GPS Guided Robotic Car

Progress Report 2

Prepared for: Dr Jason Blough
Summer Undergraduate
Research Foundation

Prepared by: Alex Morozov
June 18, 2008

"GPS Guided Robotic Car"

Progress Report 2

Objective

The objective of this document is to inform the reader of progress made towards the project's goal and all the problems encountered in process.

Procedure

General overview of the activities performed during the last three weeks:

1. The first week was dedicated to the SAE Supermileage Competition. It was held in Marshall, MI at the Eaton Corporation proving grounds. The two day competition started off with a technical inspection. Our team was asked to redesign and remanufacture two safety systems: seatbelt harness and firewall. The main problem with seatbelts was the fact that they were made out of thin and narrow material that seemed strong enough in static conditions but would most probably fail under dynamic loads. The team easily satisfied this requirement by rebuilding this system using more durable materials and safer buckles.

   The second safety hazard (the gap under the firewall, between body-shell and firewall) was potentially hazardous in case of fuel leak, or fire, or another emergency situation. So the team was required to find a way to seal the gap. This was done overnight (using some aquarium sealant and window sealing tape) and successfully passed the technical inspection the next morning.

   The second day was the actual racing day. While some teams were still adding final touches to their cars to pass technical inspection (and in some cases finishing building the car) our team headed for the track. It is important to mention that the track is 1.6 miles long and the rules of the competition require every team to complete six laps. By combining this rule with others concerning with average lap speed (should be above 15 mph and below 25 mph), one could easily figure out that the longest time a

car could be on the track without receiving any time penalties was thirty eight minutes. So every full run would take a team about an hour with all the preparations and inspections.

Our first run was successful and resulted in a figure of 386 miles per gallon. Then we completed a couple more official runs (for detailed description see appendix 1) before we detected a fuel leak (the fuel bottle was completely empty, whereas before that at the end of a run it was seven eighths full). The reason for this leak was a defective fuel pressure gauge. Luckily enough our team had a spare part and we were able to continue the race.

Two more attempts were completed with a final result of 456 miles per gallon. This was the last run our team has completed and it placed Michigan Tech eleventh against participating universities and high schools. Considering that only twenty teams made it through the technical inspection, it places our team right in the middle of the pack, but at the same time this result should not be undervalued due to the fact that it was our first-ever entry in this competition.

During the last runs, GPS data was collected and later plotted (see appendix 3 for plots). This information will now be instrumental in developing the 'cruise control' algorithm.

To read an article about our team's performance and see an interview with a couple of our team members, please navigate to http://www.wluctv6.com/news/news_story.aspx?id=148272 or go to the TV6 website (wluctv6.com) and search for supermileage. Also the competition pictures and videos will be available soon at the ASE website at www.enterprise.mtu.edu/ase .

2. Upon the return to Houghton, the work on the GPS has been continued. The next step was to connect all the functional parts of the system together and interface them. This was attempted by first starting to connect GPS module to the microcontroller using a UART port (Universal

Asynchronous Receiver/Transmitter). This part wasn't hard as far as writing code, but the part that was challenging (and actually still is challenging) was to test the functionality of the recently composed code. For that purpose an LCD (Liquid Crystal Display) was connected to another UART port on the microprocessor board.
Although the task of reading data from one port and writing it to the other port might seem trivial to a casual observer, it is not. As it was discovered there are fourteen commands for loading data into a register or a port. There are thirteen commands for the reverse operation (storing register/port data to memory). Each of these commands has its own purpose and functionality even though in some cases the names of the commands are the same. It also became clear that the compiler will accept most of these commands in the same place without giving an error, but the result of these commands will be entirely different. Another conundrum encountered along the way was the way LCD displayed the characters (in rare instances when it worked). The characters on screen were Japanese hieroglyphs (I asked a friend of mine to comment, but they didn't make any sense to him either) mixed with commas, black squares, pound signs, percents, Greek symbols and a lot of other … symbols. So this task of 'trivial' verification of the design became a four-fourteen-hour-day-nightmare and resulted in no forward progress except better familiarity with microprocessor's datasheet, instruction set and a handful of missed lunches and dinners.

3. Although this part of the project resulted in a complete failure on the software part, a brainstorming session yielded a couple of solutions to this seemingly unsolvable problem. The first one was to diagnose the hardware. What if it is a bad LCD? Bad microcontroller? For that route another identical LCD and microcontroller were ordered. Another idea was to switch to a different brand of microcontrollers (What if the registers there are mapped incorrectly inside and its outputting the data in the

wrong port or in the wrong form?). Even though this idea is still considered, it seems not very reasonable and is left for the future research. Another way to proceed was suggested by an outsider and seems to make logical sense. His main advice was switching from low level programming that is performed right now (assembly programming language) and use higher level, object oriented language (C). The main reason behind this suggestion is the idea that while the low level language provides a lot of control over processor, it requires big amounts of extremely picky programming that is very hard to debug. Whereas high level language doesn't provide as much CPU management utilities, but requires much smaller amounts of programming and provides friendlier troubleshooting atmosphere.

The last but not the worst solution is to use a bit different platform altogether. Up until recently all the development has been done on an AVR (brand) RISC (Reduced Instruction Set Computer) microprocessor, as shown in appendix 2, figure 1. What is being proposed is to use a similarly sized, but more computationally powerful system (basically it is a microprocessor plus memory plus advanced IO (input output) plus full motherboard plus a color LCD, all a size of child's palm). This system is called Gumstix (all boards are 'gum-stick'-sized) and actually runs an operating system (Linux). So one could actually call it a "full sized" computer. It has a 4.3" color LCD touch-screen, Bluetooth antenna, serial ports, USB port, Ethernet port and even a micro-SD card slot (see appendix 2, figures 2 and 3). This option is in the research stage and is still only under consideration at least because having a full computer also means performing more maintenance tasks and harder to get at IO procedures.

Conclusion

The progress made in the second quarter of the project time span was planned in one way, but software and hardware problems made the realization of these plans impossible. However, these mistakes not only made the researcher aware of some unusual quirks of the microcontroller, but also let him realize that there are alternative ways to substantiate the desired control system.

Looking back at the progress made through the half of this project, I start to realize that there were a lot of small (and big) things I didn't know about when I wrote a proposal for this project. For example, I was planning to connect the LCD to the microcontroller and it would just work. In reality it seems like the small things are the one requiring the most attention. Most of my fourteen hour day goes not into designing a Kalman filter, but instead it disappears in an endless sequence of different assembly commands. A small twenty line (core) testing program took me four days to troubleshoot and is still not finished.

So the main lesson to be learned from this period is:

> Knowing how much you can't see or predict can be very important, sometimes even more important than what you can see. The unseen obstacles can put the project completion days and weeks behind schedule.

And the main thought for the remainder of the project:

> Use time more wisely. Half of it is already gone. Always have a backup plan. If idea seems not to work out for more than thirty hours, move on to an alternative.

Appendix 1. Competition fuel efficiency data

Run 1

Tire Pressure	100	psi	**Adjustments for next run**
Total Time	35:47:00	min	1) Steering: tow in 1/2 turn
Mileage	386	mpg	

Run 2

Splits			**Notes:**
	5:59		1) Rear body came off
	6:12		
	6:03		
	5:45		**Adjustments for next run**
	6:15		1) Tow changed 3/4 turn
	5:37		2) O2 sensor auto tuned with Meqasquirt
			3) Water taken out of driver's bottle
Tire Pressure	100	psi	4) Muffler
Total Time	36:00:00	min	
Mileage	325	mpg	

Run 3

Splits			
	6:31		**Notes:**
	5:15		1) DNF
	6:30		2) Fuel pressure gage leak
	6:25		
	6:40		**Adjustments for next run**
			1) Replaced fuel pressure gage
			2) Straight pipe exhaust
Tire Pressure	100	psi	

Run 4

Splits		
	5:58	
	6:17	
	6:40	
	5:20	
	6:25	
	6:27	
Tire Pressure	100	psi
Total Time	37:15:00	min
Mileage	331	mpg

Adjustments for next run

1) Covered intake to prevent wind issues
2) Tire pressure changed to 120 psi

Run 5

Splits		
	5:31	
	5:28	
	5:24	
	6:10	
	6:38	
	6:25	
Tire Pressure	120	psi
Total Time	35:55:00	min
Mileage	456	mpg

Appendix 2. Platform comparison: Microcontroller VS Gumstix

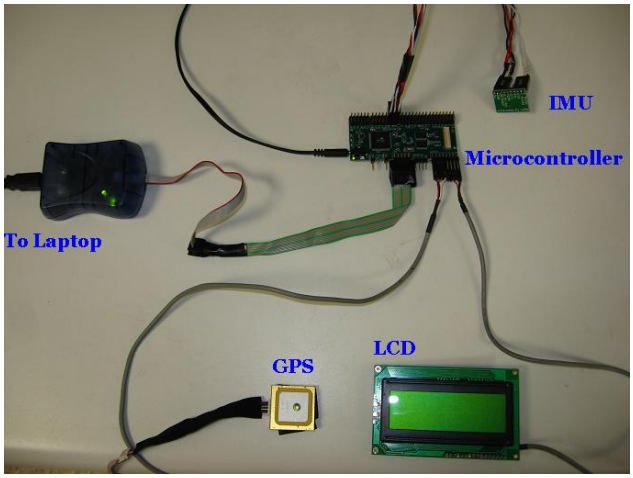

Figure 1. Current setup. IMU is Inertial Measurement Unit (3 accelerometers).

Figure 2. Gumstix assembled with LCD

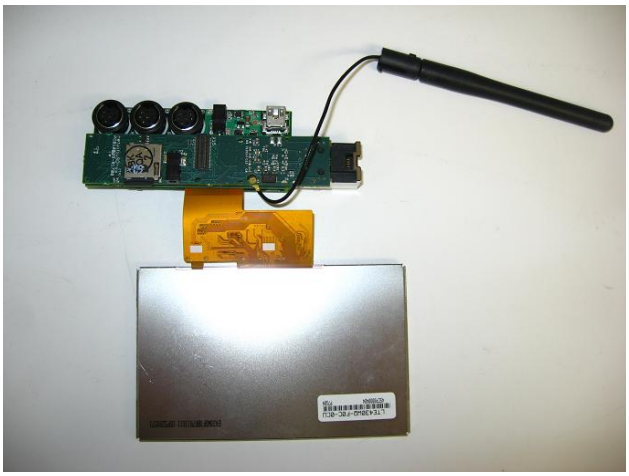

Figure 3. Gumstix. Take note of the micro-SD card, Bluetooth antenna, Ethernet, USB and serial ports.

Appendix 3. Competition GPS data

Page 11. Eaton Proving Grounds with the MTU Supermileage car's GPS data overlaid. The coloring scale is speed (see bottom left corner for a legend). The highest speeds were achieved going downhill (Southeast bend of the track/junction with the straightaway). These high speeds are shown in dark blue/purple/magenta.

Page 12. In this run the pattern is even more vivid and clear: the driver was slowing way down in southwest corner, speeding up downhill (southeast), coasting on the eastern straightaway and mixing and matching the rest of the track.
The red lines in the northeastern part signify prestart inspection (engine off).


100%
USGS aerial
Tracks:
- LOG11
Speed (km/h)
46.8
35.1
23.4
11.8
0.1
Center: 42.22375,-84.91687
200 ft
100 m
Map created at GPSVisualizer.com


100%
USGS aerial
Tracks:
- LOG12
USGS
Speed (km/h)
46.0
34.5
23.0
11.5
0.0
Center: 42.22335,-84.91687
Map created at GPSVisualizer.com