University of Massachusetts Lowell
Department of Computer Science
Lowell, Massachusetts

MCP:
A Low-Cost Robot
Developed by Extreme Prototyping

Table of Contents

History in Brief

The MCP began life as a child's toy. After being acquired by Professor Martin, it started to evolve with the efforts of the fall 2003 Robotics I class. Members of this same group continued work until the robot first went online in March of 2004. In this version it was controlled by two networked Handyboards (a programmable controller designed by Professor Martin). Its function was simple; wander and greet attendees of robotics expositions. While a good proof of concept the design lacked reliability and computational power. At the end of the fall 2004 Robotics I class there was a new team, and a new goal: the IGVC. At first many original subsystems were left in place, but it soon became clear that to compete in the IGVC a total redesign was necessary. The current version of the MCP draws heavily from what was learned by previous teams, but all the mechanical, electrical, and software components are new (excluding the drive motors and battery, both of which came with the toy).

Meaning of MCP

MCP stands for "Master Control Program" just as in the 80's movie "Tron." This arises from our basic design methodology: Each subsystem is complete and self contained, designed and built by either an individual or small sub-group. The Master Control Program off puts most of the work to these subsystems, and remains usefully abstract. This design paradigm better allows for the mixed schedules of students, and creates many components that can be used for a variety of projects. For ease of connectivity, all subsystems requiring control boards external to the CPU use an RS232 serial connector. Each student or group is responsible not only for building the electrical and mechanical components, but also for programming a daemon to control the subsystem. They then provide an UDP server with a published API to use the subsystem. This makes each component easy to plug into another application, and makes development of the Master Control Program more independent from the subsystem development. It is even possible to programmatically check if there is an instance of the proper UDP server running and start it if necessary. The basic architecture for the MCP is depicted in figure 1.

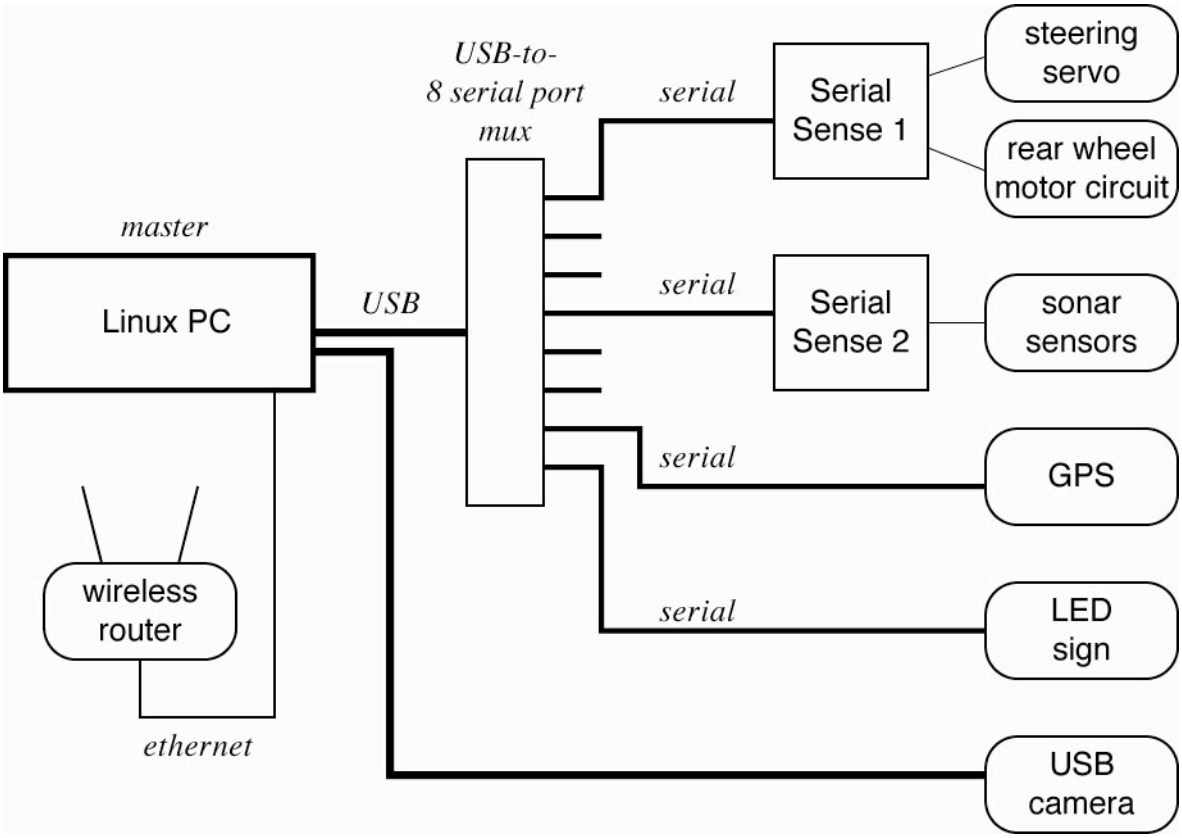

Figure 1: The basic architecture of the MCP

CPU and Software

The MCP's main logic is contained within an on-board Mini-ITX computer with a VIA x86 processor, a 40 GB notebook hard drive, and a built-in video card. Power is provided by an on-board 12V lead-acid battery. The MCP has a built-in sound adapter which outputs to Boston Acoustic speakers. In order to provide connectivity to several of the MCP's serial-based devices, there is an Edgeport adapter that provides 8 serial connections to one USB connector on the motherboard. Network connectivity comes from a wireless adapter that connects to a USB port. The MCP runs Debian GNU/Linux (Sarge) with a 2.6 kernel. Installation was performed by Jeff Rousseau.

The purpose of the navigation program is move the MCP from a starting position through a series of waypoints and then back to the starting position. Along the way, the program must adjust the MCP's course in order to avoid obstacles.

The navigation program is designed so that going to each waypoint is a subtask. The program reads in a text file which contains the order of the waypoints that the MCP should travel

to. The program creates a stack of waypoints. The MCP takes the top waypoint of the stack and travels to that waypoint. When the MCP reaches a waypoint, the top of the stack is popped. The MCP proceeds to read the next waypoint on the stack and then travels to that one. This process is repeated until there are no more waypoints.

During each subtask, the MCP may encounter obstacles that are between itself and a waypoint. When an obstacle is detected, the MCP will calculate new coordinates that are 5 meters to the left or the right of the obstacle. A new waypoint will be created from those new coordinates and will be pushed onto the stack. The MCP then works on the new subtask. Waypoints generated during the running of the program are called fake waypoints. The MCP then has a new subtask of traveling to the fake waypoint. By traveling to the fake waypoint before a real waypoint, the MCP can avoid obstacles.

Motor and Steering Control

The Power Wheels toy upon which the MCP was built has a 12V 10Ah battery that drives two 12V 20A DC motors. We first tried creating our own speed controller, but ran into many difficulties and failed attempts. Professor Martin suggested using factory made speed controller after these several mishaps despite the cost. For its main motor, the MCP uses a speed controller manufactured by MULTIPLEX. This speed controller is controlled by a SerialSense board (another product of Professor Martin's lab). The circuit (figure 2) developed to utilize the speed controller uses the SerialSense board to send it two signals, one for direction, and one for speed.

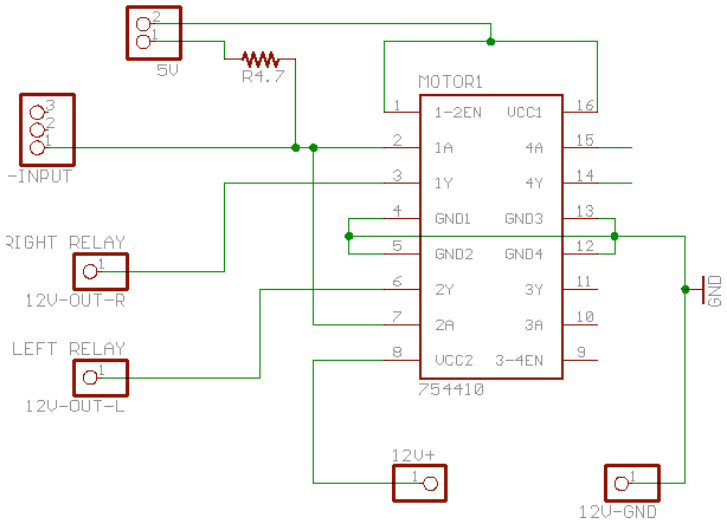

Figure 2: Motor circuit (simplified)

The SerialSense board is a vital component to the steering system as well, because it runs the code for our subsystem and the rear motor subsystem. It is connected to the servo board through a Cricket Bus and to the ITX board through a serial connection. Various commands are sent from the ITX board to the SerialSense and it is up to the program running on the SerialSense to interpret the commands and send out the appropriate signal to the servo board. The SerialSense board is also responsible for monitoring the sensors and making decisions based on the sensor readings.

The servo board has a simple function; it takes signals from the SerialSense board through the Cricket Bus and translates them into the appropriate servo command. It then takes those servo commands and sends them to the variable speed controller.

The speed controller accepts servo commands from the servo board through its Cricket Bus port, and from those commands it generates a set amount of power. The speed controller is hooked up to the steering motor (figure 3), and when it receives a servo signal it powers the motor until it is told to stop.

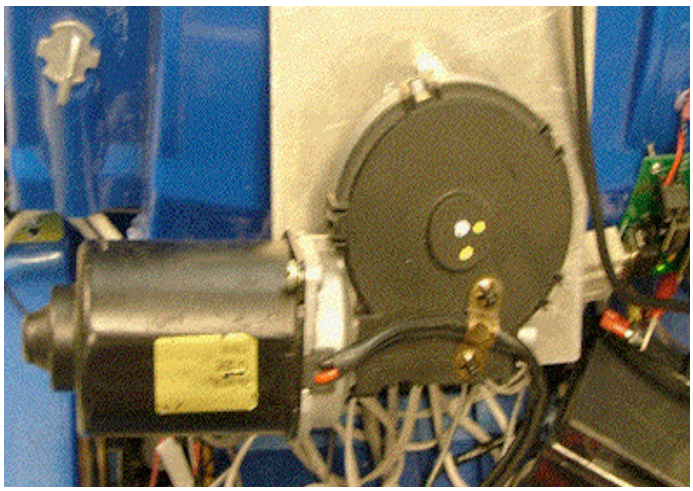

Figure 3: Steering motor and motor mount

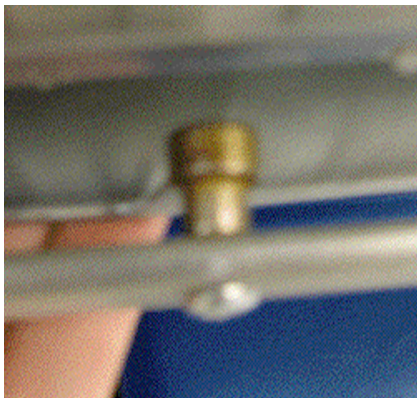

Figure 4: Magnet sensor (center)

There are three magnetic sensors (figure 4) to limit the steering capacity. To prevent the steering motor from turning too far in either direction, there is a sensor on each side. When the sensors are tripped, the steering motor stops. There is also an additional magnetic sensor that allows the wheels to stop in the center, rather than only having the full-left and full-right settings.

Sonar, Vision, and GPS

The cheapest, most reliable sensor for solid object avoidance is a sonar sensor. The first version of the MCP required the creation of a sonar multiplexer board to control the large number of sensors needed. Integrating this design with the SerialSense design gave birth to the MCP's sonar multiplexer subsystem (figure 5).

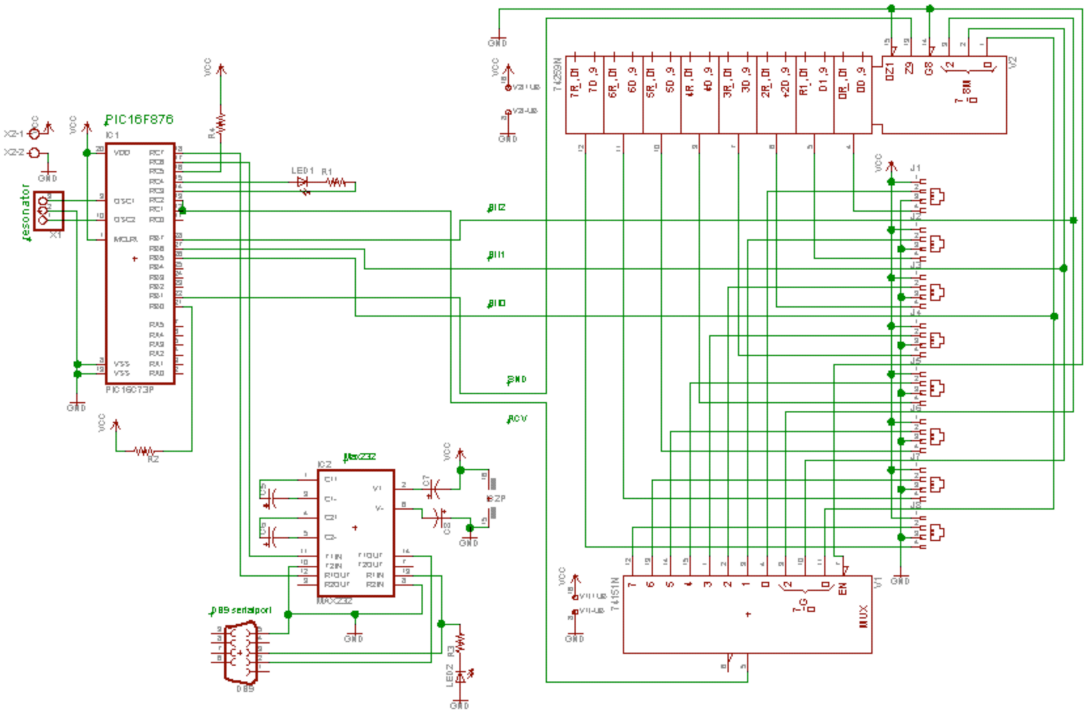

Figure 5: The sonar multiplexer, controlled by a PIC microprocessor

This design controls up to eight sonar sensors. The eight sensors used on the MCP are concentrated towards the front of the robot, as little backward movement is expected (figure 6).

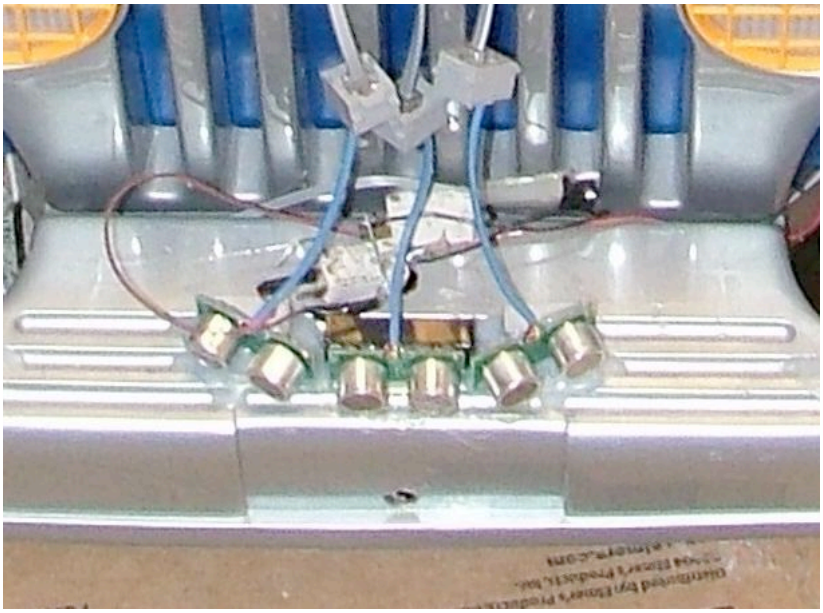

Figure 6: forward sonar sensors

For the autonomous challenge, robot vision is practically a necessity. We used a simple webcam coupled with an innovative computer vision procession system called Phission that is being developed by UML Masters student Phil Thoren. This allowed for relatively rapid development of what would normally be a very difficult component. Phission captures video from the camera in real time, interprets the image, and plots a course to travel.

The last component that assists the MCP in navigation is the Garmin model GPS 16 HVS differential Global Positioning System. This is an OEM sensor quality outdoor GPS with full differential capability. Once again drawing from the open source community, the GPS is accessed via the GPSD daemon published under the GNU General Public License.

Appendix A: Cost

Item	Quantity	Cost (each)
Power Wheels Toy	1	$200
Magnetic bicycle sensors	3	$10
Mini ITX	1	$200
40GBHD	1	$100
Memory	1	$75
Power supply	1	$75
Sonar sensors	8	$20
USB / RS232 adapter	1	$45
Pro-Lite sign	1	$100
Remote Control	1	$30
Steering Motor (donated, est. cost $75)	1	$0
Extra drive battery	1	$50
Extra logic battery	1	$40
Wireless Adapter (donated, est. $30	1	$0
Sonar MUX board	1	$100
Motor control relays	2	$10
Victor speed controller	1	$130
Airplane speed controller	2	$45
Speakers (donated, est. $45)	1	$0
Wireless Router	1	$20
TOTAL		$1,250

Appendix B: Credits

MCP Team: Kareem Abu-Zahra, Aron Barabás, Brian Corbin, Zeb Heisey, Kyewook Lee, Yan Tran, Schuyler Salz, Jon Victorine

Faculty Advisor: Professor Fred Martin

Mechanical Engineering Assistance: Matteo Forgione

SerialSense Board: Andrew Chanler

"Phission" Vision Processing Software: Phil Thoren

Linux Installation and Support: Jeff Rousseau