Tokka: The Autonomous Turtle (May 2006)

Cressel Anderson, Michael Lewallen, Jack Miller, Michael Pisarskiy,
Jason Watts, Raymond Whiting, James Wu, Members of IEEE

Abstract- In recent years, companies and the US military have been researching into autonomous robots. The purpose of these robots is to remove the human factor in certain situations that could potentially hazardous. These robots can perform their designed task with little or no human interaction thus, keeping humans out of harms way. A team of senior engineers at the University of North Carolina at Charlotte was presented with the challenge to design an autonomous and amphibious robot. This paper outlines their efforts.

Keywords- turtle, autonomous, wireless, amphibious, robotic, embedded operating system

I. INTRODUCTION:

There are many robots that exist today to maneuver on land or water. However, very few robots have been designed to adapt to changes between the two environments or operate without any assistance from a human when unforeseen obstacles get in the way [1]. For these reasons, a great deal of research has been devoted towards developing an autonomous robot that is amphibious. Oceanographers have shown an interest in developing such robots to be used as autonomous underwater vehicles (AUV) to help explore the Earth's oceans. These AUVs are designed to go into deep areas of the ocean where man is unable to explore [2]. The US Department of Defense has also awarded grants toward the research and design of such robots to be used for warfare, space missions, and oceanic studies [3].

The project, codenamed "Tokka," is a turtle shaped autonomous robot of amphibious design. While Tokka was not designed for underwater applications, it uses many components commonly found in AUVs. Tokka is equipped with an on-board camera that allows it to take any photo desired by the user. It is controlled wirelessly with embedded components and uses embedded Linux to manage the control of all the internal components. A set of two paddle wheels was designed to allow Tokka to efficiently maneuver on both land and on the surface of water. The most critical part of the project was forming a structure constructed by rapid prototyping that was able to float in water and form a water tight seal to prevent water seepage from damaging the internal components.

In this paper, a detailed description of Tokka's design and its abilities is presented. A review of previously designed robots and controlling concepts is included for comparison purposes. Additionally, information about Tokka's components that help guide and maneuver it through a variety of environments is included.

II. PREVIOUS REASEARCH

The designing of robots that are capable of operating on both land and water has only been studied closely over the past few years. Perhaps the most challenging issue that these robots face is the ability to maneuver efficiently with no

external assistance. Hence, a great deal of the research for these robots has been focused on designing components and systems to help it maneuver in a different environments.

Before any of these robots were even designed, a great deal of study was devoted toward designing an efficient system that would allow the robot to maneuver itself in a wide variety of conditions. In depth research of such controlling concepts stared back in the 1980s as artificial intelligence (AI) was becoming popular. Prior to this time, robots and other such devices used algorithmic intensive (ALG) methods to help the robot make decisions. AI offers the robot a sense of reasoning to solve problems and the ability to learn to maneuver any unforeseen obstacles. The biggest advantage ALG offers is a series of predefined steps to solve certain problems that are easy to code. For these reasons, it may be desirable for the robot to be coded using both methods [4].

Different control architectures have been developed as another method to control these robots. This type of system can be constructed with hierarchical architecture, hierarchical architecture, subsumption architecture, or hybrid architecture. The hierarchical architecture offers simple top – down approach for control, but it is not as flexible which may require the whole architecture to be re-written when changes are needed. Hierarchical architecture offers a parallel approach to control to allow for more flexibility and has a low communication overhead. However, the communication between the devices in the robot can be very intensive and may have conflicts [5].

The third method, subsumption architecture, is a layered controlled architecture that consists of components working in parallel without a high-level supervisor. This implies that there is no global data structure. Subsumption is a very flexible and robust architecture, but can cause difficulties with synchronization of components. The last architecture, hybrid architecture, is a mixture of all of the three previously mentioned architectures. While the system has the combined advantages of these architectures, it is extremely difficult to design and implement [5].

Research has also been conducted on methods that use conditional sequencing logic (CSL) and reactive action packaging. Both are different forms of AI that have been developed for use in such robots and are focused on areas that have been unexplored by man. CSL is designed for use on linear applications such as level areas while RAP tries different programmed procedures for maneuvering challenging terrain. Such methods could be used in AUVs as well as vehicles designed for exploration of space [6]. Similar logic has been developed to help these vehicles repair themselves in the event of a malfunction [7]. Also, these methods could also be used with a system that is designed to learn about its environment by detecting different levels of light [8].

After a method has been chosen to control the robot, the exterior for the robot needs to be designed. While the body of the robot can take a variety of shapes, the most critical part is designing components that will allow it to move smoothly across a surface. One such suggestion is the creation of fins to be mounted on the side of the robot. Namio Kato has designed different types of fins that are similar to that of a fish for various underwater applications [9] [10]. His designs could be modified so a robot would have the capabilities to walk on land and swim on the surface of water.

Another design that could be used for such robots is a design that has very thin legs that are similar to that of a water spider. The robot designed by Toshio Fukuda was created to swim in pipes and search for leaks. This robot is a small, light weight robot that is shaped like a water beetle. It also has the ability to make sharp turns as needed and has limited maneuverability on land [11].

One of the best designs for our application are the set of curved flippers that are on the robot AQUA designed by students for different Canadian universities. These curved flippers allow for smooth maneuvering on land, on the surface of water, and underwater. The robot has no problems when transitioning from walking on the beach to swimming in the sea. This robot is guided with visual recognition and by receiving commands wirelessly from its user. Additionally, the robot has several of the same characteristics as Tokka such as it ability to take pictures, autonomous operation, and communicate wirelessly to the operator. For a robot to have the ability to swim on the surface of water, it must have the ability to float in water. The reverse is true if a robot is to swim underwater. AQUA has the ability to float on water, but it has the ability to sink by using compartments inside of the robot designed to store and release water [12].

Another design that was recently created by at team at Carnegie Mellon University is in the shape of a large water spider. This design allows the structure to walk on the surface of water with today's modern hovering technology which also makes it possible for the robot to have limited maneuverability on land [13].

III. TOKKA'S COMPONETS

Several of Tokka's components had to be purchased for it to have the ability to function properly. These components were selected based on certain requirements such as power consumptions and data IO. All components used are individually described below.

A. GPS

The GPS unit selected differed from the one that was actually used due to budget and supply. The unit used was the Sirf III. This unit was completely self contained and included a passive antenna. Additionally, the Sirf III had the NEMA-0183 communication protocol allowing for easy reading of location.

B. Camera

In the area of autonomous devices, having a real time sensor is critical. The most useful and dynamic sensor is the optical sensor. An optical sensor will allow the turtle to dynamically react to changes in its environment like tidal changes and fallen trees as well as aid in data gathering. The specifications of the camera that was use for Tokka are listed below.

Camera Candidate:
Creative Camera: Instant Web Cam
Still image capture: Up to 640 x 480 pixels
Video capture: Up to 640 x 480 pixels
Color Quality: 16 bit
Frame rate: Up to 30 frames per second
Power Rating: Under 5V and 500 mA
Communication: USB

Table 1: Summary of camera's specifications.

On top of all these features, this camera was a great choice because of its compact size. After removing the camera's housing, the camera was able to fit into matchbox sized compartment.

Due to cost constraints, professional cameras like a Photonfocus Hurricane 40

were not a viable option. However, a non-professional camera like the Creative Instant Cam could be used at one-fifth the cost and will provide the functionality to meet the project requirements. Additionally, the Creative camera is easier to implement with sample Linux drivers available on the web.

Figure 1: Camera used for Tokka.

C. TS-7200 Embedded Board

The TS-7200 Single Board Computer (SBC), produced by Technologic Systems, will become the heart and brain of the robotic turtle. This SBC runs on a 200MHz ARM9, 32-bit, processor. The ARM, Advanced RISC Machine, architecture is ideal for embedded applications due to its power-efficient design. A TS-7200 SBC has a standard power consumption of 2Watts, 5Volts at 0.4Amps. Embedded boards built with an x86 processor, AMD or Pentium can consume up to 4.5Watts, 5Volts at 0.9Amps. This low power feature will allow the processor to generate less heat, and operate in a fan-less environment. Since the embedded board will be enclosed in a water sealed infrastructure, a processor generating less heat would be ideal.

The ARM9 processor of the TS-7200 SBC also offers a MMU, Memory Management Unit, which supports the Linux OS, Operating System. The embedded Linux OS has been chosen as the development environment for the robotic turtle. The Linux OS offers a free and stable environment for application development. This environment will allow the programmers to work closely with the embedded hardware, creating more optimized code. Code optimization will allow the applications being developed to consume less system resources and offer increased functionality.

However, there are some limitations to the TS-7200 SBC. One of these limitations is the lack of COM ports for serial interfacing. But this problem can be easily avoided by adding a daughter board to the TS-7200's PC/104 expansion ports. This daughter board, TS-SER4, has four additional COM ports, bringing the total serial ports up to six. The other limitation of the TS-7200 is its inability to effectively interface with peripheral devices that can output +5 Volts. Driving a voltage greater than +3.3 Volts, to any of the input ports of the TS-7200, can damage the TS-7200 board. An opto-oscillator can be used bring the down the voltage from +5 Volts to +3 Volts, but this can create unnecessary delays for time critical applications. The last limitation of the TS-7200 is that there is no pulse width modulation, PWM. The lack of PWM limits the amount of motors the TS-7200 can control.

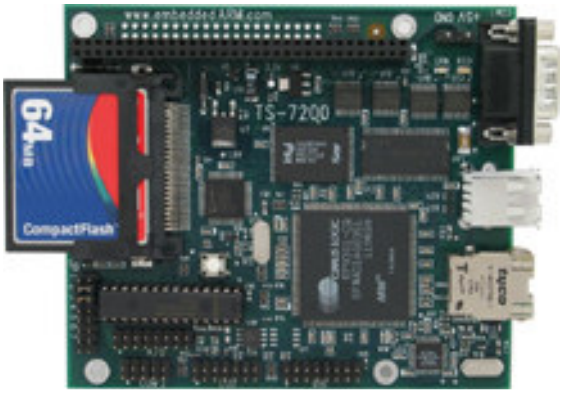

Figure 2: TS-7200 embedded ARM.

D. SKP M16C

To reduce the load placed on the main TS-7200 board, a small Renesas SKP M16C embedded board was added to control the motors and display the compass readings. This board features a small LCD screen, a 12 MHz processor, and various debugging techniques. The board can be easily programmed with c for a wide variety of applications.

Figure 3: Renesas 16C26 SKP board.

E. Motors

The motor selection for this project was a difficult one due to the need for precise controlled, torque, light weight and inexpensive. The motors used were a good compromise. They are light weight and inexpensive but did not provide the level of control desired. Not much information was obtained on these motors. However, they are DC brush motors rated at 6V. They came with controllers with Hall Effect sensors. However they were not precise enough for the intended application.

F. Motor Controllers

The Motors for this project were supplied with two different controllers (HB2 and HB4). HB4 controllers are attached to the back of the motors and the HB2 controllers are separate. Either controller could be used for the project. Originally, the HB4 was chosen for its inclusion on the motors. However, these controllers would burn-up during power up from a large initial current drain. Thus, the HB2 controllers were use for their large current specifications and robustness. Their, schematic is included with this paper.

G. Wireless Communications

The Linksys WET54G wireless G Ethernet bridge (figure 4) was chosen due to its fast transmission ability, simple connection (CAT 5) and easy of setup. The bridge's intention was to operate only when information was available for transmission. As compared to other wireless units, the Linksys bridge would save power from faster transmission resulting in less operation time. However, the actual implementation called for this unit to be continually operated. This will cause severe power drain from the batteries.

Figure 4: Linksys wireless bridge.

H. Batteries

Two 9.6V NiCd batteries were used to power Tokka's internal components. This type of battery was chosen for its durability, long-life, and its ability to be reused several times since it is rechargeable. One battery is used to power the motors that are used to control the wheels while another is used to power the embedded board, wireless receiver, and camera. A voltage regulator circuit is attached to each battery to ensure the

correct voltage is received for each component.

I. Expansion Board and Wiring

Tokka has many wired internal components. To help keep these wires organized, a hub-like board and wiring harnesses was used to ensure a tidy, easily-debugged, and reliable design. The GPS, compass unit, RS-232 level shifting circuitry, wiring harness connectors, and power connections resided on this board.

J. Power Distribution

Tokka's sensitive electronic components required power supply levels of 5Volts. To isolate these components from the noise of the motors, two isolated regulation circuits were implemented. The two highest current devices were separated resulting in the grouping of the wireless bridge, GPS, and compass on one regulator. The TS-7200 and the Renesas boards were paired on the remaining regulator circuit.

IV. TOKKA'S IMPLEMENTATION

Tokka was divided into two separate entities: hardware and software. This separation was to make the implementation easier and segregated. Each division implementation is discussed below.

A. Software:

Introduction:

Software for the amphibious surveillance robot, Tokka, has been developed on two different environments. One environment is the Technological Systems' TS-7200 single board computer, loaded with a Linux 2.4 kernel. The other, environment, is the Renesas M16C/62P microcontroller.

In order to offset some of the limitations of the TS-7200, to effectively interface and control the motors and compass, the Renesas microcontroller became the replacement navigation unit. The TS-7200, however, is still the main processing unit, handling the wireless communication, video processing, and communications with the GPS unit.

TS-7200 Setup:

Before any major software development can begin, a new kernel and file-system had to be setup for the TS-7200. Although the default configuration for the TS-7200 is suitable for most software development, rebuilding a new kernel and file-system will offer a more robust development environment.

In order to reconfigure the TS-7200 with a new kernel and file-system, a personal computer utilizing the Linux operating system must be used. The recommended Linux distribution is Debian or Ubuntu; this depends entirely upon the user's knowledge and familiarity with Linux.

Kernel Setup:

The Linux kernel configured for this project is a Linux 2.4 kernel. The kernel source and cross-compiler can be found on Technologic System's website, [14].

For this project the kernel has been patched and configured to support USB camera devices and Video4Linux. The USB and Video4Linux support were built as modules; this will help keep the final kernel image small. The Linux kernel will mount the modules whenever a recognizable camera device is attached to the USB port of the TS-7200, and un-mount the module when the camera is not in use.

Development Environment:

The development environment used for this project is Debian. This development

environment can be found on Technologic System's website, [14].

A 1GB Kingston compact flash card, formated with the ext2 file-system, was used to support the Debian development environment. The Debian development environment offers an onboard GCC compiler and a very robust package management system for finding and installing third party software.

Configuring RedBoot:

RedBoot is a complete bootstrap environment for embedded systems. Upon boot-up, RedBoot is instructed to load the Linux kernel from the onboard flash, and instructs the Linux kernel to mount the root file-system.

RedBoot can load a kernel image in one of three ways, through serial console, a TFTP server, or from the onboard flash. The default script loads the kernel image from the onboard flash.

For this project the TFTP server method was used to load a new kernel onto the TS-7200, thus erasing the default kernel provided by Technologic Systems. A standalone personal workstation with internet access was used as the TFTP server. A TFTP server is essentially a simpler version of the File Transfer Protocol; it uses the User Datagram Protocol and provides no security features. The default boot-script for the TS-7200 was also altered to mount the compact flash card as the root file-system, rather than the onboard JFFS2 file-system.

Script Setup:

A shell script is a small program that is read by the command interpreter of the operation system. Shell scripts are generally used for things like file manipulation, like renaming all the files in a directory; information gathering, like printing diagnostic information to a terminal; and program execution. Our project used it for the latter, program execution. This allowed the turtle to be operational at power up with out any user interference. This was done by putting in the startup commands for the camera, commutation software, and navigation software into the /etc/init.d/inittab file and specified them to run in /etc/init.d/tokka. After the normal boot procedure completed the startup commands would then be hit and ran.

GPS Unit & Navigation:

The Sirf III GPS unit that was utilized in the design served the purpose of navigation means. The main program would communicate via serial communication port with the GPS unit, get the reading of the GPS unit for current location, and use that to compute the heading of the turtle.

The GPS unit had the ability to communicate using the standard NEMA-0183 communication protocol, which utilizes standard serial communication means but regulates the format of the transmitted data.

The code designed to receive the GPS reading parsed the Longitude and Latitude out of the received string. When these two essential components were extracted, the computation of the heading is performed using standard vector algebra.

Navigation Unit:

The navigation unit for Tokka consists of the Renesas M16C/62P microcontroller, a digital compass with up to two degrees of accuracy, and two DC brush motors connected to a HB4 motor-controller.

The digital compass is connected to the Renesas microcontroller through two digital input/output, DIO, ports and a hardware interrupt line. Port 1_1 of the Renesas microcontroller connects to the

data line of the digital compass. This port is configured to receive and store incoming data into a buffer. Port 1_3 of the Renesas microcontroller connects to the P/C pin of the digital compass. This port is configured to output a 1 or a 0. If port 1_3 outputs a 1, the compass enters standby mode. If port 1_3 outputs a 0, the compass becomes active. Port 1_5 or hardware interrupt line of the Renesas microcontroller is connected to the clock output of the digital compass. The digital compass is designed to output data at every rising edge of its onboard clock. This clock also drives the interrupt line of the Renesas microcontroller. At the falling edge of each clock cycle an interrupt is generated, and the interrupt handler reads the data store in Port 1_1 and stores the data into a buffer. The data received from the digital compass is in binary format, a total of 16-bits. The first 7-bits of the data can be ignored, but the last 9-bits of the data must be stored because that contains the actual data.

Example of Compass Output:
XXXXXXX 000001010 = 10 Degrees
First 7 bits are ignored
Bits 8-15 contain actual data

The HB4 motor module can interface with up to two DC motors. Each motor has its own enable and direction port. The HB4 module is connected to the Renesas microcontroller through two digital input/output, DIO, ports and two timer output ports. Port 0_5 and port 0_7 connects to the direction ports of the HB4 module. In order to make the motors turn or reverse the HB4 module must receive four different 2-bit combinations. The combinations and directions can be seen below.

Port 0_5	*Port 0_7*	*Direction*
0	*0*	*Right*
0	*1*	*Reverse*
1	*0*	*Forward*
1	*1*	*Left*

Table 2: Pin combination to give Tokka directional commands.

Port 7_4 and port 7_6 connects to the enable port of the HB4 module. Both ports are timer outputs and configured for PWM. The DC motors' speed and ability to stop depends on the PWM waveform output. The speed and the timer register configuration can be seen below.

Speed	*Timer Register Config.*
FAST	*0x508B*
MED	*0x658B*
SLOW	*0x708B*
STOP	*0xFE8B*

Table 3: Tokka's basic navigational functions.

Camera Implementation:

Using a well known and popular manufacturer of a webcam was key in having a successful video solution for this project.

The Creative Instant Cam is supported by the spca5xx Video for Linux (v4l) driver [15], which provides support for webcams and digital cameras based on the spca5xx range of chips manufactured by SunPlus Sonix Z-star Vimicro Conexant Etoms and Transvision.

The driver comes in two flavors: one for x86 platform and one for the embedded platforms.

The driver was in development stages for the embedded platforms. The out of the box install of the driver did not work. Simple modification to the way the code read in the image solved the issue. The code implemented memory mapped way of reading the image from the camera, which

is hard to support on an embedded system with such low RAM (32 Mb). The best solution was to change the read() method to use the standard read function, as if you were to read a device driver.
This resolved the issue of memory overflow and ballooning that was present with the initial code setup.

After these issues were resolved the spcaView [15] was installed on the remote and the controlling laptop. This allowed for us to remotely view the live feed from the turtle also, the program had the functionality to take snapshots of the current view. At the completion of the project, there was nothing implemented in the design of the code to do any image detection like object recognition, this is an option for future work and research.

Overall Operations:

The higher layer software, or artificial intelligence, of Tokka can be broken up into two parts. The TS-7200 controls the communications, video and data processing, while the Renesas microcontroller handles the movement.

The TS-7200 is designed to wait for a connection from a remote client, or base. The socket program listens on port 2222 and waits for a connection. Once the connection has been established, a string of bits is sent from the client to the TS-7200 board, the string of bits contains the coordinates of the destination location. The TS-7200 board caches the data and parses the information. Then the TS-7200 starts to poll the GPS for the current coordinates. Once the current coordinates have been received, the TS-7200 calculates degrees and distance to the defined destination. The degrees are then sent to the Renesas microcontroller via asynchronous serial communication. Once the degrees have been received by the Renesas microcontroller, the compass is polled for its current location. After receiving the degrees from the compass, the Renesas microcontroller calculates the degrees difference to the final destination and turns the motors. These steps are then repeated until Tokka is pointing to the right direction, and moves towards the destination location.

While the robot is moving, the TS-7200 will continue to poll the GPS for its current location and calculate the distance. Once the destination location has been reached, TS-7200 sends a stop command to the Renesas microcontroller and initializes the camera. The Renesas microcontroller stops and turns the motors 360 degrees, this is done to capture a full view of the surrounding location.

After capturing the surrounding location, the Renesas microcontroller turns the motors towards the originating location and returns home. The TS-7200 will continue to poll the GPS until the starting location has been reached. After reaching the starting location, a stop command will then be sent to the Renesas microcontroller.

B. Hardware:

Introduction:

The hardware implementation of this project had to simple parts. Part one dealt with a scale model of the project. The second part was the construction of the actual end product. Both parts are elaborated below.

Scale Model:

Before the end product could be constructed several questions had to be answered. One such question was the size and buoyancy of the shell. Therefore, a scale model was constructed out of K'NEXs®. This model provided insightful knowledge into the end product. First, the overall size could be determined. Then,

experiments were preformed to provide the weight and buoyancy of the shell. This proved useful because the shell was #D modeled as explained below.

Construction:

The team decided to implement an acrylonitrile butadiene styrene (ABS) frame for Tokka's chassis. The parts were produced using stereolithography rapid prototyping technology. The finished model and breakout can be seen in Figure XX. The design allowed ample space for selected circuitry and provided an indigenous appearance. The paddle wheels were chosen to maximize the performance characteristics of both land and water propulsion.

Figure 5: Tokka assembly.

Before the pieces were built, the water sealing method was evaluated through the construction of the test stand seen in Figure 6. By using an o-ring properly sized for the shaft diameter and applying pressure to maintain the seal, Tokka's waterproof integrity was verified. Additionally, it was necessary to seal the outside ABS pieces with acrylic paint due to porousness caused by the prototyping equipment.

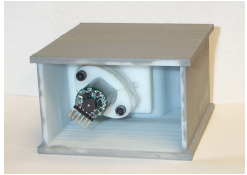

Figure 6: Test design for water proofing.

V. PROJECT OUTLINE

After all the parts were chosen, a plan of how they would be integrated was devised. The team decided the best recourse would be to split the build into two separate entities. Thus, both entities could be executed simultaneously resulting in better efficiency. One entity deals with the hardware build of the project and the other with the software development of the project. However, the team's original plan changed due to uncontrollable factors. Therefore implementations of both entities took different paths than expected and both are elaborated further below.

A. Hardware:

Due to the environmental operating conditions of the autonomous vehicle, the housing and chassis must above all be impervious to water seepage. It also must provide a reliable platform for the electrical system and for the components necessary for locomotion. This will be accomplished through a variety of innovative hardware combinations. The project planning Gantt chart Figure a shows the entire development sequence of the hardware construction of the unit.

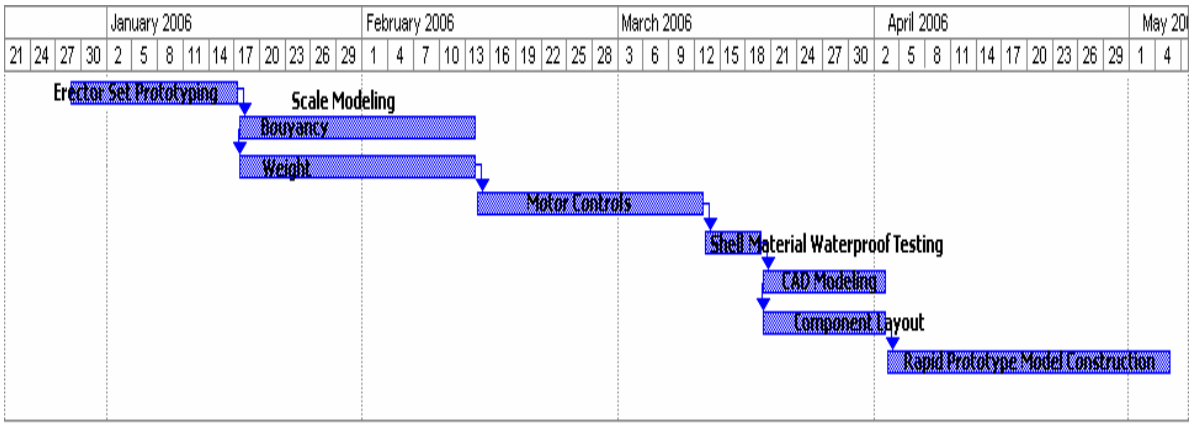

Figure 7: Gantt chart of hardware schedule.

B. Software:

The software development for the robotic turtle was segmented into three main sections or goals. From designing device drivers to artificial intelligence, each section focuses on a specific software task. This breaks up the robot software into smaller and more manageable parts. Programmers did not have to worry about higher layer applications when designing lower level device drivers. This method allowed each section to be divided among the software development team, which increased productivity. The project planning Gantt chart Figure d shows the entire development sequence of the software construction of the unit.

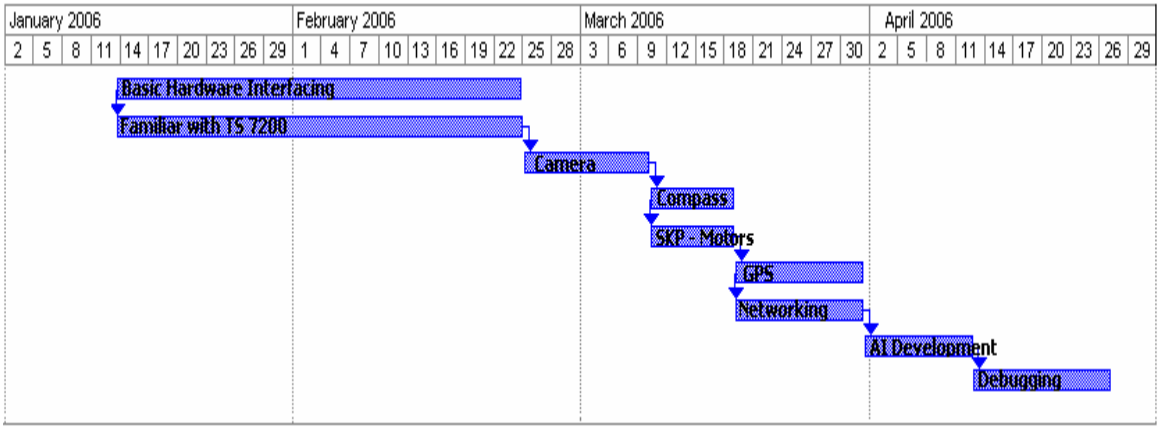

Figure 8: Gantt chart of software schedule.

VI. FUTURE DEVELOPMENTS

Tokka has a lot of potential and a lot of room for future development that was out if the scope of the current senior project time line. With respect to the content of this report, some future developments for Tokka could be:

- The completion and assembly of the outer shell of Tokka.
- Reassembly of Tokka's components inside the shell.
- Object recognition (Where it will recognize an obstacle is in its path and stop and go around the object and then continue on its path.)
- Obstacle detection. (If Tokka hits an object and stops its motion it will back up turn and go in a different direction.)
- A charging option on the outer shell so the shell will not have to be taken apart to recharge the batteries every time.

- A microphone could be added to record sounds with the general vicinity (spying techniques that could be used by the Navy for example).
- Different types of wheels that can be removable and replaceable in order to have the right wheels for the right situation.
- A more user friendly interface could be developed for easy use.
- A night vision mode for the camera.
- A light for the camera to see in the dark.
- A greater range for the wireless transmission of data.

Some of the suggested developments above were actually planned for this semesters work however the time constraints and unexpected delays in the rapid prototyping process delayed much of this work. Other developments listed are possible options that could be implemented in order to develop Tokka for a certain market. After basic functionality is completely finished Tokka can have different types of implementations added in order to attract different markets. For example search and recovery missions could use Tokka to search for people or objects within an area that has land and water such as a wetland area. New wheels may be required for navigation through muddy areas or to be able to push through low level grasses. Also Tokka could be desired by someone like the navy to help locate and destroy underwater mines therefore needing Tokka to have the option to fire an object at a potential land mine. The options that could be added to Tokka can be countless and will be different based on the target market.

VII. CONCLUSION

Developing an autonomous robot involves a careful integration of components. Even though Tokka was not fully functional at the time this paper was written, it had many of the basic capabilities and preformed as it was designed to with little difficulty. However, our team has made enough progress in the initial developments of Tokka that improvements and additional features can be implemented without reconstructing the entire concept. Above all else, this project also taught the team how important it is for each member to contribute and manage time wisely to complete as many tasks as possible. It showed how valuable project management and organization was to the success of the project.

Our team has left a copy of Tokka's design along with all of its internal components with the University of North Carolina at Charlotte so future teams of students can be formed to take on the challenge to make enhancement to Tokka's abilities.

ACKNOWLEDGMENT

The team would like to thank Dr. Conrad for his support in the project. His guidance was a critical part of Tokka's success.

REFERENCES

[1] Daisuke Iijima, Wenwei Yu, Hiroshi Yokoi, and Yukinori Kakazu. "An Acquisition of a Narrow Path Going Through Behavior for a Distributed Autonomous Swimming Robot," *Systems, Man, and Cybernetics, 1999, IEEE SMC '99 Conference Proceedings, 1999 IEEE International Conference*, Toyko, Oct. 1999, p. 602-607.

[2] Gangatheran.N, Rajasegaran G., Zahari Taha, S.Ramachdmn. "Design and Development of an Underwater Vehicle with the Structure Mounted on Spheres," *Seventh International Conference on Control*, Automation, Robotics, and Vision *(ICARCV'02h),* Singapore, Dec. 2002, p. 1470-1474.

[3] Moser, Ed. "Industry Shorts -Security: Pentagon Pouring Funds Into a Host of Land and Sea Robots." Robotic Trends, 6 Sept. 2003 <http://www.robotictrends.com>.

[4] Martian J. Dudziak, T. Renee Fields, Kenneth C. Youngman. "AI Programming VS. Conventional Programming for Autonomous Vechicles – Trade-Off Issues," *Unmanned Untethered Submersible Technology, Proceedings of the 1985 4th International Symposium*, June 1985, p. 284 – 296.

[5] Kimon P. Valavanis, Denis Gracanin, Maja Matijasevic, Ramesh Kolluru, Georgios A. Demetriou. "Control Architectures for Autonomous Underwater Vehicles," *Control Systems Magazine*, IEEE, Dec. 1997, p. 48 – 64.

[6] Gat, Erann. *Conditional Sequencing for Land, Space and Sea.* "Autonomous Underwater Vehicle Technology," *Proceedings of the 1996 Symposium*, Monterey, CA., June 1996, p. 216 – 222.

[7] Hamilton Kelvin, Lane Dave, Taylor Nick, Brown Kieth. "Fault Diagnosis on Autonomous Robotic Vehicles with RECOVERY: An Intergrated Heterogenous-Knowledge Approach," *Robotics and Automation, Proceedings 2001 ICRA. IEEE International Conference*, Vol. 4, 2001, p. 3232-3237.

[8] Daisuke Iijima, Wenwei Yu, Hiroshi Yokoi, Yukinori Kakazu. "Adaptive Behavior Acquisition for a Distributed Autonomous Swimming Robot Based on Real-World Learning," *Proceedings of the 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Kyongju, Mar. 1999, p. 230-234.

[9] Naomi Kato. "Control Performance in the Horizontal Plane of a Fish Robot with Mechanical Pectoral Fins," *IEEE Journal of Oceanic Engineering*, Vol. 25, No. 1, Toyko, Jan. 2000, p. 121-129.

[10] Naomi Kato. "Application of Swimming Functions of Aquatic Animals to Autonomous Underwater Vehicles," *OCEANS '99 MTS/IEEE. Riding the Crest into the 21st Century*, Seattle, WA, Mar. 2000, p. 1418-1424.

[11] Toshio Fukuda, Atsushi Kawamoto, Fumihito Arai, Hideo Matsuura. "Steering Mechanism and Swimming Experiment of Micro Mobile Robot in Water," *Micro Electro Mechanical Systems, MEMS '95*, Feb. 1995, p. 300 – 305.

[12] Gregory Dudek, Michael Jenkin, Chris Prahacs, Andrew Hogue, Junaed Sattar, Philippe Giguere, Andrew German, Hui Liu, Shane Saunderson, Arlene Ripsman, Saul Simhon, Luz-Abril Torres, Evangelos Milios, Pifu Zhang, Ioannis Rekletis. "A Visually Guided Swimming Robot," *Intelligent Robots and Systems (IROS 2005),* 2005 IEEE/RSJ International Conference**,** Montreal, Aug. 2005, p. 3604-3609.

[13] Bails, Jennifer. "CMU robot walks on water.", Pittsburgh Tribune-Review, Pittsburgh, 6 April 2006. <http://pittsburghlive.com/x/tribune-review/print_440639.html>.

[14] Anonymous. "Technologic Systems – PC/104 Single Board Computers and Peripherals for Embedded Systems" *Technologic Systems*, Feb 2006. <http://www.embeddedarm.com>.

[15] Xhaard Michel. "SPCA Video for Linux Drivers." <http://mxhaard.free.fr/>.