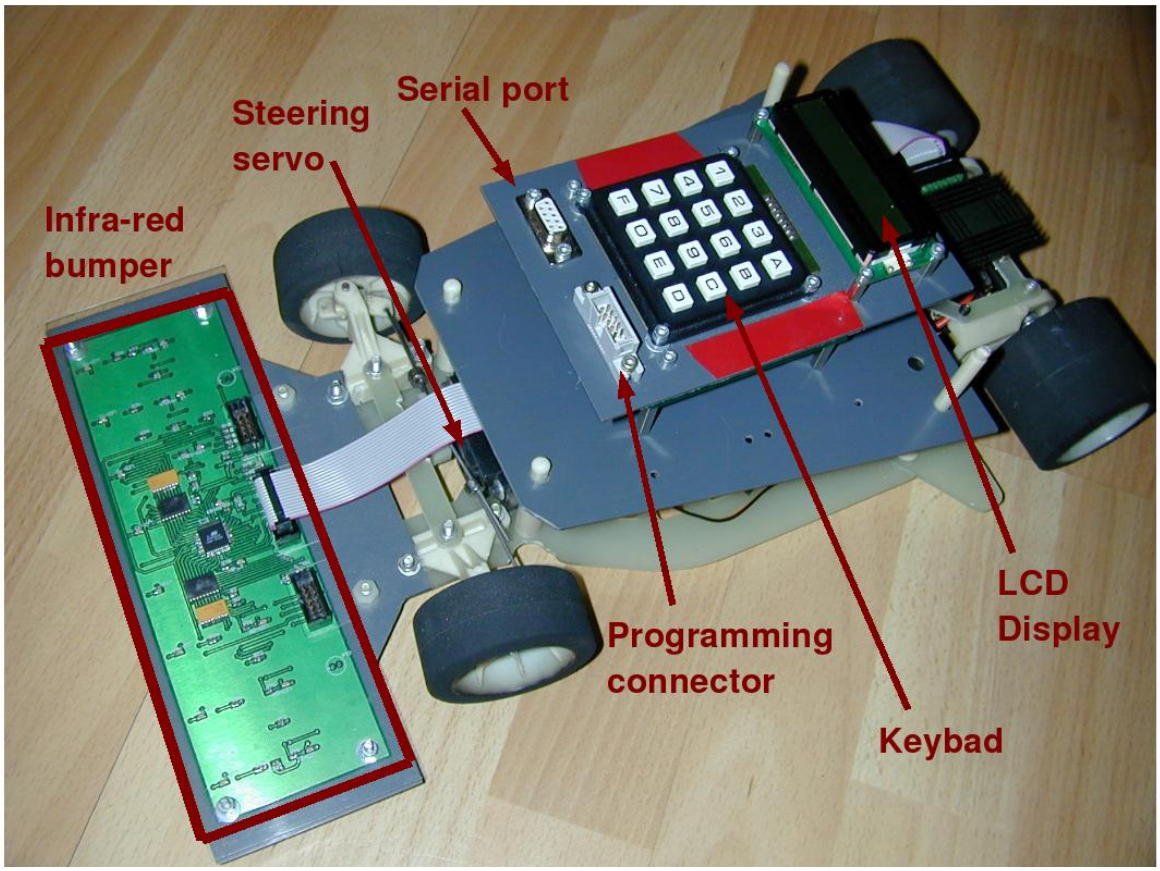
T-106.5300 Embedded System

CAR PROJECT REPORT



Done By

Amit Soni - 245470

Vu Ba Tien Dung - 206383

Swaminathan Vasanth Rajaraman - 245221

Overview:

The goal of this project is to program the Atmega32 microcontroller in order to make a car run in a track marked by reflective tape for 3 consecutive rounds.

The report is organized as follows. In the overview section the procedure for running the car is explained. In section 1 the steering algorithm is explained followed by the explanation to the driving algorithm in section 2. In section 3 the additional features for the car namely PID controller, LCD, Lap count, automatic STOP, safety measure and keypad are explained.

The software is an essentially an interrupt-driven system, with the following interrupts:

1. Interrupt using Timer-0 overflow in timing some operations such as lap count, tachometer reading
2. Interrupt using Timer-1 in Fast PWM mode for steering servo
3. Interrupt using Timer-2 in generating the PWM control signal for driving motor
4. INT0 for tachometer reading
5. INT2 for keypad reading

Procedure for running the car

1. Reset the car to its initial state by using the RESET button which can be either in front or in the back of the car.
2. Load the configuration of the car using the keypad buttons. Details are provided in the “Keypad” section.
3. Start the Car by pressing the START button which can be either in front or int he back of the car. This button is connected to PB1 of the micro-controller.
4. Stop the Car by pressing any of keypad buttons.

1. Steering Algorithm

We divide the infra-red bumper into two parts. We calibrate the zero angle turn, the rightmost turn and leftmost turn and use these value as the reference value for our algorithm. We assign a weight to each sensors in the infra-red bumper so that sensors at each ends are given higher weight than the sensors near the center. In our case, we engineered these weights as 10, 25, 75 and 100 to the sensors at one side of the bumper progressively going from center to the end. These weights are the same for the other side also but opposite in the sign. Since the difference between the rightmost turn and the zero turn is not same as the difference between the leftmost turn and the zero turn we need to consider weighted mean of turn separately for both the sides.

If some sensors in the right side reflect, we take a weighted mean of sensors and multiply with the difference between the rightmost turn and the zero turn.
If some sensors in the left side reflect, we take a weighted mean of sensors and multiply with the difference between the leftmost turn and the zero turn.
We finally take the weighted mean of left and right turn to find the actual turn value. We assign that value to the OCR1B.

If the sensor gives zero reading, it has gone out of track. so instead of backing-up and relocating the track, we steer it in maximum position in the last known track direction. This is done because it is observed that typically car looses track on sharp turns and this strategy is the best to regain track.

There have been outliers in the readings of the infra-red sensors. If some sensor readings is unexpected, we continue as per the previous reading.

2. Driving Algorithm

We calibrate the expected tachometer reading based on a pre-defined speed. For the first half second while we run the car, we run it at an arbitrary initial speed which is not calibrated hence not reliable. After one half second we read the tachometer. We compare the reading with the expected tachometer reading and then change the initial speed to the standard maximum speed.

During the normal course of the run the speed is changed according to the turn the car is making. We proportionally decrease the speed of the car as the turn angle increase. However, in the maximum turn we again increase the speed. This is because during the maximum turn the importance of reading a sensor decreases the car can run at a greater speed. This affect has been tested to give better performance.

3. Additional Features

The following additional features are implemented in the car:

1. **PID controller**:
   We have implemented the PID controller for the weighted average turn value. We take the base value to be the zero turn and error value is the difference between the zero turn and the weighted average turn value.
   - *Proportional* component is obtained by multiplying the *proportional* coefficient to the error. A high proportional component results in a big change in the output for a given change in the error value.
   - *Integral* component is obtained by multiplying the *integral* coefficient to the sum of all the errors till now. Error can be both positive and negative so the integral component doesn't go out of bound. The integral component accelerates the movement of the process towards stability and removes the residual steady-state error value which occurs with a pure proportional controller.
   - *Differential* component is obtained by multiplying the *differential* coefficient to the difference of current and previous error value. Derivative component is used to reduce the effect of the overshoot produced by the integral component. The combined used of integral and differential component can improve controller-process stability.

   The final turn value is the sum of all the individual components.

2. **Integration of Hitachi HD44780 LCD**
   The LCD of car is based upon the Hitachi HD44780 driver which has a 16 x 2 dot matrix character display. The library and LCD code of Peter Fleury was used as the driver which can act both in memory-mapped mode and in a 4 bit I/O mode. Since the LCD is not connected as per the standard way, certain modifications in the code were done such that the code can be adapted for the car project environment. The Port, Pin and clock frequency are tuned as per our requirements, leaving the functions and definition done by Peter Fleury as such. The working of LCD is as follows:
   a. It welcomes with a Welcome! Message at the start of the screen.
   b. Selected configuration of the Car is displayed based upon the key pressed.

c. After stopping the car, Car Stopped message is displayed
d. After each lap, the lap number which was finished is displayed.

3. Counting the number of Laps and automatic STOP

We have created an algorithm to track the number of laps being run by the car with the help of the LEDs in the bumper. Based upon the number of LEDs glowing at the same time the lap number is been determined. If the number of LEDs which is ON is greater than 6 then we can consider that the car has just crossed the starting line and hence we can increment the lap count by 1 and display it on the LCD.

In addition to the, we had also implemented automation STOP feature, through which the car can stop itself after certain number of stops as per our requirement. We have designed our car in such a way that it STOPS after 3 continuous laps.

5. **Safety Measure**

Safety measure is followed for the car in order to stay within the track even though the reading of the sensor shows some abrupt values. In our car we have implemented a safety mechanism in such a way that the car compares the previous sensor reading with the current sensor reading before changing the position of the steering. It always takes the sensors present in the middle of the bumper into consideration and acts accordingly. If the difference between the two readings is more than 1, then it can be ignored and we go based upon the current values.

6. **Keypad**

We have implemented the feature of keypad for using it to stop the running car and selecting the respective Car configurations. While the car is running, any buttons in keypad can stop the car. When the car is not running, .the following buttons are used to select the car configurations:

Keypad Button number	Action
Button 1	Loads configuration of Blue Car 1
Button 2	Loads configuration of Blue Car 2
Button 3	Loads configuration of Red Car
Button F	Loads configuration of Black Car

Conclusion

The usage procedure of the car was explained initially followed by an explanation to the working of the car which uses Atmega 32 microcontroller with an integrated LCD and keypad. In

addition, the features like LCD, keypad, safety measures, lap count and automatic STOP which were implemented by us were explained.

Reference

1. LCD library for HD44780U-based text LCD display by Peter Fleury. http://homepage.hispeed.ch/peterfleury/avr-software.html
2. PCF8574 Remote 8 bit I/O expander for I2C bus datasheet for keypad
3. AS5035 data sheet. Magnetic rotary encoder from Austrian microsystems
4. Keypad functionalities are modified from the original file present at www.**sparkfun**.com/datasheets/BreakoutBoards/ , Copyright of Spark Fun Electronics 2009 by Viliam Klein.