Carrie Cramer and Tarana Chowdhury

Professor Giles

EC 450: Microcontrollers (Microprocessors)

Final Project Report

Goal and design of the project:

Our goal is to create a robot that follows a marked path using optical sensing and proportion control. The sensor is made up of photo resistors and LEDs. The sensor readings are processed by the ADC and pulse with modulation is used to power the motors. We have prepared a path consisting of a black line on white paper. Our robot will follow this path. The initial step the robot takes is calibration. During this stage the user must first place the robot on a completely black surface until the LEDs begin to blink. Then the user must place the robot on a white surface until the LEDs blink again, which indicates that calibration is complete. Now the user may place the robot on the designed path. Then the robot will commence to traverse on the path.

Description of the implementation including system level and component level interaction:

Overall:

Each sensor on our robot will read in values as input from the reflection of the LED on the marked path. The microcontroller will then apply an analog to digital conversion on channels A4 and A5. We are implementing this using DTC and this stores the results in a buffer (latest_result[]). Once we have captured these reading we convert the digital values into corresponding duty cycles using the function duty_cycle(), which returns a duty cycle that is either 0 indicating that the robot is on the black line and the motor should stop or 60-75 indicating that the robot is on a white surface and the motor should move accordingly. The pulse width modulation occurs in the timerA_handler(), which counts up to the duty cycle as calculated by the duty_cycle() function. When the counter reaches a particular motor's duty cycle it turns the motor off and then is reset to 0 and when the counter reaches 100 it turns the motor on.

Main Function:

The main function stops the watchdog timer, sets the clock to 8Mhz, calls init_adc(), sets the LED pins to output, calls the calibration function calibrate(), sets the motor pins to the output direction, calls init_timerA, enables interrupts, and then enters into a forever while loop which will continuously read in sensor values and calculate duty cycles.

inti_adc():

The function initializes the ADC. The ADC10CTL1 register is set to INCH5 because the highest channel we are using is A5. We set the SHS_0 bit to use ADC10SC bit to trigger sampling. We set ADC10DIV_4 bit in order to divide the ADC10 clock by 5. We also set the ADC10SSEL_0 to set the source clock to ADC10SC, and lastly we set the CONSEQ_1 in order to do a multichannel single conversion. We have enabled A4 and A5 as the analog input channels and set the ADC10DTC1 to 2 because we perform two conversions. We also set options in register ADC10CTL0 that uses Vss and Vcc as reference voltages, selects the slowest sample and hold time, turns on the ADC10, enables conversions, and sets it to a multisource channel.

calibrate():

In order to determine the proper threshold values of the sensor readings we have a calibration process that takes place in the function calibrate(). In this function we take the average of six sample readings on both black and white surfaces to set the max and min thresholds respectively. This procedure will allow our robot to function in different lighting environments.

init_timerA():

In this initialization function we enable the CCR0 interrupts which will occur every 100 cycles. We also start the timer, use SMCLK, and set the timer to up mode.

start_converison():

This function waits for ADC10CTL1 and ADC10BUSY to be set so that there is no conversion performed simultaneously. Then we set the ADC10CTL0 to ADC10SC to trigger the conversion. ADC10SA points to the latest_result[] buffer.

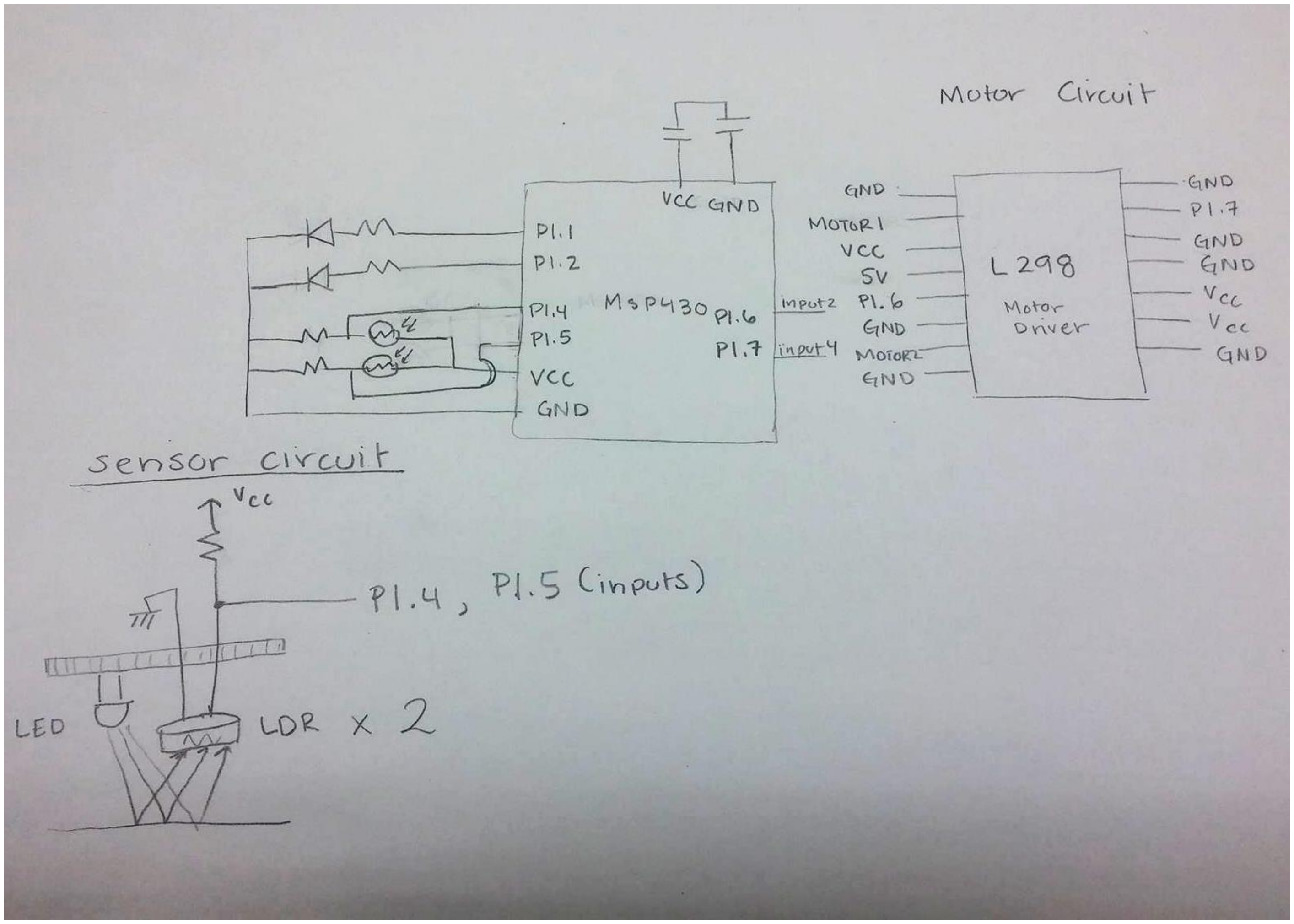
duty_cycle():

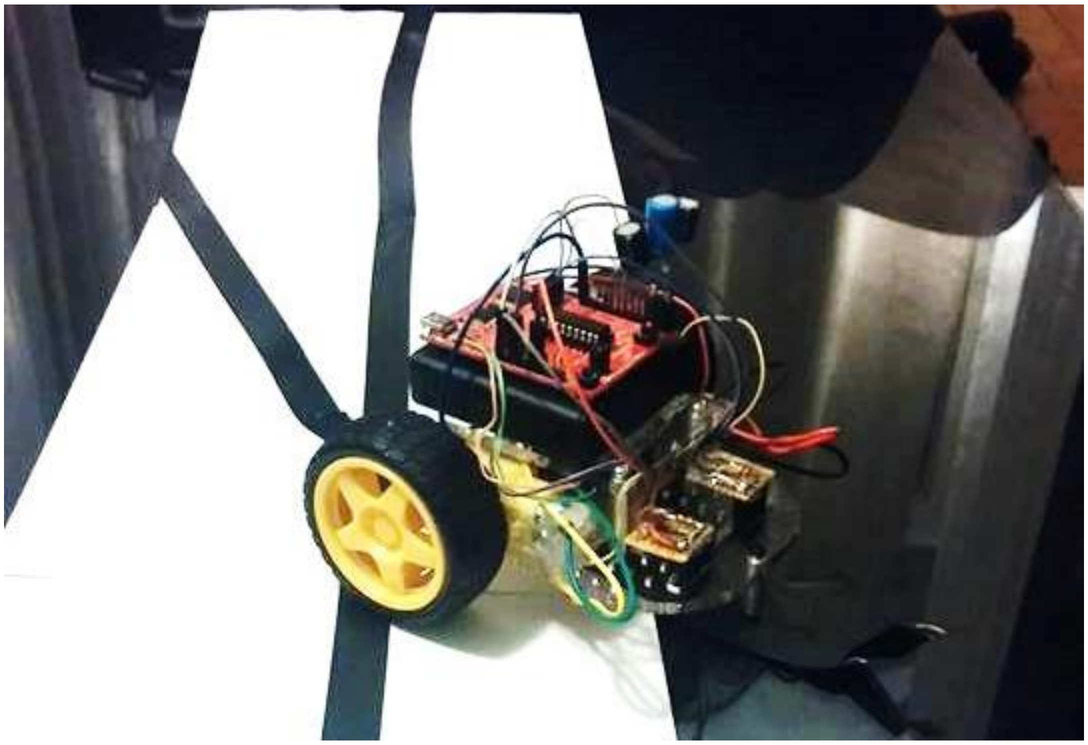
This function takes the adc value, the minimum and maximum sensor readings as inputs and outputs the duty cycle. This function ensures that the sensor reading is within the range determined by the calibration process. It then calculates the result and normalizes it to a duty cycle value between 0 and 100 percent.

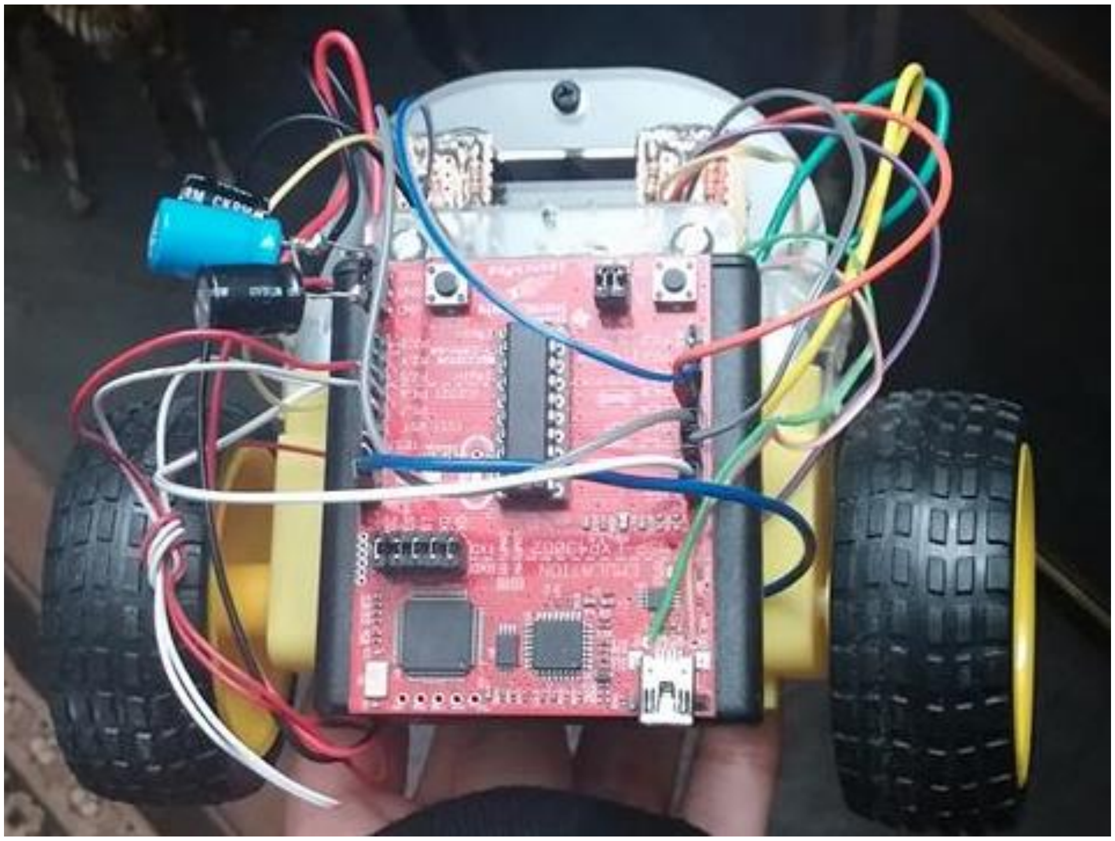
timerA_handler():

The motor speed is controlled by pulsing the motors between on and off. The duty cycle is the percentage of time the motor is active—a higher duty cycle means a faster speed. This handler increments the duty cycle counter value until it reaches 100 or equals a duty cycle. Upon reaching 100 it will reset the counter to zero and turn on the motors. If it reaches a duty cycle value it will turn off the corresponding motor.

Schematics:


Motor Circuit
VCC
GND
P1.1
P1.2
P1.4
P1.5
VCC
GND
MSP430
P1.6
P1.7
input2
input4
GND
MOTOR1
VCC
5V
P1.6
GND
MOTOR2
GND
L298
Motor Driver
GND
P1.7
GND
GND
Vcc
Vcc
GND
sensor circuit
Vcc
P1.4 , P1.5 (inputs)
LED
LDR x 2





Assessment of the success of the project:

Project mostly successful, sometimes robot does not respond fast enough if it is moving above a certain speed. Power saturation problem exists, so batteries need to be changed frequently. Otherwise runs very smoothly.

Next steps you might take to make it more successful:

-LED’s not in the visible spectrum to reduce noise

-Object avoidance

- implementing more sophisticated control (PID, PI, etc)

***See code and video included in repository**

Summary of team members' contributions to the project:

Carrie Cramer:

-software component, debugging program, soldering, robot hardware assembly

Tarana Chowdhury:

-software component, debugging circuit, bread boarding, soldering