order to make the communication channels with multiple plant agents.

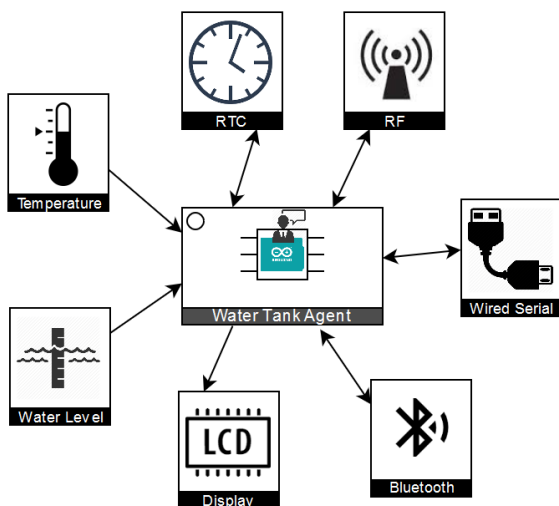


Figure 7.4 : Water Tank Agent Configuration

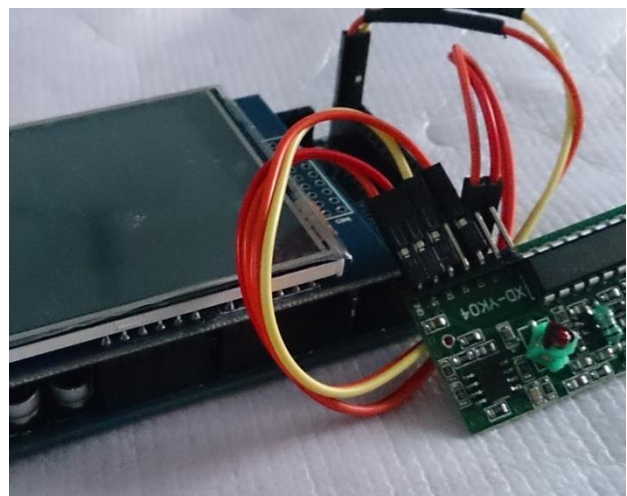


Figure 7.5 : Water Tank Agent

Resource agent module is powered by Arduino Mega (ATmega1280) with the capability of cloud connectivity. As a result of that this agent is able to extract the required information from the cloud whenever needed. Type of information includes weather forecast, historical watering records, etc.

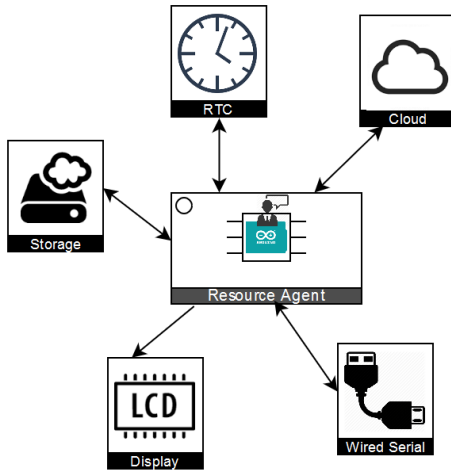


Figure 7.6 : Resource Agent Configuration

Please refer to the Appendix C - Multi-Agent based Home Garden Monitoring System for the complete design details for the modules and detail specifications of each agent module and communication channels.

7.2.2 Define process and behaviors

Home garden monitoring system is capable of executing different processes with the help of different types of agent behaviors. All those agent behaviors are implemented using the previously discussed software framework.

Some of the plant agents have wireless communication channels to communicate with the tank agent while other plant agents have just wires (serial) connections. So that each agent connection was established using the framework accordingly.

One of the major behaviors of the system is, when the water tank agent senses the water level in the water tank is high it will send CFP (call for proposal) for the plant agents. Once plant agents receive the CFP it will calculate the required water amounts based on their sensory inputs. Then send the proposals to the tank agent. The tank agent will evaluate each proposal individually and

accept or reject the proposals accordingly. Finally that result will be transmit to the plant agents with the accept/reject flag. Based on that response plant agents will activate the actuators to turn the water lines on-off. This entire CFP process was implemented using the framework with the help of ACL messages generation and few behaviors such as '*ReceiverBehaviour*', '*OneShotBehaviour*', etc.

During the CFP process which is discuss in the previous section, some plant agent proposals can be rejected. In that case plant agent may be executing the negotiation process in order to get the water access. Based on the sensory inputs and some previous water supply records plant agent will decide to execute the negotiation process or not. If decided agent will adjust the water request (since the original request was rejected) and send back to the tank agent, so that tank agent can process the negotiation request and accept / reject accordingly.

Tank agent will communicate with the resource agent to get additional information such as weather forecast, historical records, etc. before accept / reject the proposals from the plat agents. In addition to that plant agents will communicate with each other's in order to effectively coordinate the watering system.

Appendix D - Sample Codes contains the implementation of the CFP process by using the software framework.

7.2.3 Evaluation results discussion and conclusion

Home garden monitoring system, which was developed using the software framework able to perform essential multi agent features such as communication, coordination and negotiation.

Since the framework contain software functions to establish communication links for both wires and wireless (RF, Wi-Fi and Bluetooth) it was easy to connect different agent with different communication links.

Behavior Engine which is a part of the software framework makes it easier to add required behaviors to the agents. Since the framework supports for ACL message generation and interpretation writing the agent program become really simple.

To conclude this section; it was clear that software framework for the multi agent system makes this entire system development easy to develop, debug and execute.

7.3 Performance Matrices

In this evaluation process same multi agent functionality (ex: communication, negotiation, etc) was develop using framework and from the scratch (base to compare with). And compare the several parameters including;

- Time taken to implement the code
- Quality of the code
- Readability of the code
- Erroneous of the code

Following set of functionalities were used to bench mark the result

- Establish a wired connection between agents
- Establish a wireless connection between agents
- Send messages between agents
- Perform a Call for Proposal (CFP) process
- Perform cyclic behavior
- Agent negotiation process

Results from the each process were analyzed in order to evaluate the quality/ usability of the framework. Following table contains the analysis results and descriptive information about each parameter.

Task	Description	
Establish a wired connection between agents	From Scratch	Need to establish a serial connection between modules and use that connection to communicate with agent. But need to check the communication type before each and every send operation.
	Frame-work	Define connected agent with serial connectivity <code>plantAgent.registerAgent(0, SERIAL_1, "P2");</code>

Establish a wireless connection between agents	From Scratch	Need to initialize and configure several libraries based on the wireless type (ex: RF, Wi-Fi, Bluetooth, etc.) Based on the library / communication type messenger type and protocol will be different
	Frame-work	<p>Define connected agent with Radio Frequency wireless connectivity</p> <pre>plantAgent.registerAgent(0, COMM_RF, "P2", TX_PIN, RX_PIN);</pre> <p>Define connected agent with Wi-Fi connectivity</p> <pre>plantAgent.registerAgent(0, COMM_WF, "P2", TX_PIN, RX_PIN);</pre> <p>Define connected agent with Bluetooth connectivity</p> <pre>plantAgent.registerAgent(0, COMM_BT, "P2", TX_PIN, RX_PIN);</pre>
Send messages between agents	From Scratch	It is required to identify the communication channel type before sending the messages, ex: sending a message over serial communication channel is completely different from sending message from wireless (Wi-Fi) connection.
	Frame-work	<p>Message sending is NOT depending on the communication channel defined.</p> <pre>ACLMessage aclMessage={"P1", "TANK", P_INFORM, "T1", "MSG"};</pre> <pre>plantAgent.send(aclMessage);</pre>
Perform a CFP process	From Scratch	Implementing this operation will take considerable time, user need to implement the flow control for the CFP as well as define the constants to identify the state of each request / agent. In addition to that sending / receiving functions need to be writing according to the each agent's communication type.
	Frame-work	Can be archived by using the <code>BEHAVIOUR_RECEIVER</code> and Performative <code>P_CFP</code> , <code>P_PROPOSE</code> , <code>P_REJECT_PROPOSAL</code> and <code>P_ACCEPT_PROPOSAL</code>
Perform cyclic behavior	From Scratch	User need to use a timer (internal timer or external interrupt based timer) to make a cycle and execute the required functionality within that cycle.

	Frame-work	<p>Add Ticker behavior to the agent</p> <pre>plantAgent.addBehaviour(BEHAVIOUR_TICKER, 1, on5SecTick, 5);</pre> <p>Add code segment to execute periodically</p> <pre>void on5SecTick (){ // code segment }</pre>
Agent negotiation process	From Scratch	Implementing this operation will take considerable time, user need to implement the flow control for the negotiation as well as define the constants to identify the state of each request / agent. In addition to that sending / receiving functions need to be writing according to the each agent's communication type.
	Frame-work	Can be archived by using the BEHAVIOUR_SIMPLE and Performative P_INFORM, P_REJECT_PROPOSAL and P_ACCEPT_PROPOSAL

Table 7.1 : Task Description Table

7.4 Memory Analysis

Since most of the hardware platforms are limited in memory resources; it is required to design and implement the framework with minimum memory foot-print. In order to evaluate that separate memory analysis process is used.

Basically these hardware platforms consists of three types of memories,

- *Flash Memory* : This is the place where executable code of the program stores, memory required for a program will be increase when the LOC is increased.
- *SRAM* : This is the dynamic memory where all the global variables are created. If program uses more variables dynamic memory required for the program will be high. Basically this memory will be cleared once the power-off.
- *EEPROM* : This is an separate memory, which will remain even the power-off. Since the framework did not store any information in EEPROM this was not taken in to the consideration when analyze the memory.

Framework can be use in two modes, debug mode and normal mode. In debug mode framework will produce logging information about each agent action which can be used to identify potential issues with the multi agent system. But in normal mode these information will not be generated. So that framework takes less memory in normal mode compared to the debug mode.

Following table summarize the memory foot-print (percentage of memory consume by the framework) of the framework (in both normal mode and debug mode) for each hardware platform. Please refer the Appendix A for more information about these hardware platforms.

Hardware Platform	Flash Memory			SRAM		
	Total	Normal Mode	Debug Mode	Total	Normal Mode	Debug Mode
<i>Arduino ZERO - ATSAM21G18</i>	256 KB	3%	4%	32 KB	4%	4.5%
<i>Arduino Mega - ATmega2560</i>	256 KB	3%	4%	8 KB	14%	18%
<i>Arduino Uno - ATmega328P</i>	32 KB	27%	35%	2 KB	45%	58%
<i>Arduino Nano - ATmega328</i>	32 KB	29%	37%	2 KB	21%	58%
<i>Arduino Pro Mini - ATmega328</i>	32 KB	29%	37%	2 KB	45%	58%
<i>Arduino Micro - ATmega32U4</i>	32 KB	38%	49%	2.5 KB	35%	45%
<i>Arduino Nano -ATmega168</i>	16 KB	58%	74%	1 KB	42%	-

Table 7.2 : Framework Memory Consumption

7.5 Summary

This chapter discuss about the evaluation process which is used to evaluate the framework as well as analyze the performance and memory foot-print of the framework in different platforms.

Note: *Multi-agent based home garden monitoring system which is design and implement only for the demonstrating the framework functionality. So that it will not have any sophisticated hardware modules which available in most of modern irrigation control system. Goal of developing such a system is not for highlight the features of modern irrigation control but for evaluate the framework functionalities.*

Conclusion and Further work

8.1 Introduction

This chapter summarize the multi agent system which was developed in order to make easier the multi agent system development in embedded systems. Content of this chapter includes to what extend the main objective(s) were archived by the designing and implementation phase and verify using the evaluation process.

8.2 Conclusion

It was hypothesized that the ‘Availability of a common framework for multi agent systems in embedded platforms will improve the development process of multi agent systems in embedded environments.’ In order to prove that hypothesis completely new software framework was developed from the ground up.

Development process starts with the design of the framework modules and interaction with each module. Implementation was done for the most of the modules for the major types of hardware platforms available in the market.

Finally, the evaluation process and results were presented in the chapter 7. According to the evaluation results it is clear that framework improves the development process, thus the objective was archived.

In addition to the main objective following objectives were also define at the initial stage of the process.

- Identify the popular hardware platforms and develop a complete software framework for those identified platforms to make the multi agent based development easy.
- Provide support for the major multi agent behaviors, protocols and communication standards.

- Make the software framework available to the general public with the complete documentation, under the license of free and open source.

Thus all the above objectives were met in the research as anticipated, as a concluding note it is clear that overall system development and evaluation process was succeeded.

8.3 Limitations and Further Works

Regarding the limitations, one of the major limitations is that framework is not implemented for every hardware platform available in the market at the time of development is done, but only for the few selected major hardware platforms only. Since there are number of various platforms it is difficult to implement code for the every platform is nearly impossible with the available time frame. Since the modular design of the framework allows users to extend the implementation very easily, users can implement the modules for other hardware platforms as well.

Few things were identify as potential further works, to make the framework more useful and powerful.

One will be integrate the framework with the dashboard like UI to see the status of the each agent separately. In addition it will make easier to display the intermediate state of all agents in more readable manner. ‘Cayenne’ from ‘myDevice’[19] was identified as potential platform to establish this connection.

Another identified further work will be integrate the framework with some cloud based data collection / storage, so that each agent can store the data related to the status of the agent and use that whenever needed. ‘Thingspeak’[20] which is an open data platform was identified as potential candidate for this work.

8.4 Summary

This chapter summarize about the conclusion, limitations and future works of the developed multi agent development framework for the embedded platforms. The limitations were identified and potential solutions also discussed. Few addition works were identified as further works in order to make the framework more useful and powerful.

References

- [1] F. Bellifemine, A. Poggi, and G. Rimassa, “JADE—A FIPA-compliant agent framework,” in *Proceedings of PAAM*, 1999, vol. 99, p. 33.
- [2] “Welcome to the Foundation for Intelligent Physical Agents,” *Foundation for Intelligent Physical Agents (FIPA)*. [Online]. Available: <http://www.fipa.org/>. [Accessed: 21-Aug-2015].
- [3] “Arduino - Home.” [Online]. Available: <https://www.arduino.cc/>. [Accessed: 21-Aug-2015].
- [4] A. C. -, M. C. R. -, F. S. -, M. D. H. -, and J. I. E. -, “Multi-Agent and Embedded System Technologies Applied to Improve the Management of Power Systems,” *Int. J. Digit. Content Technol. Its Appl.*, vol. 4, no. 1, pp. 79–85, Feb. 2010.
- [5] H.-M. Kim, Y. Lim, and T. Kinoshita, “An Intelligent Multiagent System for Autonomous Microgrid Operation,” *Energies*, vol. 5, no. 12, pp. 3347–3362, Sep. 2012.
- [6] S. Madakam, R. Ramaswamy, and S. Tripathi, “Internet of Things (IoT): A Literature Review,” *J. Comput. Commun.*, vol. 03, no. 05, pp. 164–173, 2015.
- [7] Q. Sun, W. Yu, N. Kochurov, Q. Hao, and F. Hu, “A Multi-Agent-Based Intelligent Sensor and Actuator Network Design for Smart House and Home Automation,” *J. Sens. Actuator Netw.*, vol. 2, no. 3, pp. 557–588, Aug. 2013.
- [8] J. Baumann, F. Hohl, Prof. Dr. K. Rothermel, and M. Straßer, “Mole - A Java based Mobile Agent System,” *Spec. Issues Object Oriented Program.*, pp. 301–308, 1997.
- [9] “SquidBee.” [Online]. Available: <http://www.atmel.com/products/avr32/>.
- [10] T. Kinoshita and K. Sugawara, “ADIPS Framework for Flexible Distributed Systems,” in *Multiagent Platforms*, T. Ishida, Ed. Springer Berlin Heidelberg, 1998, pp. 18–32.
- [11] F. Bellifemine, A. Poggi, and G. Rimassa, “JADE—A FIPA-compliant agent framework,” in *Proceedings of PAAM*, 1999, vol. 99, p. 33.
- [12] “Microcontroller,” *Wikipedia, the free encyclopedia*. 25-Feb-2016.
- [13] “Pulse-width modulation,” *Wikipedia, the free encyclopedia*. 20-Jan-2016.

- [14] “Microcontrollers - A Beginner’s Guide - Introduction to Interrupts using an LED and 330 ohm resistor.” [Online]. Available: <https://www.newbiehack.com/IntroductiontoInterrupts.aspx>. [Accessed: 05-Mar-2016].
- [15] C. Madrigal Mora, “A model-driven approach for organizations in multiagent systems,” 2013.
- [16] T. FIPA, “Fipa communicative act library specification,” *Found. Intell. Phys. Agents Httpwww Fipa Orgspecs fipa00037SC00037J Html 306 2004*, 2008.
- [17] D. Isern, S. Abelló, and A. Moreno, “Development of a multi-agent system simulation platform for irrigation scheduling with case studies for garden irrigation,” *Comput. Electron. Agric.*, vol. 87, pp. 1–13, Sep. 2012.
- [18] S. Balbi, S. Bhandari, A. K. Gain, and C. Giupponi, “Multi-agent agro-economic simulation of irrigation water demand with climate services for climate change adaptation,” *Ital. J. Agron.*, vol. 8, no. 3, p. 23, Sep. 2013.
- [19] “Cayenne Docs.” [Online]. Available: http://www.cayenne-mydevices.com/docs/?utm_campaign=website&utm_source=sendgrid.com&utm_medium=email#introduction. [Accessed: Oct-2016].
- [20] “Internet Of Things - ThingSpeak.” [Online]. Available: <https://thingspeak.com/>. [Accessed: Oct-2016].
- [21] M. Wooldridge, “*An Introduction to MultiAgent Systems*”. 2002.
- [22] Russell, Stuart J., Peter Norvig, and John Canny. “*Artificial Intelligence: A Modern Approach*”. 2003.

Appendices

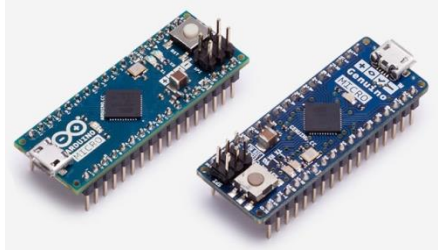
Appendix A:



Arduino Hardware Platform

A.1 Introduction

Arduino hardware platform is one of the major embedded system platforms available in the market. There are number of different types of boards are available with different hardware specification. Based on the requirement (memory, number of I/O pints, interrupts, timers, PWM, etc) user can select most appropriate board for the development. Following section summarize hardware specifications of major Arduino platforms available in market.

A.2 Hardware Specifications

Arduino Platform	Specifications	
<p>Arduino MICRO</p> 	Microcontroller	ATmega32U4
	Operating Voltage	5V
	Input Voltage	7-12V
	Input Voltage (limit)	6-20V
	Digital I/O Pins	20
	PWM Channels	7
	Analog Input Channels	12
	DC Current per I/O Pin	20 mA
	DC Current for 3.3V Pin	50 mA
	Flash Memory	32 KB (ATmega32U4)
	SRAM	2.5 KB (ATmega32U4)
	EEPROM	1 KB (ATmega32U4)
	Clock Speed	16 MHz
	LED_BUILTIN	13
	Length	48 mm
	Width	18 mm
	Weight	13 g

<p style="text-align: center;">Arduino NANO</p> 	<table border="0"> <tr> <td>Microcontroller</td> <td>Atmel ATmega168 or ATmega328</td> </tr> <tr> <td>Operating Voltage (logic level)</td> <td>5 V</td> </tr> <tr> <td>Input Voltage</td> <td>7-12 V</td> </tr> <tr> <td>Input Voltage (limits)</td> <td>6-20 V</td> </tr> <tr> <td>Digital I/O Pins</td> <td>14 (of which 6 provide PWM output)</td> </tr> <tr> <td>Analog Input Pins</td> <td>8</td> </tr> <tr> <td>DC Current per I/O Pin</td> <td>40 mA</td> </tr> <tr> <td>Flash Memory</td> <td>16 KB (ATmega168) or 32 KB (ATmega328)</td> </tr> <tr> <td>SRAM</td> <td>1 KB (ATmega168) or 2 KB (ATmega328)</td> </tr> <tr> <td>EEPROM</td> <td>512 bytes (ATmega168) or 1 KB (ATmega328)</td> </tr> <tr> <td>Clock Speed</td> <td>16 MHz</td> </tr> <tr> <td>Dimensions</td> <td>0.73" x 1.70"</td> </tr> <tr> <td>Length</td> <td>45 mm</td> </tr> <tr> <td>Width</td> <td>18 mm</td> </tr> <tr> <td>Weight</td> <td>5 g</td> </tr> </table>	Microcontroller	Atmel ATmega168 or ATmega328	Operating Voltage (logic level)	5 V	Input Voltage	7-12 V	Input Voltage (limits)	6-20 V	Digital I/O Pins	14 (of which 6 provide PWM output)	Analog Input Pins	8	DC Current per I/O Pin	40 mA	Flash Memory	16 KB (ATmega168) or 32 KB (ATmega328)	SRAM	1 KB (ATmega168) or 2 KB (ATmega328)	EEPROM	512 bytes (ATmega168) or 1 KB (ATmega328)	Clock Speed	16 MHz	Dimensions	0.73" x 1.70"	Length	45 mm	Width	18 mm	Weight	5 g				
Microcontroller	Atmel ATmega168 or ATmega328																																		
Operating Voltage (logic level)	5 V																																		
Input Voltage	7-12 V																																		
Input Voltage (limits)	6-20 V																																		
Digital I/O Pins	14 (of which 6 provide PWM output)																																		
Analog Input Pins	8																																		
DC Current per I/O Pin	40 mA																																		
Flash Memory	16 KB (ATmega168) or 32 KB (ATmega328)																																		
SRAM	1 KB (ATmega168) or 2 KB (ATmega328)																																		
EEPROM	512 bytes (ATmega168) or 1 KB (ATmega328)																																		
Clock Speed	16 MHz																																		
Dimensions	0.73" x 1.70"																																		
Length	45 mm																																		
Width	18 mm																																		
Weight	5 g																																		
<p style="text-align: center;">Arduino UNO</p> 	<table border="0"> <tr> <td>Microcontroller</td> <td>ATmega328P</td> </tr> <tr> <td>Operating Voltage</td> <td>5V</td> </tr> <tr> <td>Input Voltage</td> <td>7-12V</td> </tr> <tr> <td>Input Voltage (limit)</td> <td>6-20V</td> </tr> <tr> <td>Digital I/O Pins</td> <td>14 (of which 6 provide PWM output)</td> </tr> <tr> <td>PWM Digital I/O Pins</td> <td>6</td> </tr> <tr> <td>Analog Input Pins</td> <td>6</td> </tr> <tr> <td>DC Current per I/O Pin</td> <td>20 mA</td> </tr> <tr> <td>DC Current for 3.3V Pin</td> <td>50 mA</td> </tr> <tr> <td>Flash Memory</td> <td>32 KB (ATmega328P)</td> </tr> <tr> <td>SRAM</td> <td>2 KB (ATmega328P)</td> </tr> <tr> <td>EEPROM</td> <td>1 KB (ATmega328P)</td> </tr> <tr> <td>Clock Speed</td> <td>16 MHz</td> </tr> <tr> <td>LED_BUILTIN</td> <td>13</td> </tr> <tr> <td>Length</td> <td>68.6 mm</td> </tr> <tr> <td>Width</td> <td>53.4 mm</td> </tr> <tr> <td>Weight</td> <td>25 g</td> </tr> </table>	Microcontroller	ATmega328P	Operating Voltage	5V	Input Voltage	7-12V	Input Voltage (limit)	6-20V	Digital I/O Pins	14 (of which 6 provide PWM output)	PWM Digital I/O Pins	6	Analog Input Pins	6	DC Current per I/O Pin	20 mA	DC Current for 3.3V Pin	50 mA	Flash Memory	32 KB (ATmega328P)	SRAM	2 KB (ATmega328P)	EEPROM	1 KB (ATmega328P)	Clock Speed	16 MHz	LED_BUILTIN	13	Length	68.6 mm	Width	53.4 mm	Weight	25 g
Microcontroller	ATmega328P																																		
Operating Voltage	5V																																		
Input Voltage	7-12V																																		
Input Voltage (limit)	6-20V																																		
Digital I/O Pins	14 (of which 6 provide PWM output)																																		
PWM Digital I/O Pins	6																																		
Analog Input Pins	6																																		
DC Current per I/O Pin	20 mA																																		
DC Current for 3.3V Pin	50 mA																																		
Flash Memory	32 KB (ATmega328P)																																		
SRAM	2 KB (ATmega328P)																																		
EEPROM	1 KB (ATmega328P)																																		
Clock Speed	16 MHz																																		
LED_BUILTIN	13																																		
Length	68.6 mm																																		
Width	53.4 mm																																		
Weight	25 g																																		

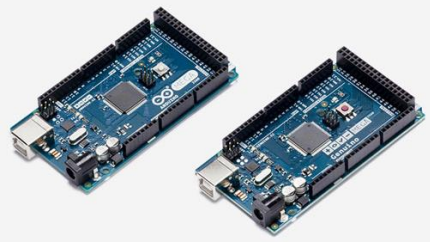
<p style="text-align: center;">Arduino MEGA</p> 	Microcontroller	ATmega2560
	Operating Voltage	5V
	Input Voltage	7-12V
	Input Voltage (limit)	6-20V
	Digital I/O Pins	54 (of which 15 provide PWM output)
	Analog Input Pins	16
	DC Current per I/O Pin	20 mA
	DC Current for 3.3V Pin	50 mA
	Flash Memory	256 KB
	SRAM	8 KB
	EEPROM	4 KB
	Clock Speed	16 MHz
	LED_BUILTIN	13
	Length	101.52 mm
Width	53.3 mm	
Weight	37 g	

Table A.1 : Arduino Hardware Specifications

Hardware Modules

B.1 Introduction

Number of hardware modules was used during the implementation and evaluation process of the framework. Including sensor modules, communication modules, actuators, etc. Following sub sections will provide detail about each and every hardware modules.

B.2 RF Module

Pair of RF (Radio Frequency) Modules were used in the evaluation process as Transmitter and Receiver. Hardware specifications of these two modules are summarizing in the following table.



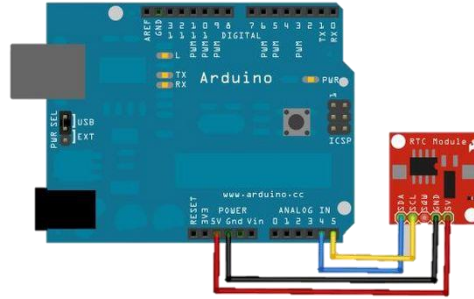
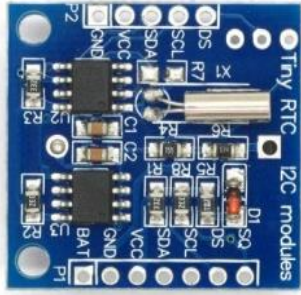
	Transmitter Module	Receiver Module
		
Working voltage	3V - 12V	5.0VDC +0.5V
Working current	9mA - 40mA	≤5.5mA max
Resonance mode	SAW	OOK/ASK
Modulation mode	ASK	-
Working frequency	315MHz Or 433MHz	315MHz-433.92MHz
Transmission power	25mW	-
Frequency error	+150kHz (max)	-
Velocity	less than 10Kbps	<9.6Kbps
Connections	VCC, GND and DATA	

Table B.1 : RF Module Specifications

B.3 RTC Module

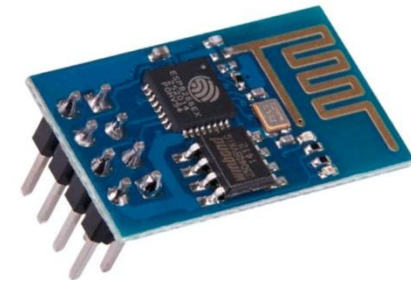
RTC (Real Time Clock) module used in the system was based on the DS1307. This module contains the DS1307, Crystal and a Battery. Communication with the module can be established by the I2C connection.



Connections	
GND	→ +5v
VCC	→ Ground
SCL	→ Clock Signal
SDA	→ Data Signal

B.4 Wi-Fi Module

ESP-8266 module was used to establish the Wi-Fi communication channel between agents. Since the module operational voltage is 3.3v and Arduino operate on 5v it is required to have a regulator in between. FTDI FT232RL was used to archive the voltage regulation.

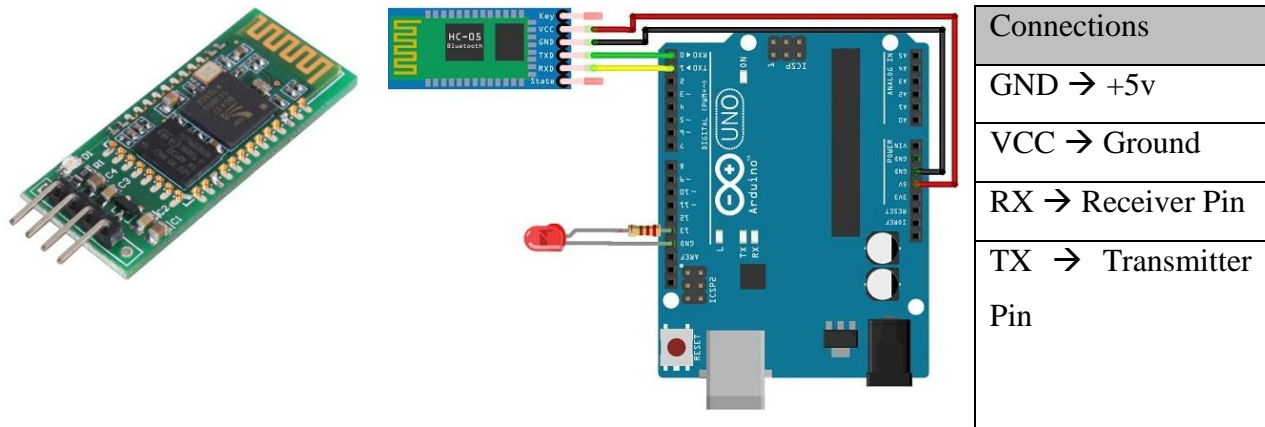


Connections	
VCC	shall be connected to the 3.3V power supply
GPIO0	controls the module mode (programming or normal operation)
CH_PD	Chip Enable High (3.3V) for normal operation
RST	Reset, High (3.3V) for normal operation. 0V to reset the chip.
TX	Transmit Pin
RX	Receive Pin
GND	Ground

Table B.2 : Wi-Fi Module Connection Details

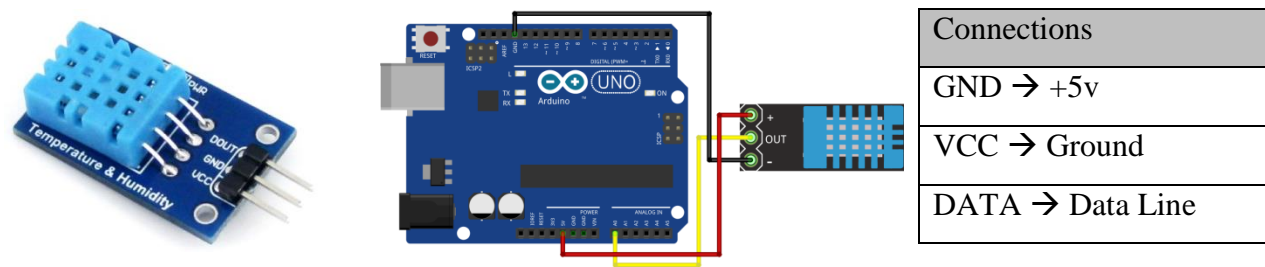
B.5 Bluetooth Module

Bluetooth module used in the development of the system is HC-05, which contains both master and slave modes. Basic connection can be establish through the serial communication channel and can configure using the AT commands.



B.6 DHT Module

DHT11 digital temperature and humidity sensor is a composite Sensor contains a calibrated digital signal output of the temperature and humidity. Communication with the sensor can be establish using the given 1-wire protocol.



Relative Humidity	Temperature
Resolution : 16Bit Repeatability : ±1% RH Accuracy : At 25°C±5% RH Interchangeability : fully interchangeable Response time : 1 / e (63%) of 25°C 6s Hysteresis : < ±0.3% RH Long-term stability : < ±0.5% RH	Resolution : 16Bit Repeatability : ±0.2°C Range : At 25°C±2°C Response time : 1 / e (63%) 10S

Table B.3 : DHT Module Specifications

B.7 OLED Display Module

OLED Display module used in system was an OLED monochrome display (yellow and blue) module which contains 128X64 pixels; drive by the SSD1306 driver IC.

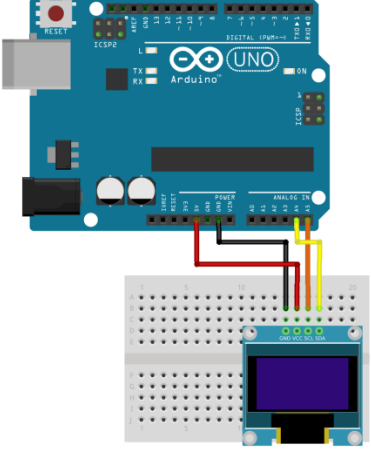
Features			Connections
Dimensions :	2.2 cm x 2.8 cm x 1.2 cm		
Screen size :	0.96'' (128px by 64px)		VCC → Ground
Display Color :	Monochrome		SCL → Clock Line
Total Connection :	Wires : 4		SDA → Data Line
Power :	5V DC (PIN-VCC)		
Communication :	I2C		

Table B.4 : OLED Module Features

B.8 Ethernet Shield

Arduino Ethernet shield was used to establish the connection to the internet. This shield can be plug in to most of the Arduino platforms and connect to the internet using RJ45 standard Ethernet connection.

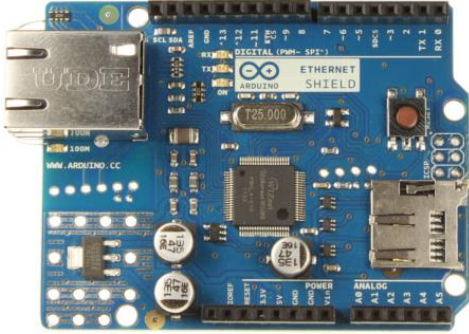
	Features
	Low output ripple and noise (100mVpp)
	Input voltage range 36V to 57V
	Overload and short-circuit protection
	9V Output
	High efficiency DC/DC converter: 75% at 50% load
	1500V isolation (input to output)

Table B.5 : Arduino Ethernet Shield

Multi-Agent based Home Garden Monitoring System

C.1 Introduction

This section will discuss the design and implement process of the multi-agent based Home Garden Monitoring System. This system was developed in order to evaluate the framework.

This system basically contains three types of modules (agents). Plant Module will place near the plant to be monitored. Three plant modules were used during the evaluation process. All three plant agent modules are identical to each one except from the capabilities of the communication. First agent module contains single wireless communication line, Second agent contains one wired communication link and third with both wired and wireless communication links.

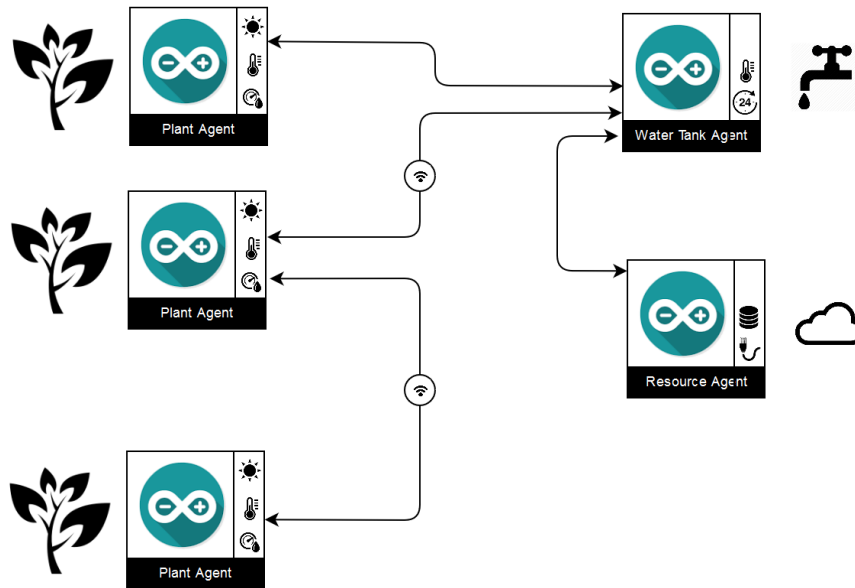
C.2 Hardware Module Specifications

Plant agent module was implemented using Arduino NANO platform. OLED screen module was used to display the status / information about the agent. DHT11 sensor was connected to the plant agent to sense the temperature and humidity of the atmosphere. Real Time Clock (RTC) module in the plant agent was used to keep the date and time information. Wired communication link in the plant agent module is establish by using the serial interface, while wireless channel was implemented using the two RF modules (Receiver module and Transmit module)

Water tank modules was build using the Arduino MEGA platform and consist of thermostat, a RTC module and LCD screen was used to display the agent status and communication logs between the agents. Wired communication was established using the inbuilt serial interface while wireless communication was archived by the pair of RF modules connected to the module.

Resource agent module was build using the Arduino MEGA platform. It contains Ethernet shield which is used to connect to the internet. In addition to that SD card reader / writer module is connected to the resource agent to store the daily records locally.

C.3 Module Connection Details



Two plant agents are connected to the tank agent directly, while third plant agent is connected to the second plant agent only. This type of agent configuration is used to demonstrate the indirect agent communication behavior of the framework. (Third plant agent is indirectly connected to the tank agent via the second plant agent.)

Water tank agent has three communication links, two wired links and one wireless link. It was connected to the resource agent via one of this wired link. Resource agent has only one wired link, and separate link to connect to the Internet.

Appendix D:

Sample Codes

D.1 Introduction

Following sub section will discuss how to use the software framework when implementing multi agent system. Please note that this will not cover each and every functionality, methods and features of the framework. Complete documentation on multi-agent framework will be available on-line.

D.2 Agent Initialization

In order to initialize the agent it is only required to add the following line segment, First parameter for the constructor is the name of the agent and the second parameter will be number of possible communication channels.

```
Agent plantAgent( "P1", 2 );
```

Once you define the agent; communication channels can be registered as below. Following code segment will register one wired communication channel to agent "P2" and one wireless - RF communication channel to agent "TANK"

```
plantAgent.registerAgent(0, SERIAL_1, "P2");  
plantAgent.registerAgent(1, SERIAL_RF, "TANK", RX_PIN, TX_PIN);
```

D.3 ACL Message

Once the communication channels were added to the agent, user can send and receive ACL messages to/from other agents as shown below.

Send ACL message from agent P1 to P2,

```
ACLMessage aclMessage = { "P1", "P2", P_INFORM, "Topic", "Content" };  
plantAgent.send( aclMessage );
```

Receive ACL message from other agents,

```
plantAgent.addMessageReceivedEvent( onMessageReceive );

void onMessageReceive( ACLMessage aclMessage ){
    Serial.println (aclMessage.sender + " " + aclMessage.content );
}
```

D.4 CFP Process Implementation

Following code segments demonstrate the implementation of simple CFP process between agents.

```
String agentName = "M";
Agent motorAgent( agentName, 2 );

void setup(){
    motorAgent.registerAgent(0, SERIAL_1, "S1");
    motorAgent.registerAgent(1, SERIAL_2, "S2");
    motorAgent.addMessageReceivedEvent( onMessageReceive );

    setupCFP();
}

void setupCFP(){
    String topic = "Distance";
    // Initiate the process by sending ACL message - CFP
    ACLMessage cfpMessage = { agentName, "-", P_CFP, topic, "CM" };
    motorAgent.sendToAll( cfpMessage );
    // register new behaviour for the CFP
    motorAgent.addBehaviour( BEHAVIOUR_RECEIVER,0,onReceiverBehaviour );
}

// status 0 - CFP is send, status 1 - accepted
int status = 0;
int lDistance = 1000;
String aAgent = "";
int pCount = 0;

void onReceiverBehaviour( ACLMessage aclMessage ){
    if ( aclMessage.topic == "Distance" ){
        // continue the CFP process
        switch ( status ){
            case 0 :
            {
                // We are getting proposals from the other agents
```

