

# 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](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

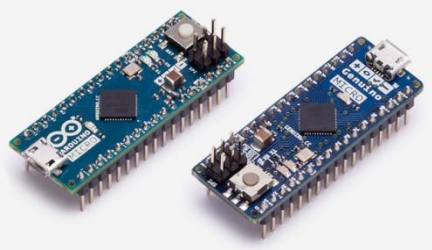
## Appendix A:



### 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																																		

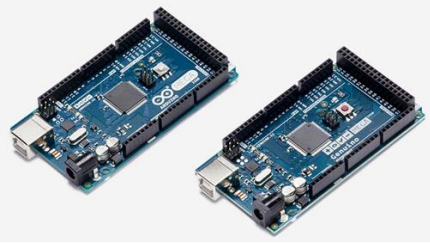
<p style="text-align: center;">Arduino MEGA</p> 	Microcontroller	ATmega2560
	Operating Voltage	5V
	Input Voltage	7-12V
	Input Voltage (limit)	6-20V
	Digital I/O Pins	54 (of which 15 provide PWM output)
	Analog Input Pins	16
	DC Current per I/O Pin	20 mA
	DC Current for 3.3V Pin	50 mA
	Flash Memory	256 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

## Appendix B:

### 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.



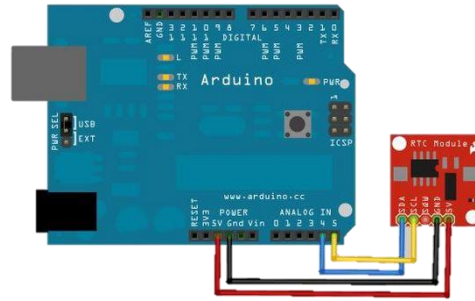
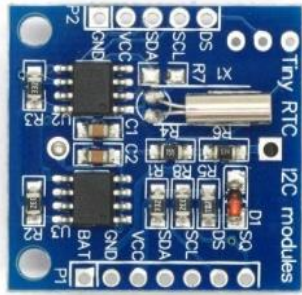
	Transmitter Module	Receiver Module
		
Working voltage	3V - 12V	5.0VDC +0.5V
Working current	9mA - 40mA	≤5.5mA max
Resonance mode	SAW	OOK/ASK
Modulation mode	ASK	-
Working frequency	315MHz Or 433MHz	315MHz-433.92MHz
Transmission power	25mW	-
Frequency error	+150kHz (max)	-
Velocity	less than 10Kbps	<9.6Kbps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

Table B.1 : RF Module Specifications



### B.3 RTC Module

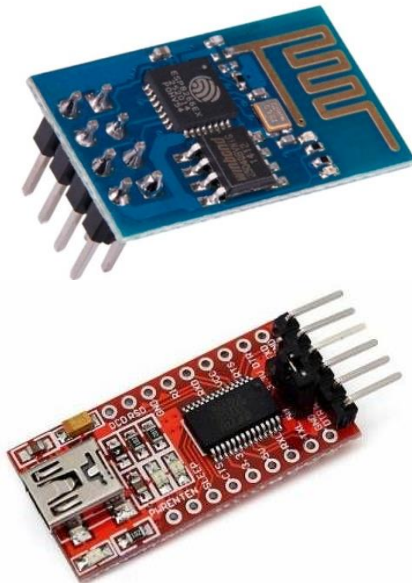
RTC (Real Time Clock) module used in the system was based on the DS1307. This module contains the DS1307, Crystal and a Battery. Communication with the module can be established by the I2C connection.



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.

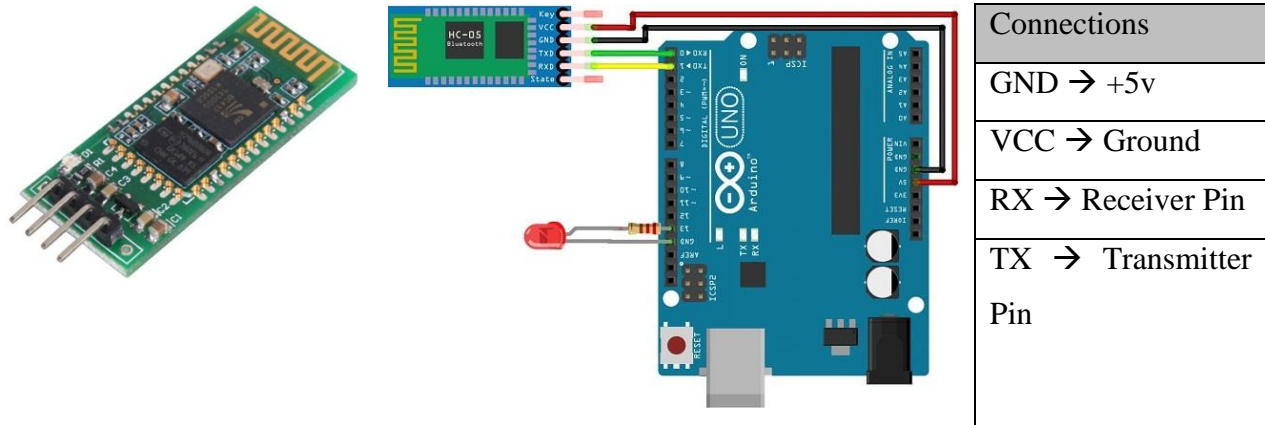


Connections	
VCC	shall be connected to the 3.3V power supply
GPIO0	controls the module mode (programming or normal operation)
CH_PD	Chip Enable High (3.3V) for normal operation
RST	Reset, High (3.3V) for normal operation. 0V to reset the chip.
TX	Transmit Pin
RX	Receive Pin
GND	Ground

Table B.2 : Wi-Fi Module Connection Details

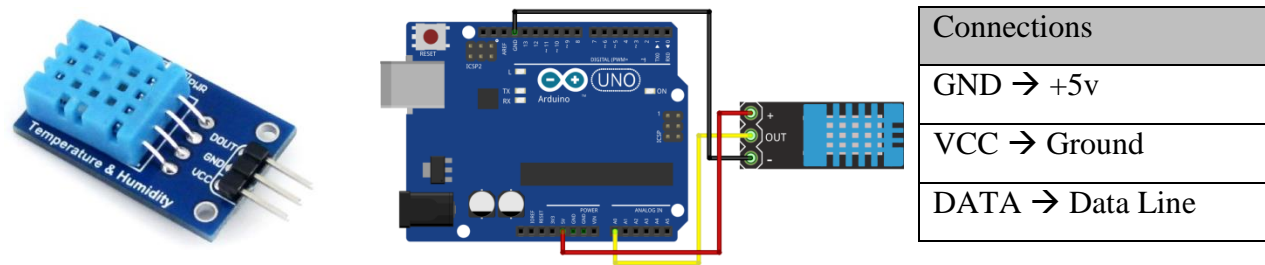
### B.5 Bluetooth Module

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.

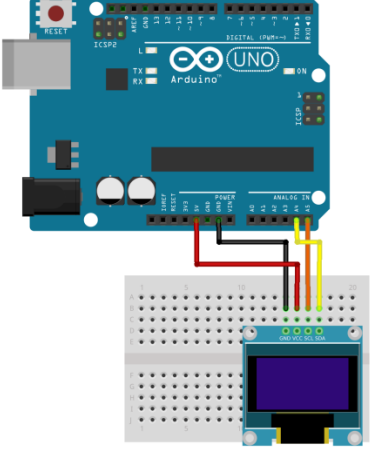
Features			Connections
Dimensions :	2.2 cm x 2.8 cm x 1.2 cm		GND → +5v
Screen size :	0.96'' (128px by 64px)	VCC → Ground	
Display Color :	Monochrome	SCL → Clock Line	
Total Connection :	Wires : 4	SDA → Data Line	
Power :	5V DC (PIN-VCC)		
Communication :	I2C		

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.

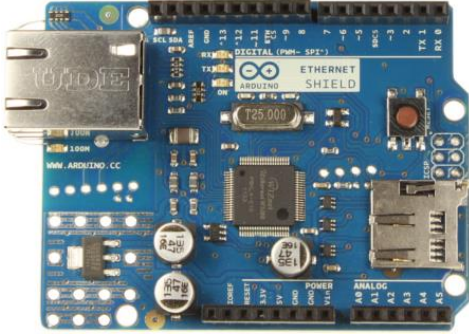
	Features
	IEEE802.3af compliant
Low output ripple and noise (100mVpp)	
Input voltage range 36V to 57V	
Overload and short-circuit protection	
9V Output	
High efficiency DC/DC converter: 75% at 50% load	
1500V isolation (input to output)	

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.

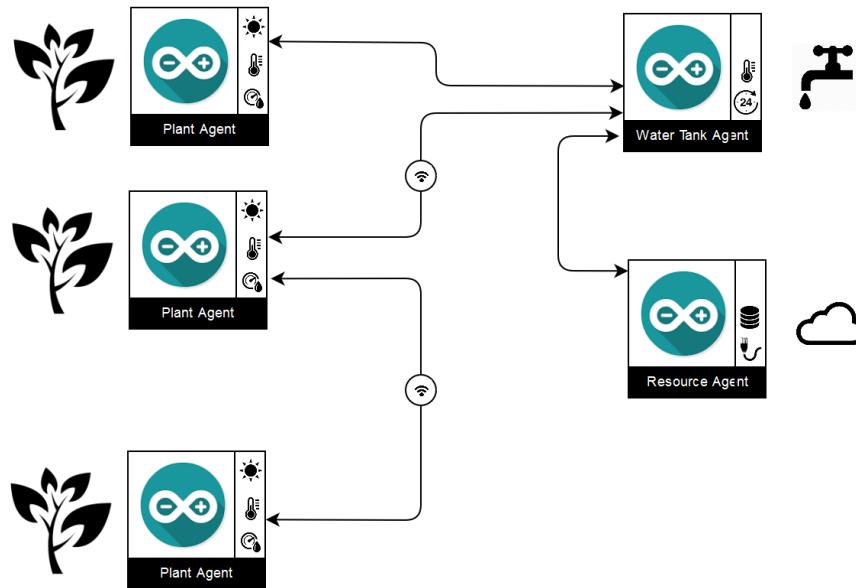
## C.2 Hardware Module Specifications

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
```

