Swarming Robot Design, Construction and Software Implementation

Karl A. Stolleis
Kennedy Space Center / NASA
Master of Computer Science
Lab Intern – Center Innovation Fund
Date: 22/07/2014

Swarming Robot Design, Construction and Software Implementation

Karl A. Stolleis
University of New Mexico, New Mexico Spacegrant Fellow, Albuquerque, NM 87131

In this paper is presented an overview of the hardware design, construction overview, software design and software implementation for a small, low-cost robot to be used for swarming robot development. In addition to the work done on the robot, a full simulation of the robotic system was developed using Robot Operating System (ROS) and its associated simulation. The eventual use of the robots will be exploration of evolving behaviors via genetic algorithms and builds on the work done at the University of New Mexico Biological Computation Lab.

Nomenclature

ROS	=	Robot Operating System
Swarmie	=	Small Robot used for Swarm Robotics Research
iAnt	=	Robot developed at University of New Mexico for Swarm Robotics
Gazebo	=	Simulation Environment for Robotics Research
GA	=	Genetic Algorithm
GPS	=	Global Positioning System
IMU	=	Inertial Measurement Unit
CPU	=	Central Processing Unit
SRD	=	Software Requirements Document
KSC	=	Kennedy Space Center
UNM	=	University of New Mexico
ISRU	=	In Situ Resource Utilization
RASSOR2	=	Regolith Advanced Surface Systems Robot – 2nd Generation
COTS	=	Commercial Off the Shelf
ARM	=	Commercial Reduced Instruction Set Architecture for Microcontrollers
OS	=	Operating System
IEEE	=	Institute of Electrical and Electronics Engineers
IDE	=	Integrated Development Environment
SD	=	Secure Digital
SDHD	=	Secure Digital High Density
NMEA	=	National Marine Electronics Association
TTL	=	Transistor-Transistor Logic
RS232	=	Serial Communication Protocol
LTS	=	Long Term Support
KBit	=	Kilobit
MP	=	Mega-Pixel
GB	=	Gigabyte
CIF	=	Center Innovation Fund
QR Tag	=	Quick Response Tag Encoding System
x86	=	Common 32 or 64 bit CPU Architecture
CAD	=	Computer Aided Design

I. Introduction

The field of swarm robotics has come into its own, in the last few years, as a dedicated research topic within the robotics community. The Swarmie project was begun as a Kennedy Space Center (KSC) Center Innovation Fund Project for 2014 under the official title of "Evolving Power Management and Work Flow in Swarming Robots." The eventual use of the robots will be to explore the use of genetic algorithms to evolve the swarm behaviors as the robots set about a "central place foraging," task as described in research conducted at the University of New Mexico [9-15]. The first goal of the Swarmie project was to design and construct a new robot platform that maintains the small size and low-cost goals of the iAnt project at UNM. The second goal is to use Robot Operating System (ROS) to create the simulation and onboard software that will run the system and third is to adapt UNM's genetic algorithm to allow for adaptation of the autonomous operation of the robots. A secondary goal is to share as much onboard software as is possible with the RASSOR2 platform being developed at KSC's "SwampWorks." The ultimate goal of the project is to provide meaningful insight into the possibility of operating a swarm of robotic agents as a precursor to human exploration or alongside human exploration in an extra-planetary setting [1-8].

II. Swarmie Hardware Design

The Swarmie hardware design was begun early on using the following requirements as a foundation for the design. The cost of the robot must be kept as low as possible and the size of the robot must be kept small, both for safety and ease of operation. The robot was to be a four-wheel, differential drive platform with sensors to gather position data, range data to objects, at least one on-board camera for target detection, a way of establishing robot attitude, the ability to communicate over standard IEEE 802.11 wireless connection, an onboard computer capable of running the Linux operating system and the ability to run ROS "on top," of the Linux OS.

A. Differential Drive Platform

The UNM iAnt project was a model for the development of the Swarmie. The iAnt robot platform uses custom-cut acrylic pieces for the physical platform and the decision was made to adapt a COTS chassis to the Swarmie project. The chassis chosen was the Lynxmotion A4WD1v2. The chassis was larger than the iAnt but still deemed "small" in the spirit of the iAnt while allowing for larger motors, more battery capacity and fitment of a larger CPU board for more processing capabilities. The motors used on the chassis are 12 volt, gear-head motors with a 100:1 gear reduction ratio and an approximate speed of 65 RPM at 12V. The chassis uses wheels and tires adapted from scale-model, "monster trucks," interchangeable top and bottom plates and the ability to fit additional accessories not used in the Swarmie project. Figure 1 shows the bare chassis as supplied by the manufacturer, post-assembly.

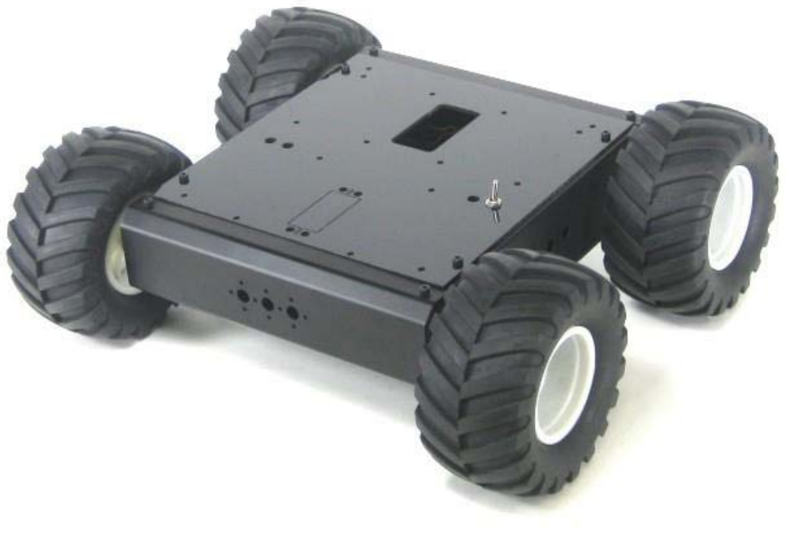


Figure 1 - Assembled Lynxmotion A4WD1 chassis.

B. Microcontroller and Sensors

The microcontroller chosen for motor control and sensor control is the Digilent ChipKit Max32 development board which uses a MicroChip PIC32 microprocessor. This processor has onboard power regulation, four available serial ports, well over 50 general purpose digital pins and a set of pins that can be configured for pulse-width modulation, necessary for motor operation. The board is programmed using a variant of the Arduino language using an integrated development environment (MPIDE) provided by the manufacturer. The PIC32 processor is a true 32-bit processor running at 80 MHz, uses a common 3.3 volt logic level and has 512K of onboard flash memory available for programming.

The motor controller, or half-bridge, was manufactured by Pololu Robotics and uses two ST Microelectronics VNH5019 motor drivers capable of driving motors from 5-24 volts and delivering 15 amps of current for a short period. The motor chosen, in addition to being 12V, have a stall-current of approximately 5A and with two motors being driven by each half-bridge there is an almost 33% operational safety margin. The half-bridges have a nominal logic level of 5V but are capable of operation with the 3.3V level used by the PIC32. A total of eight microcontroller pins are needed for the motor drivers with an additional two providing current load measurements. The current sensing capability is not implemented in this stage of the project. A custom library was written in C++ to allow for integration into the MPIDE/Arduino programming environment.

The range sensors chosen are the Devantech SRF05 ultrasonic rangefinders. Three SRF05's are mounted on the front of the chassis with one aligned along the robot's x-axis and one each placed both left and right at a 25° angle (plus and minus) from the x-axis. The rangefinders have an operational range of approximately five meters and produce a 55° cone from the sensor's center axis. The layout provides overlap between the center sensor and each of the two side sensors allowing for detection of objects in a total of eight different combinations. Given that sonic rangefinders are not directional, the additional location provided by the sensor overlap is important and useful in obstacle avoidance and mapping. The rangefinders utilize two digital pins on the PIC32, one for triggering the sensor and one for reading the echo time. The sensors require a 5V+ power supply but operate normally at the 3.3V logic level. As with most other sensors, a custom C++ library was adapted from the manufacturer to use in the MPIDE/Arduino environment.

Localization data is provided via a UBlox LEA-6S GPS sensing module that is mounted on a small development board. The board provides for connection to the UBlox unit over the PIC32 serial connection (UART1) and a custom library that allows for altering the data rate and structure of the output message was written. The unit uses a native 3.3V logic level and standard data is provided in the NMEA0183 standard sentences at an update rate of 5Hz. An active, external antenna is connected, via coaxial cable, to provide additional precision to the unit and allow for flexible mounting locations onboard the robot. The decision was made to use GPS, and sonar for that matter, despite the fact that robots operating in an extra-planetary environment will be unable to use these sensors. The field of robot localization, or position finding, is a broad research field and is well beyond the scope of this one-year project.

Attitude data is provided via a Pololu Robotics AltIMU-10 which contains a three-axis gyroscope, three-axis accelerometer, three-axis magnetometer and barometric pressure sensor. This IMU is capable of providing robot attitude, heading with reference to earth's magnetic frame (compass heading) and altitude data. The IMU is connected the PIC32 via the I2C bus and each sensor is addressed independently. Care was taken to mount the sensor as far away as practical from electro-magnetic sources and large metal objects in the chassis. The manufacturer provides an additional software library used to calibrate the sensor with regard to static sources of interference, such as mounting bolts or chassis metal.

Communication with the micro-controller is handled via USB connection with the main CPU. Communication is standard RS-232 protocol using serial ASCII characters to both provide sensor data and accept motor commands from the CPU.

Due to the micro-controller having an onboard voltage regulator, power is supplied via a switched connection to a two-cell lithium-polymer batter with a 7.4V nominal voltage and 1300mA-hour discharge rating. The board is capable of drawing power from the USB connection but the USB standard is limited to 0.5A current draw. Measurements of the microcontroller, with all sensors connected, showed 0.45A of current draw and the decision was made to provide a separate battery so as to not damage the CPU board USB port in case of current spikes caused by the motor drivers.

C. Main Computer / CPU and Associated Hardware

The main computer board used on the Swarmie is the open-source BeagleBoardxM-RevC designed in conjunction with Texas Instruments (TI) and capable of running a full Linux operating system on its ARM based processor. The board is approximately 3.75" square and provides four USB ports, an Ethernet and serial communication port and an external power connector. The CPU runs at a clock speed of 1GHz and uses 512K of onboard ram for program execution. Figure 2 shows some of the salient features of the board.

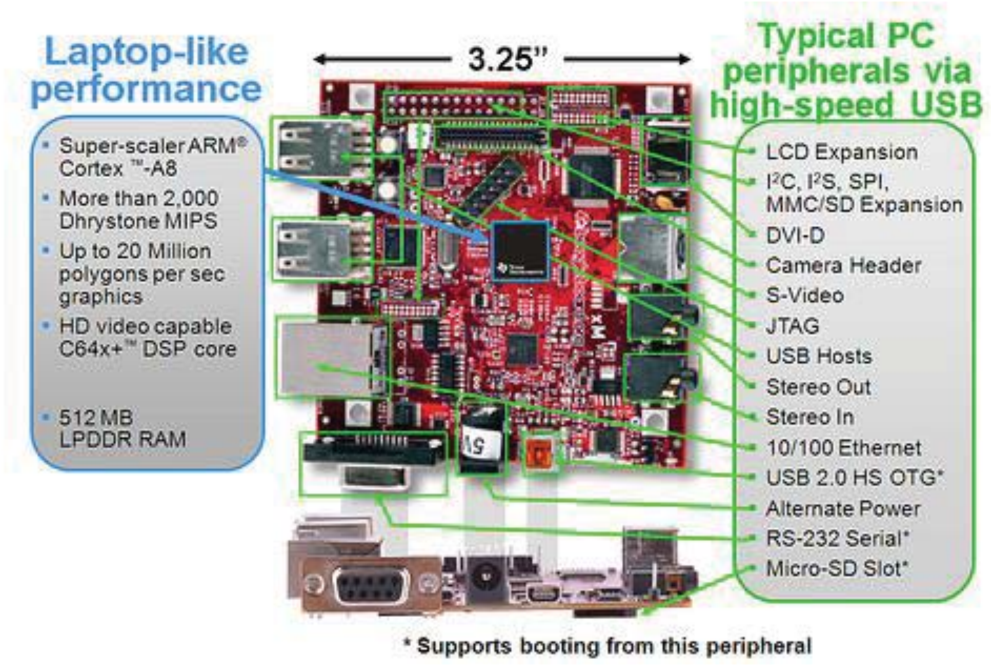

Laptop-like performance
• Super-scaler ARM® Cortex ™-A8
• More than 2,000 Dhrystone MIPS
• Up to 20 Million polygons per sec graphics
• HD video capable C64x+™ DSP core
• 512 MB LPDDR RAM
3.25"
Typical PC peripherals via high-speed USB
LCD Expansion
I²C, I²S, SPI, MMC/SD Expansion
DVI-D
Camera Header
S-Video
JTAG
USB Hosts
Stereo Out
Stereo In
10/100 Ethernet
USB 2.0 HS OTG*
Alternate Power
RS-232 Serial*
Micro-SD Slot*
* Supports booting from this peripheral

Figure 2 - BeagleBoard-xM board.

Wireless network connectivity is provided by a small USB "dongle" or wireless adapter as there is no provision on the main board for built in wireless communication. The adapter is capable of a 150Kbit/sec data rate which is common for wireless adapters of this form.

Target detection is provided by a small, Logitech USB web-camera with a 1.7 MP sensor. The camera is connected directly to the CPU, via USB and is the only data sensor that is not connected to the microcontroller.

The board does not have an on-board power regulator so power is provided via a switched, three-cell lithium-polymer battery with a nominal voltage of 11.1V and 1800mAh capacity. A DC/DC regulator translates the battery voltage to a constant 5V.

Onboard data storage, as well as the storage of the actual operating system, is handled via micro-SD card. The CPU can recognize SDHC cards and cards with a formatted size of 16GB are used for the project. The operating system occupies approximately 2GB of the card and the remaining space can be used in lieu of a traditional disk storage medium. More discussion of the operating system and onboard software will follow in the software section of this document.

One additional USB port is available, over the current configuration, which allows for use of a USB flash-storage device or additional USB camera if needed.

D. CAD Model / 3D-Printed Parts

Several of the sensors needed to have mounting hardware designed and printed using a 3-D, plastic filament printer. The mounts include the webcam, IMU, GPS antenna and a support bracket for an accessory cover that fits over the top of the chassis. The parts were created using a -3D CAD program, converted the files to a generic CAD format and then printed using a Makerbot Replicator2 3-D printer. The initial camera mount fit the camera such that it sat slightly above the ultrasonic sensors but further testing revealed this mount to be too low for proper target detection. A second mount, sitting higher on the chassis, was developed and both mounts are retained in the event that a second camera is utilized. The lid support was needed, in lieu of traditional threaded standoffs, to prevent interference between the support and the GPS module. Future models of the robot should have the microcontroller relocated slightly to allow the use of a normal standoff. Figure 3 shows examples of the parts as modeled.

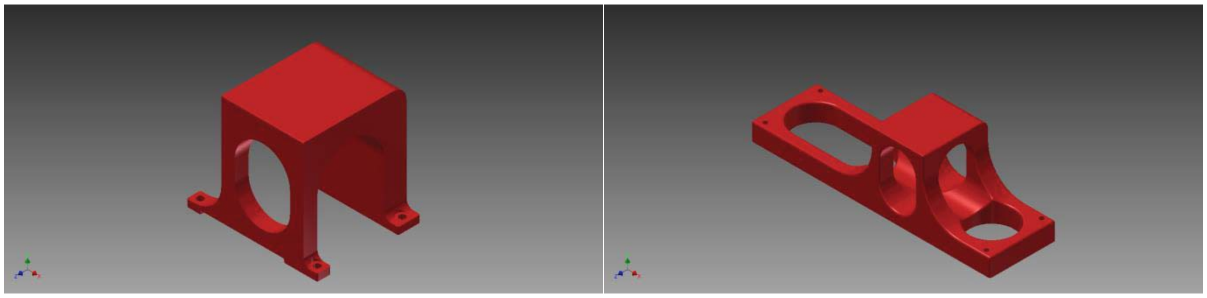
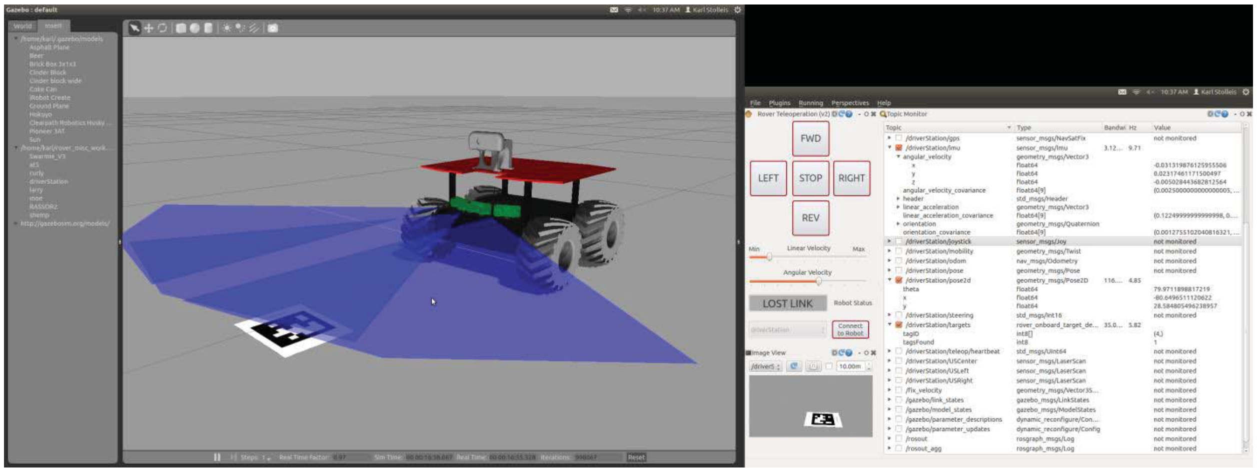


Figure 3 - 3D model of original camera mount (L) and top plate camera mount (R)

At the same time the sensor mounts were modeled, a simple CAD model was also created for the robot chassis and tire assemblies. These models were then converted to a 3-D "mesh" using a third-party software program and allowed for an accurate model to be created for use in the Gazebo simulation environment. More information about the model and sim will be presented in a later section of this document. Figure 4 shows the completed model in the simulation environment with the associated operation module.



Topic	Type	Bandwi	Hz	Value
/driverStation/gps	sensor_msgs/NavSatFix			not monitored
/driverStation/imu	sensor_msgs/Imu	3.12...	9.71	
angular_velocity	geometry_msgs/Vector3			
x	float64			-0.0313198761259555506
y	float64			0.02317461171500497
z	float64			-0.005028443682812564
angular_velocity_covariance	float64[9]			(0.0025000000000000005, ...
header	std_msgs/Header			
linear_acceleration	geometry_msgs/Vector3			
linear_acceleration_covariance	float64[9]			(0.12249999999999998, 0....
orientation	geometry_msgs/Quaternion			
orientation_covariance	float64[9]			(0.0012755102040816321, ...
/driverStation/joystick	sensor_msgs/Joy			not monitored
/driverStation/mobility	geometry_msgs/Twist			not monitored
/driverStation/odom	nav_msgs/Odometry			not monitored
/driverStation/pose	geometry_msgs/Pose			not monitored
/driverStation/pose2d	geometry_msgs/Pose2D	116....	4.85	
theta	float64			79.9711898817219
x	float64			-80.6496511120622
y	float64			28.584805496238957
/driverStation/steering	std_msgs/Int16			not monitored
/driverStation/targets	rover_onboard_target_de...	35.0...	5.82	
tagID	int8[]			(4,)
tagsFound	int8			1
/driverStation/teleop/heartbeat	std_msgs/UInt64			not monitored
/driverStation/USCenter	sensor_msgs/LaserScan			not monitored
/driverStation/USLeft	sensor_msgs/LaserScan			not monitored
/driverStation/USRight	sensor_msgs/LaserScan			not monitored
/fix_velocity	geometry_msgs/Vector3S...			not monitored
/gazebo/link_states	gazebo_msgs/LinkStates			not monitored
/gazebo/model_states	gazebo_msgs/ModelStates			not monitored
/gazebo/parameter_descriptions	dynamic_reconfigure/Con...			not monitored
/gazebo/parameter_updates	dynamic_reconfigure/Config			not monitored
/rosout	rosgraph_msgs/Log			not monitored
/rosout_agg	rosgraph_msgs/Log			not monitored

Figure 4 - Robot model in Gazebo simulation with simulated April Tag and RQT driver station user interface.

All other parts were mounted using threaded aluminum standoffs and nylon washers to isolate the boards from potential short. Appendix A contains a complete parts list and cost schedule for the robot.

III. Hardware Construction

The actual construction of the robots took place over an approximately two-week period. Four robots in total were built. Despite the use of COTS products the robot still requires several hours of assembly time, per robot. Further discussion of the hardware construction is detailed in a supporting document, created by another team member and will not be discussed in depth as part of this paper. [16]

IV. Onboard Software Design and Implementation

A. Software Design / ROS

The first major hurdle to the software design was prototyping small pieces of the software to determine fitness, feasibility and issues. The Robot Operating System, or ROS, was chosen as the system that would underlie the individual processes and given that the team had little experience with ROS, some initial prototyping was a must. A second question needing to be answered was whether the hardware chosen had complete implementation of ROS available or if the ROS system for the ARM CPU had limited functionality. ROS has several hundred pre-built software modules available for installation and use by anyone developing a robot. These modules are mostly contributed by other robotic development groups and are available under common open-source software agreements and can be modified and used at will [17].

In concert with the prototyping work, the team completed an internal 30% and 60% design review of the onboard software. Using the ROS model means developing small executable programs, similar in spirit to Unix OS "processes" and allowing these processes to communicate via a common messaging bus. The ROS model of inter-process communication follows the traditional "publisher subscriber" design pattern but extends the model by creating a public messaging bus, with associated networking capabilities, that allow for processes to communicate either within a robot or share data across a network. This powerful capability was one of the driving decisions behind the choice of ROS for the project.

ROS also contains a set of standard message types that further streamline the inter-process communication. These data types include traditional computer language "primitive types," but also many types that are suited for robot specific applications such as IMU data, localization data and image data.

The ROS modules are written using either the Python or C++ programming languages, which adds flexibility to developer decisions. Any image processing must be done using C++ and due to performance constraints, on the Swarmie CPU, the decision was made early on to utilize the C++ capabilities for the entirety of the project.

The last, and not insignificant feature of ROS, is inclusion of a modular user interface that can be used to monitor data topics and control robots or features on the robots. This system is authored in C++ or Python and makes use of the QT framework for building actual user interface modules. The availability of pre-built modules prevented the team from spending valuable time developing a custom interface but when necessary the ability to develop custom modules is being utilized.

As with most design decisions there are tradeoffs, and there were items balanced in this case also. The use of ROS, and the team's lack of experience, were considered risks due to the need to learn the system and the additional computational resources needed to support ROS. Final consideration was made based on the fact that ROS would allow faster development, in the time constrained environment of the project. The capability for fast development, in concert with the associated simulation environment, finally convinced the team members to use ROS as the base operating system.

B. Linux

ROS is not a true operating system, in the classic sense, and as such requires an underlying operating system for the CPU and standard hardware. For this project the choice of OS was driven by the overarching need for ROS. The ROS framework is optimized for the Linux OS and in most cases the development and testing is done on the Ubuntu distribution of ROS. Given the processing constraints presented by the BeagleBoard-xM, the choice was made to utilize the so-called "server" image of Ubuntu 12.04LTS on the Swarmies and use the full desktop version of Ubuntu 12.04LTS on developer stations, driver stations and other support computers. The major difference between the two versions are the server version does not come with many of the supporting features that the full-desktop does, most notably a graphical user interface or "desktop." The interactions with the Swarmie onboard system are then limited to remote access from a command line terminal.

The advantages of a full operating system are numerous and beyond the scope of this document but it should be obvious that Linux and ROS, working in concert with each other, provide a strong framework for rapid robotic software development.

C. Software Messaging and Node Design

To fit the typical ROS architecture the first step was to set up a framework of processes or "nodes," that would be used onboard and identify the ROS message types or "topics," that would be passed between nodes. The list of nodes, and associated message types presented here are not comprehensive and subject to change due to the rapid development model used during this project. The node architecture, as originally designed, is shown in Figure 5 and includes nodes that are common between the Swarmie and RASSOR2 system as well as the nodes that are unique to each individual platform.

One of the overarching goals of the node configuration was to create nodes that performed one basic function rather than wrapping multiple functions up into one process. Whenever possible the messages passed between nodes are kept as one way communications and an attempt to minimize "circular," connections was made. At the time this document was written only one custom message type, another ROS capability, had been developed. The one particular custom message was used to communicate the number of targets located by the robot and the information encoded in each target. Additional information will be presented on the targets in a later section of this paper.

Additional nodes shown in the diagram are for communication to the robot via a "driver" or teleoperation station. The original iAnt robots are fully autonomous but it was decided that provision for manual control should be added to the Swarmies as an extended capability. Additional design and execution of these driver nodes are beyond the scope of this document but it should be noted that they are ROS nodes and share many of the design features and capabilities presented by ROS. This document is also not intended to be a comprehensive discussion of the individual nodes or their unique capabilities.

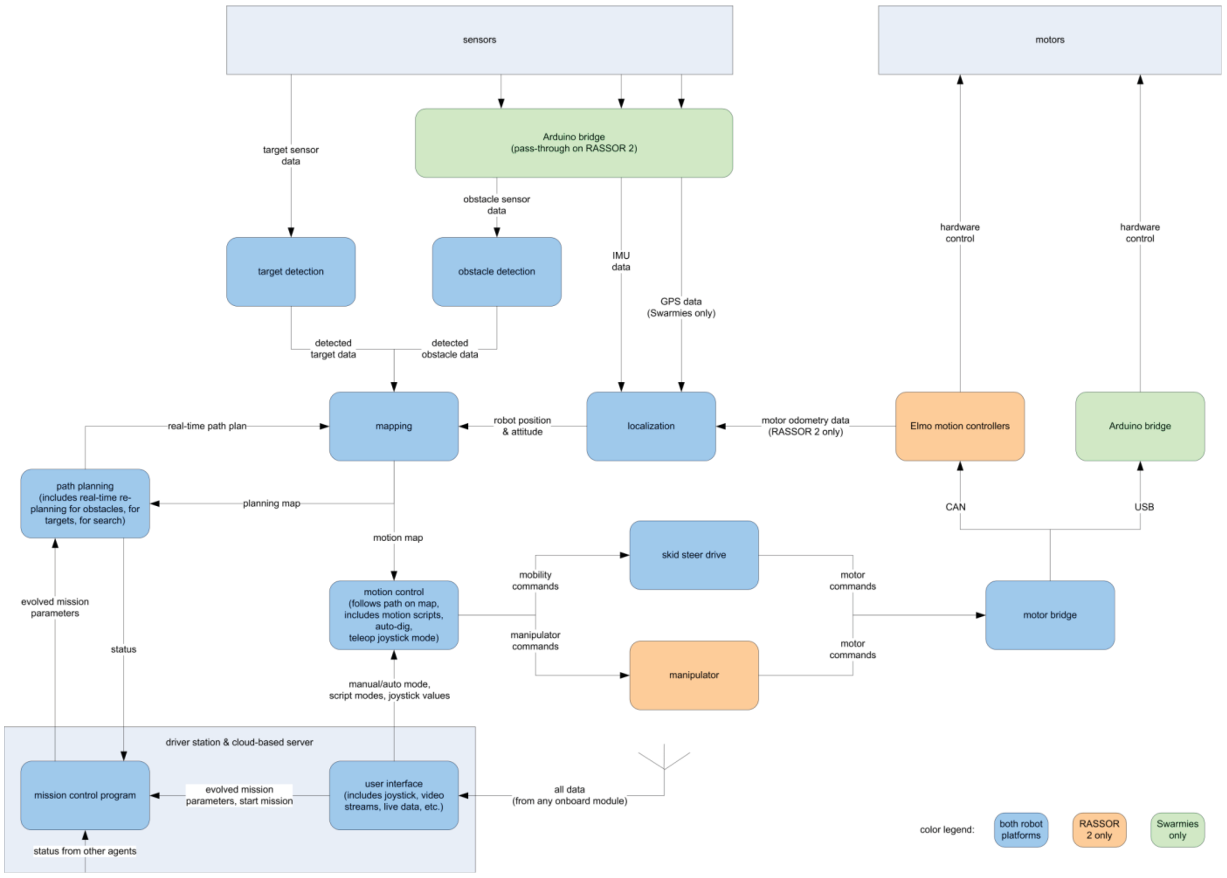

sensors
motors
Arduino bridge (pass-through on RASSOR 2)
target sensor data
obstacle sensor data
IMU data
GPS data (Swarmies only)
hardware control
hardware control
target detection
obstacle detection
detected target data
detected obstacle data
mapping
robot position & attitude
localization
motor odometry data (RASSOR 2 only)
Elmo motion controllers
Arduino bridge
real-time path plan
path planning (includes real-time re-planning for obstacles, for targets, for search)
planning map
motion map
CAN
USB
skid steer drive
mobility commands
motor commands
motion control (follows path on map, includes motion scripts, auto-dig, teleop joystick mode)
motor bridge
evolved mission parameters
status
manipulator commands
motor commands
manipulator
manual/auto mode, script modes, joystick values
driver station & cloud-based server
mission control program
evolved mission parameters, start mission
user interface (includes joystick, video streams, live data, etc.)
all data (from any onboard module)
status from other agents
color legend:
both robot platforms
RASSOR 2 only
Swarmies only

Figure 5 - Proposed software architecture from 90% design review.

D. April Tags

The robots ultimate, autonomous behavior is based on so-called "central place foraging," which is a term developed to explain the tasks performed by ants while out gathering food. Physical collection or gathering of objects by robots is a difficult sub-task in robotics and again beyond the scope of this project. A suitable analog for physical targets is needed. In the iAnt project QR tags were utilized so a robot could find a tag, make note of the number encoded in the tag, and not double count a tag previously found. The system allows for the "collection" of the resource in a temporal manner similar to ants finding and collecting a finite food source in the environment. For the Swarmie project the decision was made to utilize April Tags, a system developed by the April Robotics Lab at the University of Michigan [19]. The April tag system creates families of tags that, at first glance, appear similar to QR tags. The major difference is that April tags encode far less information than QR tags but with the added benefit of more reliable tag recognition and the ability to extract distance and position information from the tag. Discussion has been made with regard to using April tags for a visual docking alignment system for robots upon return to an operating base. An example of various tag families is shown in Figure 6.

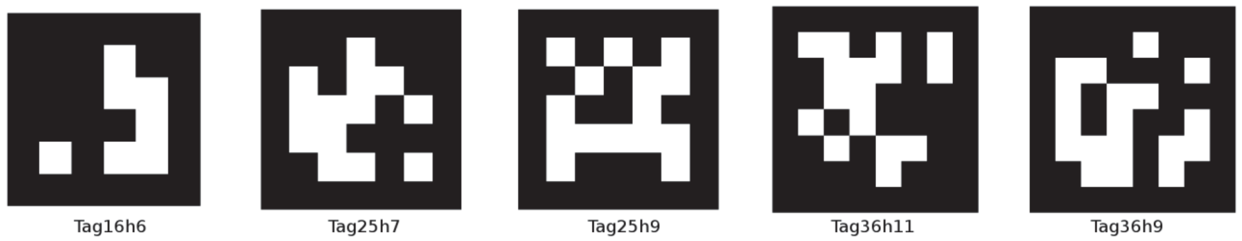


Figure 6 - Examples of April tags from various families. The family name contains the number of bits encoded in a tag and minimum Hamming distance used for tag decoding.

The source code for reading April tags was compiled from a modified, open-source version, for both x86 and ARM architecture. After implementation the tag reading capabilities were tested and prompted a re-location of the onboard camera to a higher point, to improve reading capabilities. The changes are highlighted in the previous section of this document and the accompanying hardware assembly manual. Testing revealed that too shallow an angle, between the camera central axis and plane of the tag, could inhibit proper tag reading. Also of note is the case where the tag cannot be read where the tag occupies more than 75% of the visual field. No ultimate cause was determined but it was demonstrated that shrinking the physical size of the tags by 66% increased tag reading reliability to virtually 100%. It should also be noted that during testing it was discovered that the entire tag must be in the camera view for it to be read. Even a slight obscuration of the tag margin resulted in a read failure.

Of continuing concern to the team, with regard to target finding, is the percentage of CPU time occupied by the camera and the tag reading module. The two modules account for approximately 75% of total CPU time available but currently no improved solution is under development. It is a well-known issue in computing that image processing is a difficult computational task and has potential to utilize more than its fair share of CPU time.

E. Evolving Behaviors and Adding Power Management

The final stage of robot development is still underway at the time this document was authored. I would refer readers to the papers referencing the iAnt project for a full discussion of evolving autonomous robot behavior via genetic algorithm, in the context of the central-place foraging task. That said, the work being done on the Swarmie project is adapting the iAnt algorithms as a foundation for Swarmie behavior. In addition, work is being done to include two additional parameters with regard to allowing the robot to decide the proper time to return to base and recharge versus proper time to rejoin the search task after some period of charging [20-26]. Future papers will be presented with regard to this additional behavior.

V. Robot Simulation and Testing

An additional capability provided by the ROS framework is its accompanying simulation environment, Gazebo. An advantage of Gazebo [18] is the ability to import a CAD model and associate inertial characteristics of the robot and achieve realistic operation via the simulation's physics engine. The various onboard sensors are simulated via ROS pre-built modules for GPS, sonar, camera, differential drive and IMU. The sensor simulators have the capability to incorporate noise into the data stream and set up and test various configurations of initial parameters. Of note is the modeling of the robot camera, which is able to accurately detect April tags, which are imported into the simulation as standard graphic files. Refer to Figure 4 for a screenshot showing a Swarmie model and associated ROS user interface for interacting with the robot. One of the keys to successful creation of a simulation model was utilizing as many of the standard ROS message types for the particular sensors. This returns to the need to develop, using ROS, without heavy dependence on custom generated message types.

The ability to have a complete and accurate simulation for the project is paramount and completing work on the simulation is a good, internal milestone. The evolution of behavior, via the genetic algorithm, utilizes the simulation environment as a way of qualifying evolved parameters and allowing for new behaviors to develop without running in real-time.

The simulation also has the benefit of using the exact same modules, sans one, used on the physical robots. The one module that is unique to the physical platform is the one node that interacts with the microcontroller and it should be quite obvious why this module is not necessary in the simulation.

Testing of the onboard and driver software is also being undertaken in the Gazebo environment.

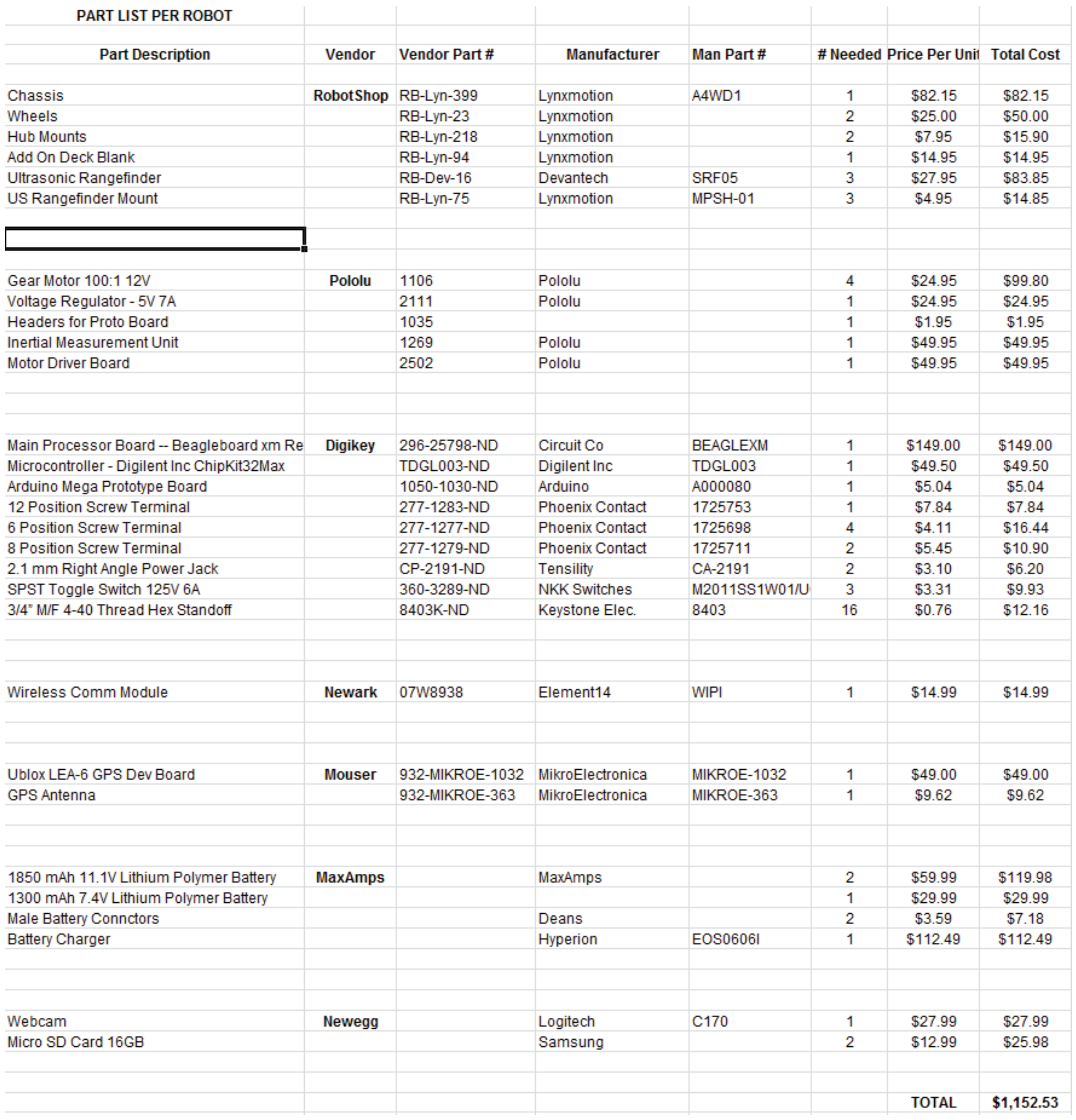
VI. Conclusion

Presented in this paper is an overview of the hardware and software design used in the Swarmie project. The project is scheduled to be complete in March of 2015 and it is hoped by the team members that the research will be beneficial with regard to extra-planetary mining and ISRU.

VII. Future Work

The material presented in this paper is an overview of the Swarmie hardware and software design. The author has already proposed taking the next step, with regard to hardware development and establishing a modular robot carrier and standard, similar to the CubeSat standard, that would apply to small, terrestrial robots. Continuing research will also need to be done with regard to localization in an environment where GPS or compass headings are infeasible or unavailable.

Appendix A



PART LIST PER ROBOT

Part Description	Vendor	Vendor Part #	Manufacturer	Man Part #	# Needed	Price Per Unit	Total Cost
Chassis	RobotShop	RB-Lyn-399	Lynxmotion	A4WD1	1	$82.15	$82.15
Wheels		RB-Lyn-23	Lynxmotion		2	$25.00	$50.00
Hub Mounts		RB-Lyn-218	Lynxmotion		2	$7.95	$15.90
Add On Deck Blank		RB-Lyn-94	Lynxmotion		1	$14.95	$14.95
Ultrasonic Rangefinder		RB-Dev-16	Devantech	SRF05	3	$27.95	$83.85
US Rangefinder Mount		RB-Lyn-75	Lynxmotion	MPSH-01	3	$4.95	$14.85
Gear Motor 100:1 12V	Pololu	1106	Pololu		4	$24.95	$99.80
Voltage Regulator - 5V 7A		2111	Pololu		1	$24.95	$24.95
Headers for Proto Board		1035			1	$1.95	$1.95
Inertial Measurement Unit		1269	Pololu		1	$49.95	$49.95
Motor Driver Board		2502	Pololu		1	$49.95	$49.95
Main Processor Board -- Beagleboard xm Re	Digikey	296-25798-ND	Circuit Co	BEAGLEXM	1	$149.00	$149.00
Microcontroller - Digilent Inc ChipKit32Max		TDGL003-ND	Digilent Inc	TDGL003	1	$49.50	$49.50
Arduino Mega Prototype Board		1050-1030-ND	Arduino	A000080	1	$5.04	$5.04
12 Position Screw Terminal		277-1283-ND	Phoenix Contact	1725753	1	$7.84	$7.84
6 Position Screw Terminal		277-1277-ND	Phoenix Contact	1725698	4	$4.11	$16.44
8 Position Screw Terminal		277-1279-ND	Phoenix Contact	1725711	2	$5.45	$10.90
2.1 mm Right Angle Power Jack		CP-2191-ND	Tensility	CA-2191	2	$3.10	$6.20
SPST Toggle Switch 125V 6A		360-3289-ND	NKK Switches	M2011SS1W01/U	3	$3.31	$9.93
3/4" M/F 4-40 Thread Hex Standoff		8403K-ND	Keystone Elec.	8403	16	$0.76	$12.16
Wireless Comm Module	Newark	07W8938	Element14	WIPI	1	$14.99	$14.99
Ublox LEA-6 GPS Dev Board	Mouser	932-MIKROE-1032	MikroElectronica	MIKROE-1032	1	$49.00	$49.00
GPS Antenna		932-MIKROE-363	MikroElectronica	MIKROE-363	1	$9.62	$9.62
1850 mAh 11.1V Lithium Polymer Battery	MaxAmps		MaxAmps		2	$59.99	$119.98
1300 mAh 7.4V Lithium Polymer Battery					1	$29.99	$29.99
Male Battery Connctors			Deans		2	$3.59	$7.18
Battery Charger			Hyperion	EOS0606I	1	$112.49	$112.49
Webcam	Newegg		Logitech	C170	1	$27.99	$27.99
Micro SD Card 16GB			Samsung		2	$12.99	$25.98
						TOTAL	$1,152.53

Acknowledgments

The author would like to thank the team members who contributed to the design and implementation : Cheryle Mako (primary investigator), Kurt Leucht (software lead), Lien Moore, Matt Nugent and Gilbert Montague. Thanks also to UNM Assistant Professor Melanie Moses and PhD Joshua Hecker for assistance in the design and research. This work is funded via KSC Center Innovation Fund and New Mexico Spacegrant Foundation.

References

[1] National Aeronautics and Space Administration, *“Robotics, Tele-Robotics and Autonomous Systems Roadmap, Technology Area 04,”* April, 2012

[2] Washington, Richard, et al. *"Autonomous rovers for Mars exploration."* Aerospace Conference, 1999. Proceedings. 1999 IEEE. Vol. 1. IEEE, 1999.

[3] Gat, Erann, et al. *"Behavior control for robotic exploration of planetary surfaces."* Robotics and Automation, IEEE Transactions on 10.4 (1994): 490- 503.

[4] Brooks, Rodney A., et al. *"Lunar base construction robots."* Intelligent Robots and Systems '90.'Towards a New Frontier of Applications', Proceedings. IROS'90. IEEE International Workshop on. IEEE, 1990.

[5] Bellingham, James G., and Kanna Rajan. *"Robotics in remote and hostile environments."* Science 318.5853 (2007): 1098-1102.

[6] Kennedy, Fred, et al. *"A Study of Cooperative Control of Self-Assembling Robots in Space with Experimental Validation."* (2009).

[7] Pavone, Marco, et al. *"Spacecraft Autonomy Challenges for Next Generation Space Missions."*

[8] Singh, Sanjiv, et al. *“Technology for autonomous space systems.”* Carnegie Mellon University, the Robotics Institute, 2000.

[9] K. Stolleis, *“AntBot Version One Owner's Manual,”* June 2011

[10] *“The iAnt Project,”* https://sites.google.com/site/unmantbot/home, retrieved 2/2014

[11] N. Bezzo, et.al,. *“Exploiting Heterogenous Robotic Systems in Cooperative Missions,”* 2011 (under review)

[12] Hecker, Joshua P., et al. *"Formica ex machina: ant swarm foraging from physical to virtual and back again."* Swarm Intelligence. Springer Berlin Heidelberg, 2012. 252-259.

[13] Hecker, Joshua, et al. *"Evolving Error Tolerance in Biologically-Inspired iAnt Robots."* Advances in Artificial Life, ECAL. Vol. 12. 2013.

[14] Moses, Melanie E., et al. *"Beyond Pheromones: An integrated ant-inspired approach to swarm robotics."* (under review)

[15] Hecker, Joshua Peter, and Melanie E. Moses. *"An evolutionary approach for robust adaptation of robot behavior to sensor error."* Proceeding of the fifteenth annual conference companion on Genetic and evolutionary computation conference companion. ACM, 2013.

[16] Gilbert Montague, *"Swarmie Hardware Assembly Manual,"* KSC Internal Document, 7/2014

[17] *"Robot Operating System,"* http://www.ros.org/about-ros/, retrieved 7/2014

[18] *"Gazebo Robot Simulator,"* http://gazebosim.org/, retrieved 7/2014

[19] Olson, Edwin. *"AprilTag: A robust and flexible visual fiducial system*." Robotics and Automation (ICRA), 2011 IEEE International Conference on. IEEE, 2011.

[20] O'Hara, Keith J., et al. *"Autopower: Toward energy-aware software systems for distributed mobile robots."* Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on. IEEE, 2006.

[21] Kelly, Ian, Owen Holland, and Chris Melhuish. *"Slugbot: A robotic predator in the natural world."* Proceedings of the Fifth International Symposium on Artificial Life and Robotics for Human Welfare and Artificial Liferobotics. 2000.

[22] Michaud, François, and Etienne Robichaud. *"Sharing charging stations for long-term activity of autonomous robots."* Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on. Vol. 3. IEEE, 2002.

[23] Floreano, Dario, and Francesco Mondada. *"Evolution of homing navigation in a real mobile robot."* Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on 26.3 (1996): 396-407.

[24] Arvin, Farshad, Khairulmizam Samsudin, and Abdul Rahman Ramli. *"Swarm robots long term autonomy using moveable charger."*Future Computer and Communication, 2009. ICFCC 2009. International Conference on. IEEE, 2009.

[25] Zebrowski, Pawel, and Richard T. Vaughan. *"Recharging robot teams: A tanker approach."* Advanced Robotics, 2005. ICAR'05. Proceedings., 12th International Conference on. IEEE, 2005.

[26] Howard, Andrew, Maja J. Mataric, and Gaurav S. Sukhatme. *"An incremental deployment algorithm for mobile robot teams."* Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on. Vol. 3. IEEE, 2002.