Multi-Disciplinary Senior Design Conference
Kate Gleason College of Engineering
Rochester Institute of Technology
Rochester, New York 14623

Project Number: P11216

WANDERING CAMPUS AMBASSADOR (PART 6)

Nicholas Leathe (*Team Lead*)
Mechanical Engineering

Anna Gilgur
Mechanical Engineering

Terra McAndrew
Industrial Design

Rui Zhou
Electrical Engineering

Kenneth Hertzog
Computer Engineering

Joseph Stevens
Software Engineering

Philip Gibson
Software Engineering

David Ladner
Software Engineering

ABSTRACT

The Wandering Campus Ambassador is a robotic system designed to raise awareness of self-sustaining energy. The main purpose of the P11216 project team is to fully define, develop, and implement the software programming to make the Wandering Campus Ambassador move and take care of the plant autonomously. Additional objectives included working with P11215 (Wandering Campus Ambassador (part 5)) to complete and integrate the electrical, computer, and mechanical aspects of the build, ensure the robot's safety and reliability issues were resolved, and to prepare the robot for the ImagineRIT event in May 2011. All of the customer needs that were of the highest importance were met, appropriate safety and reliability mitigation aspects were implemented, and the Wandering Ambassador navigated and took care of the plant autonomously.

NOMENCLATURE

Angstrom-Linux based operating system
Beagleboard-Single board computer used for majority of data processing
HTML-HyperText Markup Language
I2C-Inter-Integrated Circuit
Java-Programming Language
JNI-Java Native Interface
GPS-Global Positioning System
GUI-Graphic User Interface
MSP430- Microcontroller made by Texas Instruments
MySQL-Database management system
OS-Operating System
PCB-Printed Circuit Board
PHP-PHP: Hypertext preprocessor
QNX-Unix-like operating system
RIT- Rochester Institute of Technology
UML-Unified Modeling Language
XAMPP-Cross Platform, Apache HTTP Server, MySQL, PHP, Perl

PROJECT BACKGROUND

The Wandering Campus Ambassador is a robot intended to raises the campus awareness of self-sustainable energy by taking care of a plant. The robot will make maximum use of natural conditions, such as sunlight/shade, temperature, and water, to take care of the plant and allow the robot to be self-sustaining. There were five other groups that worked on this project previously. This work follows the work of P10215, P10216, P10217, P10218, and P11215.

P10215 and P01216 were the first two teams to work on the Wandering Campus Ambassador project and worked during the fall and winter terms of the 2009 academic school year. P10215, "Robot Locomotion and Plant Platform", was responsible for the shell design, frame, motors, motor drive train, plant, MSP430's for the motor, plant and navigation sensors, plant electronics and MSP430 software.[1] P10216, "Robot Navigation Plant Platform", was responsible for the Beagleboard, Java OS, I2C serial interfaces, GPS, accelerometer, wireless to remote server, and the initial autonomous software.[2]

The next two teams to work on the Wandering Campus Ambassador project were P10217 and P10218. They worked on the project during the winter and spring terms of the 2009 academic school year. P10217, "Robot Integration and Field Testing", was responsible for the body shell build, wireless game pad, refining the motor control software with motor encoders, motherboard PCB, power, and interface protocol documentation updates.[3] P210218, "Robot Applications", was responsible for UML architecture/class diagrams, HTML, PHP, JNI, XAMPP for Windows, RIT server interface, webcam, autonomous software, sonar sensor debugging, and social media integration.[4]

P11215, the fifth team to work on the Wandering Ambassador, worked during the fall and winter terms of the 2010 academic school year. P11215, "Wandering Campus Ambassador (Part 5)", was responsible for refining and redesigning the mechanical, electrical, and software functions. [5]

In addition to designing and implementing the software programming and making the Wandering Ambassador fully autonomous, the current project, P11216, was tasked with many additional tasks in order to ready the robotic system for ImagineRIT. This included improved the range of the sonar sensors, adding a physical deterrence to the alarm system, building holders and allocated space for additional plants, resolved safety and reliability issues, and integrating all aspects of the Wandering Ambassador together. [6] Figure 1 is the completed Wandering Ambassador with the P11216 team.

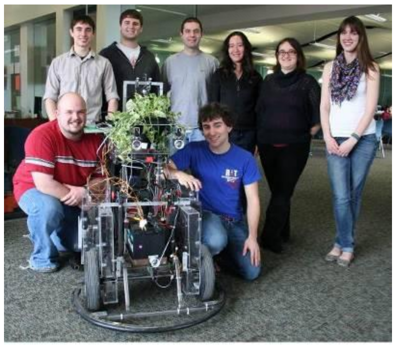
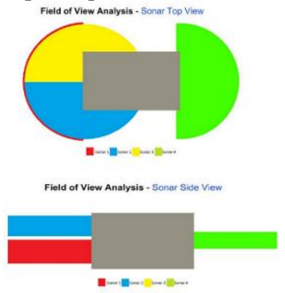


Figure 1 - Completed Wandering Ambassador

MECHANICAL DESIGN

Servo/Sonar Holders

The sonar sensors are responsible for the detection of objects and safe robot navigation, preventing the Wandering Campus Ambassador from colliding with other objects. The placement was optimized from previous teams to maximize field of view. Four sonar sensors were mounted on servo motors and rotate 180°. There is one located on the back of the robot, and three located on the front. Two forward facing sonars are located near the top of the robot chassis, on either side of the plant holder, while the other forward facing sonar is located near the bottom of the chassis. The placement of the sonars and servos on the top, front of the Wandering Ambassador lead to the impression that it has a face; a customer need for previous teams. Figure 2 is a field of view analysis for the placement of the sonars and servos. Figure 3 is an image part that was made to attach the servos to the sonars. Figure 4 shows the sonars and servos on the front of the Wandering Ambassador. Finally, Figure 5 shows the sonar and servo that is located on the back of the Wandering Campus Ambassador.


Field of View Analysis - Sonar Top View
Field of View Analysis - Sonar Side View
Figure 2 - Field of View Analysis for Placement of Sonars and Servos

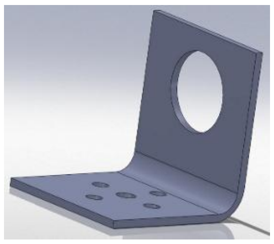


Figure 3 – Part Model for the Servo-Sonar Bracket

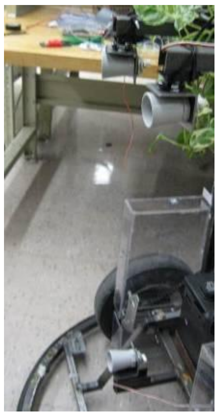


Figure 4 – Sonars/Servos Located at the Front of the Wandering Ambassador

Figure 5 – Sonar/Servo Located at the Back of the Wandering Ambassador

Physical Deterrence System

Integrated by the previous teams was an alarm system that activates when the plant is removed from its location and serves to act as an anti-vandalism measure. When P11216 received the project, the alarm system was only auditory in nature. The idea was proposed to the customer to include a physical deterrence as well, of which the customer approved. An additional reservoir and pump were installed on the Wandering Ambassador for the physical deterrence system. They were attached to two windshield washer nozzles that are located on the top of the chassis. The nozzles glow red when turned on. Figure 6 shows the reservoirs and pumps on the Wandering Ambassador. Figure 7 shows the nozzles on the Wandering Ambassador, while figure 8 shows the nozzles glowing.

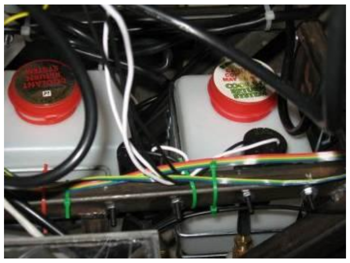


Figure 6 - Reservoirs and Tanks on Wandering Ambassador

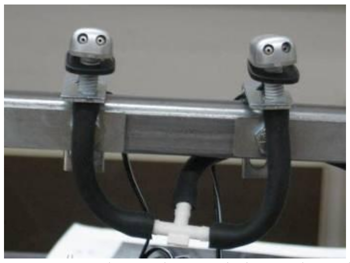


Figure 7 – Nozzles on the Wandering Ambassador

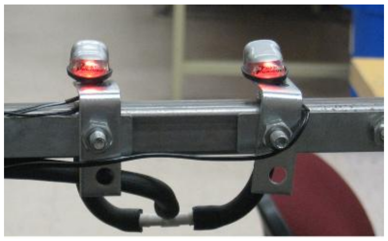


Figure 8 - Red Glowing Nozzles on the Wandering Ambassador

Additional Plant Holder

The customer requested that additional plants be added to the robot to promote the technology and environment integration. Currently the Wandering Ambassador does not care for these plants, but the ability to do so will be added in future iterations of the project. It was determined that the best location for the additional plants was on top of the gears and driveshaft since there was a large, open space there. The plant holders were made and attached with Velcro on top of the gear cover so that; if needed, it will be possible to easily remove the additional plants. The plants grew on a structure that was constructed out of reeds and attached to various points along the side of the robot. Figure 9 is an image of the additional plant holders. Figure 10 is the additional plant holders with plants on the Wandering Ambassador.

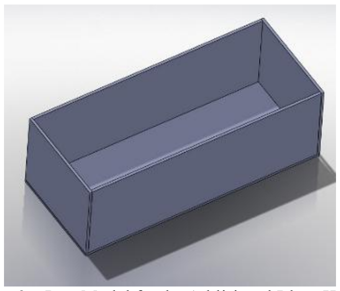


Figure 9 – Part Model for the Additional Plant Holder

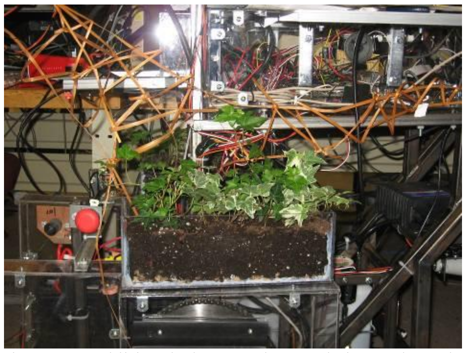


Figure 10 - Additional Plants on the Wandering Ambassador

ELECTRICAL AND COMPUTER DESIGN

The electronics are all stored inside a box located towards the back of the Wandering Ambassador. Figure 11 is a 3D CAD model of the location of the electronics box. Figure 12 is an image of the wiring inside the box.

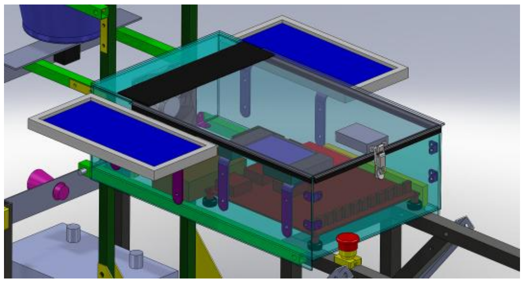


Figure 11 - 3D CAD Model of Electronics Box

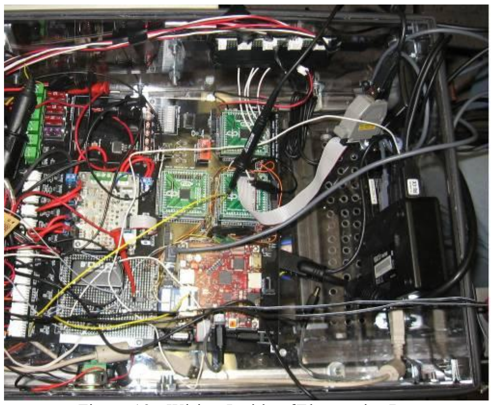
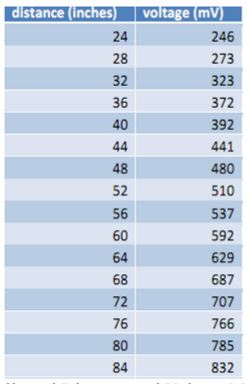


Figure 12 - Wiring Inside of Electronics Box

Sonars/Servos

The sonar range was tested in the electrical engineering laboratory. The voltage the sonars read was tested up to a range of 7 feet. A linear relationship was found between the distance and the voltage read by the sonar. A voltage reading indicated that the sonar identified an object in its range of vision, which is two feet diameter. The sonars were attached onto the robot and wired to the Beagleboard. The servos were soldered to the protoboard, which was then attached to the Beagleboard. Table 1 is the collected distance and voltage measurements. Figure 13 is the Beagleboard.



distance (inches)	voltage (mV)
24	246
28	273
32	323
36	372
40	392
44	441
48	480
52	510
56	537
60	592
64	629
68	687
72	707
76	766
80	785
84	832

Table 1 - Collected Distance and Voltage Measurements

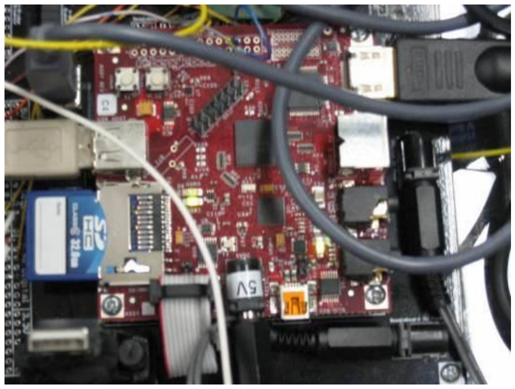
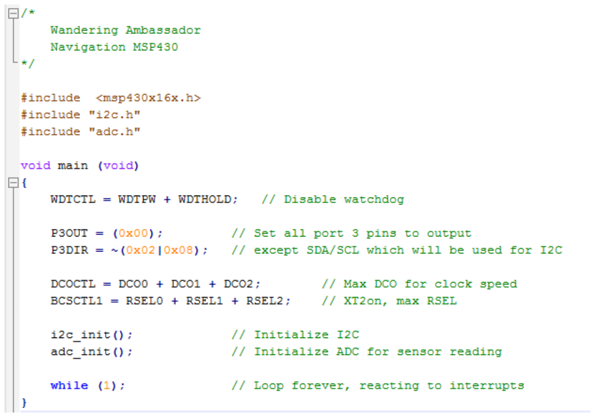


Figure 13 - Beagleboard

MSP430s

The code that existed for the MSP430s from prior teams proved to be unmanageable. It was poorly constructed, had inconsistent coding standards, and was not properly documented. The code was rewritten so that it worked properly and would be more maintainable in the future. Figure 14 is sample code for the MSP430s. Figure 15 is an image of the three MSP430s.



```
/*
    Wandering Ambassador
    Navigation MSP430
*/

#include  <msp430x16x.h>
#include "i2c.h"
#include "adc.h"

void main (void)
{
    WDTCTL = WDTPW + WDTHOLD;   // Disable watchdog

    P3OUT = (0x00);          // Set all port 3 pins to output
    P3DIR = ~(0x02|0x08);    // except SDA/SCL which will be used for I2C

    DCOCTL = DCO0 + DCO1 + DCO2;        // Max DCO for clock speed
    BCSCTL1 = RSEL0 + RSEL1 + RSEL2;    // XT2on, max RSEL

    i2c_init();              // Initialize I2C
    adc_init();              // Initialize ADC for sensor reading

    while (1);               // Loop forever, reacting to interrupts
}
```

Figure 14 - Sample Code for the MSP430s

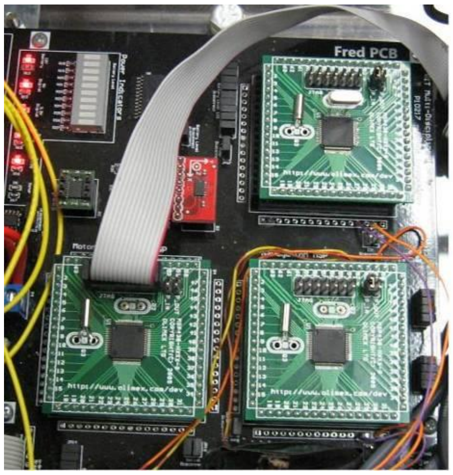
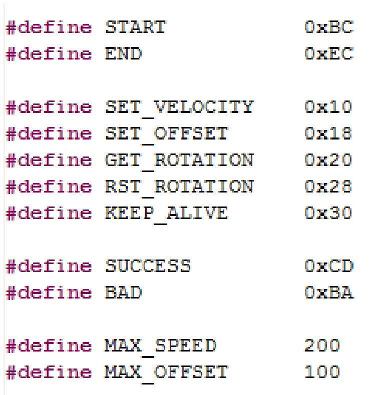


Figure 15 - MSP430s

Motor and Plant Controllers

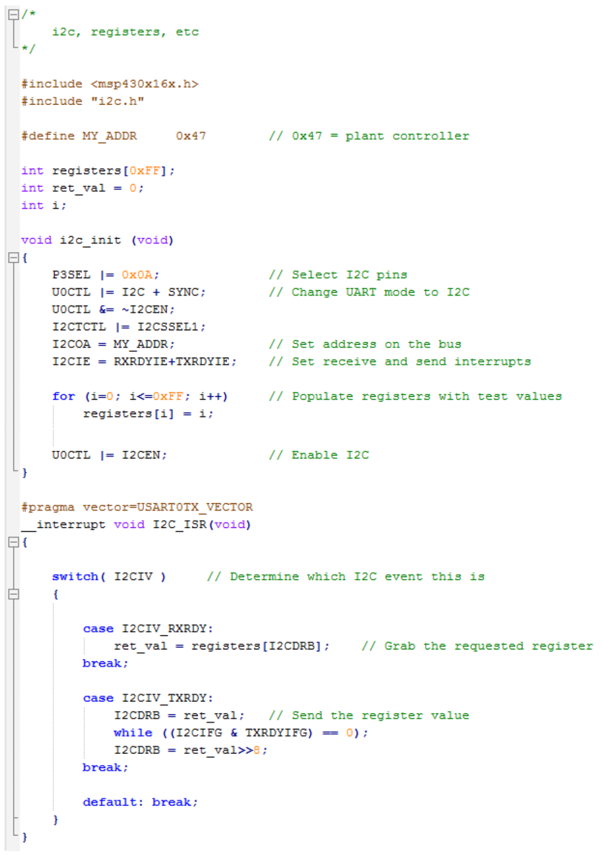
The old motor controller code had a complicated protocol that was not fully implemented. It also had bugs in the PWM code that caused an inconsistent duty cycle which performed less than optimally. The new controller code has proper PWM functionality, allowing for more consistent operation along with faster overall speed. The new protocol is simpler while maintaining all required functionality, allowing for easer implementation by the software engineers. Figure 16 is an example of the current protocol being implemented.



```
#define START           0xBC
#define END             0xEC

#define SET_VELOCITY    0x10
#define SET_OFFSET      0x18
#define GET_ROTATION    0x20
#define RST_ROTATION    0x28
#define KEEP_ALIVE      0x30

#define SUCCESS         0xCD
#define BAD             0xBA

#define MAX_SPEED       200
#define MAX_OFFSET      100
```

Figure 16 - Example of Current Protocol

The motor and plant controllers operate off the same base code. The functionality is compartmentalized to a great extent allowing for easy maintainability. The I2C portion handles the communication between the MSP430 and the Beagleboard, as well as the general register management. This is identical between the two controllers. The ADC portion has the same code base between the two controllers, which some changes for the different sensor requirements. For instance, the sonars cannot operate at the same time due to cross-talk, so the navigation ADC code cycles the power through the sonars when reading. Figure 17 is a sample of the base code for the motor and plant controllers.



```
/*
    i2c, registers, etc
*/

#include <msp430x16x.h>
#include "i2c.h"

#define MY_ADDR     0x47        // 0x47 = plant controller

int registers[0xFF];
int ret_val = 0;
int i;

void i2c_init (void)
{
    P3SEL |= 0x0A;              // Select I2C pins
    U0CTL |= I2C + SYNC;        // Change UART mode to I2C
    U0CTL &= ~I2CEN;
    I2CTCTL |= I2CSSEL1;
    I2COA = MY_ADDR;            // Set address on the bus
    I2CIE = RXRDYIE+TXRDYIE;    // Set receive and send interrupts

    for (i=0; i<=0xFF; i++)     // Populate registers with test values
        registers[i] = i;


    U0CTL |= I2CEN;             // Enable I2C
}

#pragma vector=USART0TX_VECTOR
__interrupt void I2C_ISR(void)
{

    switch( I2CIV )     // Determine which I2C event this is
    {

        case I2CIV_RXRDY:
            ret_val = registers[I2CDRB];    // Grab the requested register
        break;

        case I2CIV_TXRDY:
            I2CDRB = ret_val;   // Send the register value
            while ((I2CIFG & TXRDYIFG) == 0);
            I2CDRB = ret_val>>8;
        break;

        default: break;
    }
}
```

Figure 17 - Sample of Base Code for Motor and Plant Controllers

Alarm System Circuit

The alarm circuit wiring was updated from previous teams to include the LEDs and the pump that was connected to the physical deterrence. The schematic of the alarm circuit is located in Figure 18.

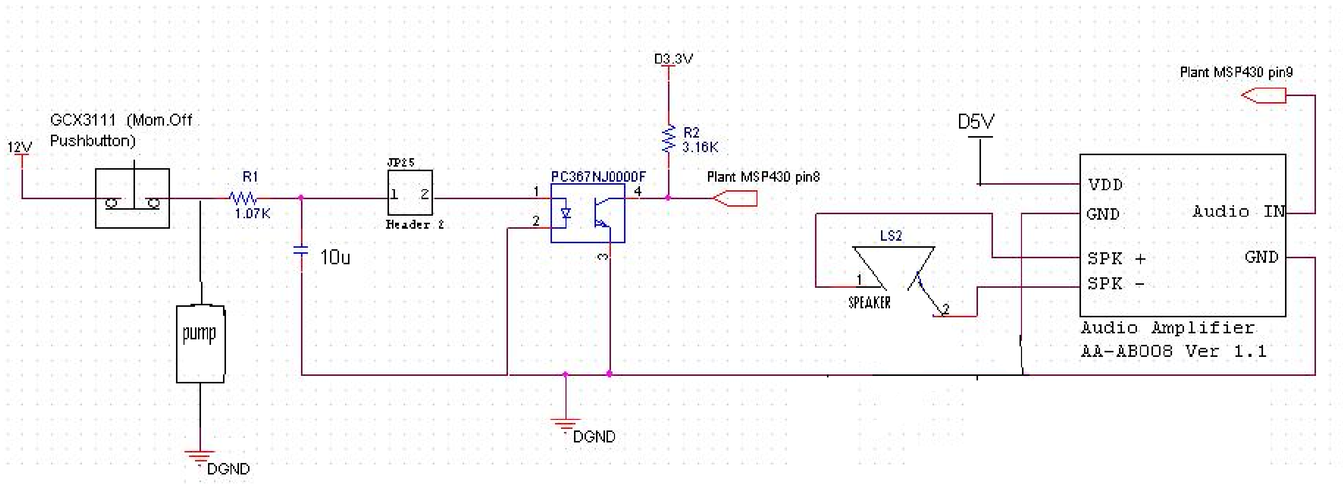


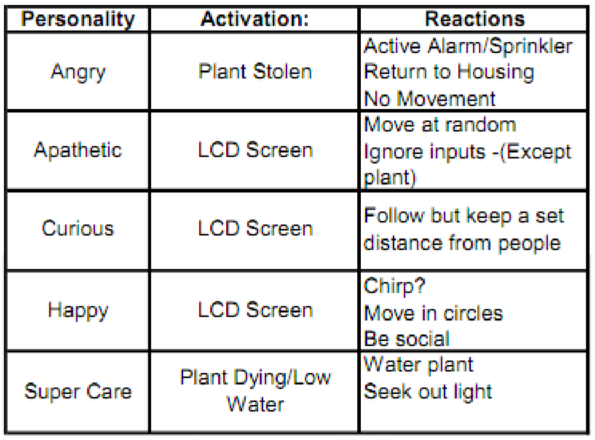
Figure 18 - Alarm System Circuit

SOFTWARE DESIGN

The software aspect of the Wandering Ambassador project was the most involved of the project. The initial requirements stated that the robot was to be able to navigate autonomously to a reasonable degree of efficiency while simultaneously caring for the potted plant. The navigation system was designed from the ground up to utilize the numerous sensors available on the robot.

Personalities

At the customer's request, the system was also to implement multiple personalities to be showcased via different wandering styles and movement habits. As a result, five different personalities were characterized. Figure 19 shows the personalities and their respective traits.



Personality	Activation:	Reactions
Angry	Plant Stolen	Active Alarm/Sprinkler Return to Housing No Movement
Apathetic	LCD Screen	Move at random Ignore inputs -(Except plant)
Curious	LCD Screen	Follow but keep a set distance from people
Happy	LCD Screen	Chirp? Move in circles Be social
Super Care	Plant Dying/Low Water	Water plant Seek out light

Figure 19 – Personality Chart

Each personality represents an intended state of the system. To simplify design, a diagnostic mode was implemented as a possible sixth personality to allow the robot to have a state in which developers could easily test specific sensors and functionalities.

Sensor Data

In order to efficiently navigate without potentially endangering itself or those around it, the robot was also required to continuously check and analyze its surroundings. To accomplish this, a series of interpreter classes were created, with each monitoring a subset of available sensors. Each interpreter was its own independent thread, and continuously pulled incoming data from various sensors, making real time data always available to the implementing personality classes that required it to make decisions.

Wandering and Navigation

Another independent thread was the implementing personality, which handled all decision making in terms of the robot's movement. All six implementing personalities inherit from an abstract personality class, and add to the thread that it is running. This works so that there is a default series of decisions each personality must make (i.e. which sensors to check and how to move) and each implementing personality is free to customize how to make them without altering existing code. The abstract class is extremely modular, as it includes numerous methods that allow the implementing personalities to check specific sensors and make specific movements, making it easy to piece together a decision engine in another personality. Figure 20 shows the basic decision tree used in the Apathetic wandering personality.

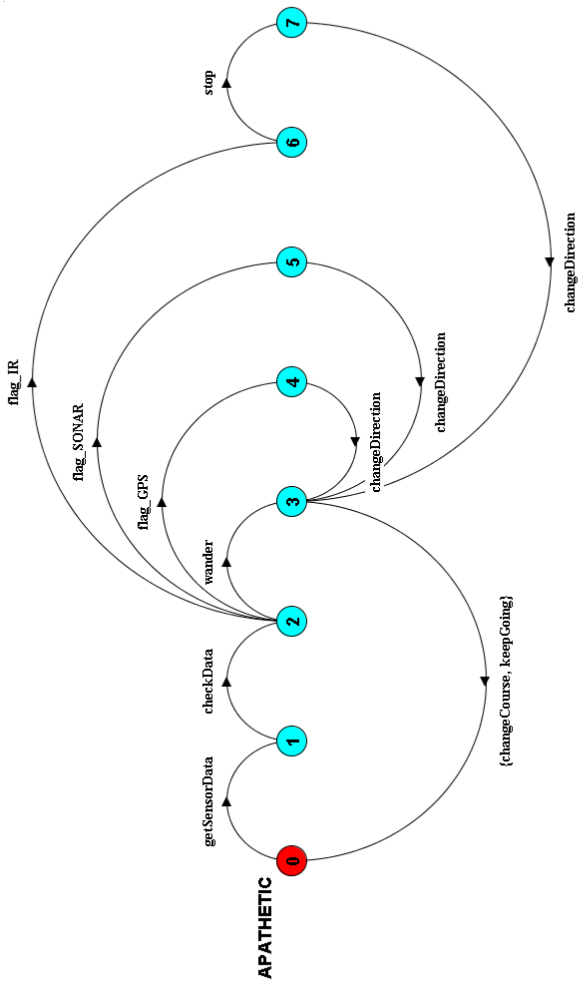


Figure 20 – Apathetic Decision Diagram

Plant Care

In addition to the interpreters, another thread exists to solely monitor the health of the plant. This thread samples the data provided by the plant-related sensors approximately every thirty seconds to ensure that all values are within acceptable thresholds. If the values breach these predefined thresholds, the class will take action to correct them.

System Design

Figure 21 shows the system wide design diagram that was used as the basis for the implementation. The main classes can be separated into two categories, logic classes and the sensor inputs and outputs. Individual boxes represent classes, and arrows connecting them show which classes are used by others.

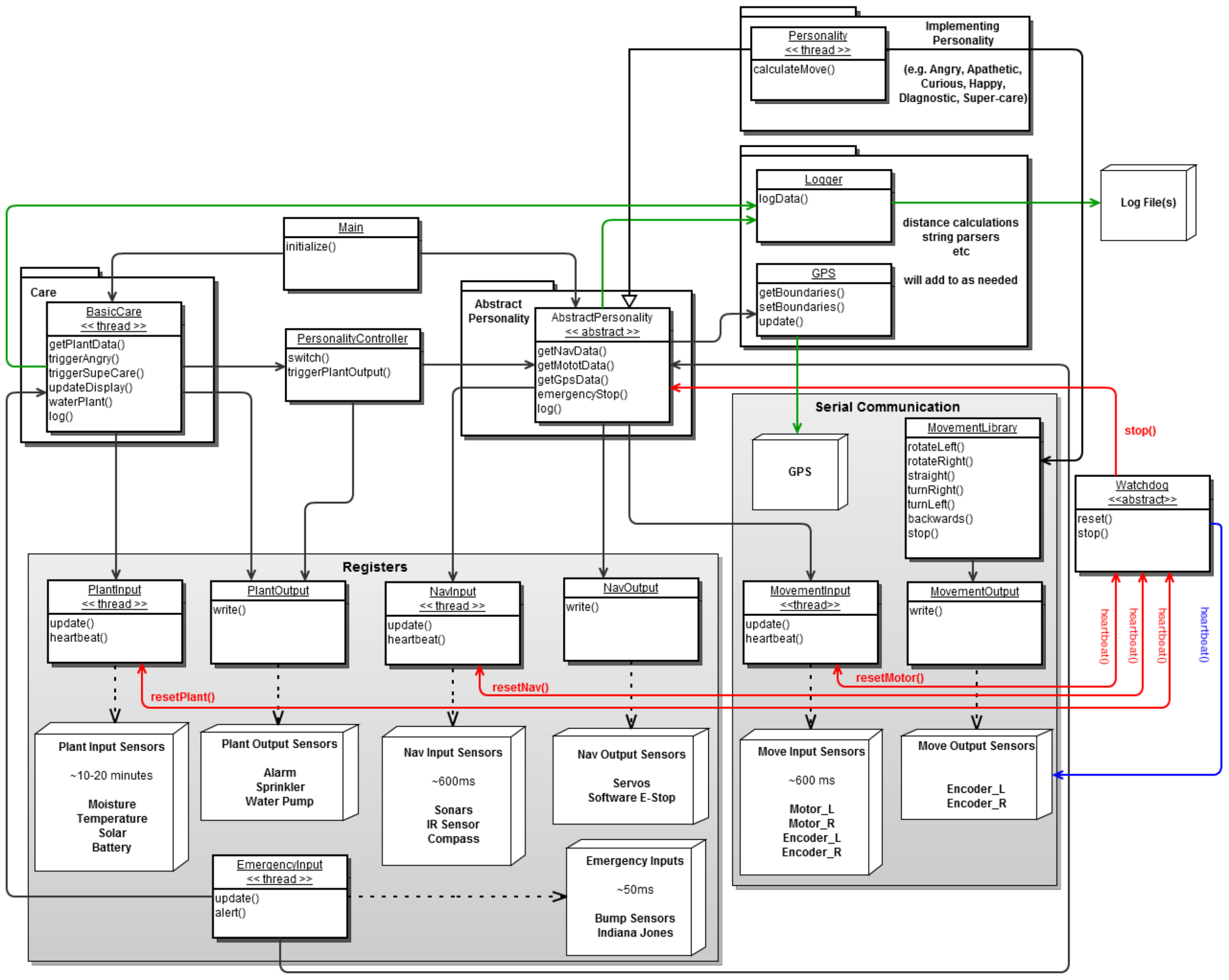


Figure 21 - UML Diagram

Logic Classes

There are eight important logic classes:

Main is an initializing class that starts the BasicCare thread and the Diagnostic personality thread.

PersonalityController will start and stop personality threads during runtime.

MovementLibrary contains a set of movement options that can be utilized by any implementing personality.

BasicCare is a thread that will check the data held in PlantInput. Based on that data it will water the plant, switch the personality, or do nothing.

AbstractPersonality is an abstract class that is meant to be overwritten by PersonalityMove.

Personality is a thread that represents the active personality of the robot. It checks the data stored in NavInput and based on the data it determines what movement should be taken. It then calls to the MovementLibrary to take that action.

Watchdog is a thread that checks to make sure that all the software and hardware interactions are still valid.

GPS receives input from the GPS, parses and translates it, and holds onto it until the implementing personality needs it.

Sensors Input/Output

PlantInput is a thread that continuously checks register values and holds onto the data those registers contain. It monitors the moisture sensor, the thermometers, the solar panels, and the water level sensor.

PlantOuput writes to the registers to indicate that an action should be taken. It can command the alarm, sprinkler, or water pump. The alarm command sets the alarm off. The sprinkler command sprays water from the sprinklers, or nozzles. The water pump command waters the plant.

NavInput is a thread that continuously checks register values and holds onto the data those registers contain. It monitors the sonars, GPS, and IR sensors.

NavOuput writes to the registers to indicate that an action should be taken. It can command the servos. The servo command rotates the sonars mounted on the servos.

EmergencyInput is a thread that continuously checks register values. If the data indicates that an action needs to be taken, it will call the corresponding method to take that action in either BasicCare or CalculateMove. It monitors the bump sensors and the "Indiana Jones" switch, and triggers the robot's responses accordingly.

RESULTS AND DISCUSSION

The mechanical, electrical, and computer engineering customer needs were met include: improving the sonar range and area of vision, testing the bumpers, adding a physical deterrence for the alarm, and allowing for additional plants to be carried.

Mechanical, electrical, and computer engineering customer needs that were not met were: increasing the adjustability of the plant mount, adding a switch to disengage the alarm, building a shelter for the Wandering Ambassador for the night, incorporating energy sockets for recharging the batteries within the shelter, and collecting and storing water for later use. Increasing the adjustability of the plant mount was not added due to budget and time constraints.

The main focus of the work for P11216 was to deliver autonomous motion of the robot. As such, the majority of the customer needs were based on Software Engineering. The completed customer needs were storing the sensor input in the registers, developing a virtual machine for development, connecting to the Wandering Ambassador through the internet, watering the plant when the moisture level is low, staying within a certain set of coordinates, not falling off of ledges, sensing people and not running into them, apathetic, angry and diagnostic personality modes, and being able to run each motorized wheel independently.

Software engineering customer needs that were not met or were changed were: QNX loaded onto the beagleboard, seeking out more intense light when the light currently available is not enough, seeking out other plants, monitoring the battery life charge, and the curious, friendly, super-sustain, and test personality modes. QNX was not loaded onto the beagleboard because the decision was made to use Angstrom instead. The test personality mode was combined with the diagnostic personality mode, which was completed. Similarly to the mechanical, electrical, and computer engineering customer needs, the other software customer needs that were not fulfilled, were not implemented due to time constraints.

CONCLUSIONS AND RECOMMENDATIONS

The customer needs that were of the highest importance were all completed. Given more time and a larger budget, all of the customer needs, even those of low importance, could have been completed. Other setbacks include the electrical components taking longer to implement than expected and poor understanding of the hierarchy that tasks need to be completed in at the start of the project. As the project advanced, the hierarchy was better implemented and time management improved.

While not all of the specified customer needs were met, the critical need to develop an autonomous navigation system was completed. Future iterations of this project should focus on the implementation of the unmet needs, as well as the addition of new personalities. With these improvements the Wandering Ambassador will become more robust and further its ability to raise awareness of self-sustaining energy.

REFERENCES

[1] Russell, Brian, Thomas Anderson, Vincent Baier, Daniel Chin, Eric Harty, Richard L. Reyes, Emile Lebrun, Manasi Manjrekar, and Matthew Padula. "P10215 / Home." *P10215: Robot Locomotion and Plant Platform*. 17 Sept. 2010. Web. 20 Apr. 2011. <http://edge.rit.edu/content/P10215/public/Home>.

[2] Alam, Maha, Corey Provencher, Alan Olson, Marcus Gwillim, and Nicolas Bouret. "P10216 / Home." *P10216: Robot Navigation Plant Platform*. 22 Apr. 2010. Web. 20 Apr. 2011. <http://edge.rit.edu/content/P10216/public/Home>.

[3] Arrigo, Pat, Wes Coleman, Aaron Zimmerman, Vernon Vantucci, Steven Guenther, and Adam Spirer. "P10217 / Home." *P10217: Robot Integration and Field Testing*. 19 May 2010. Web. 20 Apr. 2011. <http://edge.rit.edu/content/P10217/public/Home>.

[4] McAfee, Jeff, David Leeds, Katlyn Walsh, Alex Ford, and Steve Wright. "P10218 / Home." *P10218: Robot Applications*. 19 May 2010. Web. 20 Apr. 2011. <http://edge.rit.edu/content/P10218/public/Home>.

[5] Ackerman, Justin, Hersh Anand, Jonathan Notaro, Eric Tripp, Chris Hoerbelt, Gregory Mucks, and Anthony Poli. "P11215 / Home." *P11215: Wandering Campus Ambassador (part 4 of N)*. 21 Feb. 2011. Web. 20 Apr. 2011. <http://edge.rit.edu/content/P11215/public/Home>.

[6] DeBartolo, Elizabeth, George Slack, Andreas Savakis, and James Vallino. "P11216 Project Readiness Package." *P11216 Project Description*. 11 Feb. 2011. Web. 20 Apr. 2011. <http://edge.rit.edu/content/P11216/public/Project%20 Description>.

ACKNOWLEDGEMENTS

Professor George Slack, Dr. James Vallino, Dr. Mario Gomes, Dr. Mark Kempski, Dr. Wayne Walter, William Spath, Professor Scott Hawker, Dr. Ken Hsu, Stan Rickel, Daniel Phillips, Steve Kosciol, Rob Kraynik, P11215, RIT Provost's Learning Innovations Grant, KGCOE EE Department, Imagine RIT Staff, Christine Fisher, RIT, Amazon.com, Robot Shop, PepBoys, Lowe's, Newark.com, RIT machine shop, MP3Car.com, Cdiweb.com, McMaster Carr, Gro-Moore Farms