CMPE 443 PRINCIPLES OF EMBEDDED SYSTEMS DESIGN
The Final Project
"Autonomous Car"

1. Description

In the final project of the CMPE 443 – Principles of Embedded Systems Design class, we were asked to design the software and hardware of a car that would operate in two modes: manual and autonomous. This time, using hardware I have managed to implement both the manual and autonomous scenarios.

There are numerous differences to the simulation code. Firstly, the speed of the motors is now controlled by RPM value and not by duty cycle value.

The LEDs now do not each have a separate timer match pin attached to them, but rather have a single timer match pin acting as a common ground to all of them. This is a much more clever way to deal with the lack of enough timer match pins, and also saving on the number of pins used.

The ultrasonic sensor and the LDRs now work. They take readings and update the status of the car every 90 milliseconds, which seems to be more than enough for the use at hand.

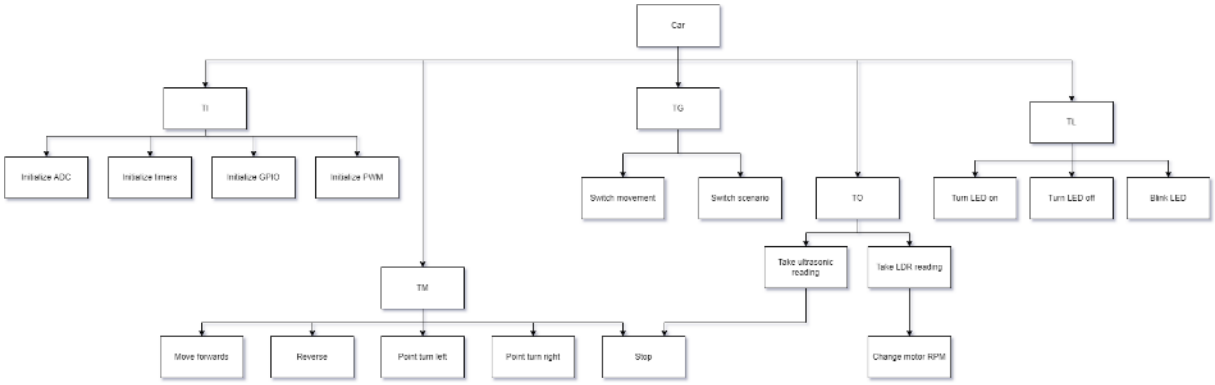
I have also used some functionalities which were not possible to use in the simulation due to Keil and Proteus misbehaving. I have used the PWM peripheral instead of timer match pins for driving the motors, and also for giving a trigger pulse to the ultrasonic sensor. Also, I have utilized the burst mode of the analog-to-digital converter to take continuous readings from the LDR, making the real-time RPM change for motors possible.

The basic structure of my code still depends on shared variables being set by interrupt service routines. The details of these shared variables will be given in the shared variable table below. After these shared variables are set by interrupt service routines, the update method is called, which checks the status of the shared variables and makes a call to the proper user-written interrupt handler to deal with the necessities of the interrupt. Different to the simulation code, since now I have managed to implement the autonomous scenario, there are two update functions depending on which scenario the car is currently executing. These differ slightly: in the autonomous scenario, all GPIO inputs except for JOYSTICK-UP, JOYSTICK-CENTER and PUSH-BUTTON are ignored. Also, LDR ADC readings are taken alongside ultrasonic readings.

Also, naturally we are now using the joystick and push button provided on the Experiment Base Board and the QuickStart Board instead of generic push buttons.

I have made use of the __WFI(); function, which is sure to save a lot of power in comparison to busy-waiting.

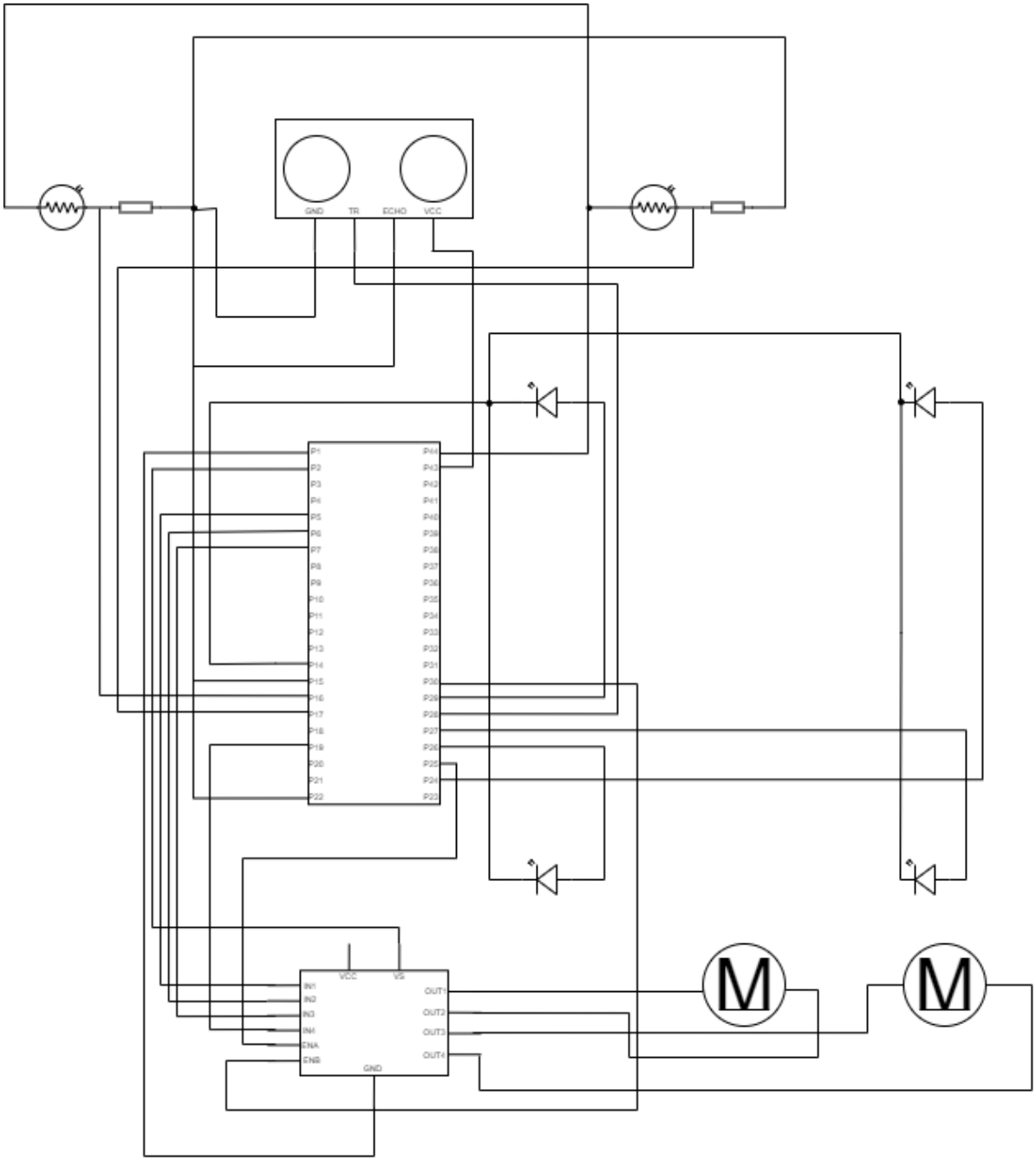
2. Structural Block Diagram



3. Pin table

PORT AND PIN NUMBER	USAGE	DESCRIPTION OF FUNCTIONALITY
P1	MC-GND	GROUND
P2	MC-POW	5V POWER SUPPLY
P5 = P1.24	MC-IN1	O
P6 = P1.23	MC-IN2	O
P7 = P1.20	MC-IN3	O
P14 = P0.6	LED-COMMON-GND	T2_MAT_0
P15 = P0.23	US-ECHO	T3_CAP_0
P16 = P0.24	LDR-LEFT	ADC0_IN1
P17 = P0.25	LDR-RIGHT	ADC0_IN2
P19 = P1.30	MC-IN4	O
P22	SENSORS-COMMON-GND	GROUND
P23 = P2.10	PUSH-BUTTON	I
P24 = P1.12	LED-FRONT-RIGHT	O
P25 = P1.11	MC-ENA	PWM0_6
P26 = P1.7	LED-BACK-LEFT	O
P27 = P1.6	LED-BACK-RIGHT	O
P28 = P1.5	US-TR	PWM0_3
P29 = P1.3	LED-FRONT-LEFT	O
P30 = P1.2	MC-ENB	PWM0_1
P31 = P5.3	JOYSTICK-CENTER	I
P32 = P5.2	JOYSTICK-UP	I
P37 = P5.4	JOYSTICK-RIGHT	I
P38 = P5.1	JOYSTICK-DOWN	I
P39 = P5.0	JOYSTICK-LEFT	I
P43	US-VCC	5V POWER SUPPLY
P44	LDR-COMMON-VSS	3.3V POWER SUPPLY

4. Circuit schematic



5. Shared variable table

Variable	Set	Clear
duty_cycle	ADC_Init	-
MOTOR1_ONTIME	ADC_Init	-
MOTOR2_ONTIME	ADC_Init	-
interrupt_flag	TIMER0_IRQHandler, TIMER1_IRQHandler, TIMER3_IRQHandler,	update, update_autonomous
scenario	change_orientation, change_orientation_auto	main

direction	go_forward, reverse, turn_right, turn_left	-
rising_edge	get_pulse_width	
pulse_width	get_pulse_width	update, update_autonomous
ic_pin	get_pulse_width	get_pulse_width