

IEEE/CETA Robotics Workshop

Lab Manual (Day 4) v002 How to Program the Inputs and Outputs

Dennis Cecic, P. Eng.
TISP Coordinator, IEEE Toronto Section
(d.cecic@ieee.org)

Brad North
Computer Engineering Teacher
Rick Hansen Secondary School
(brad.north@peelsb.com)





Disclaimer

The information in this workshop is distributed on an “as is” basis, without warranty. Although every precaution has been taken in the preparation of this work, neither the author(s) nor IEEE shall have any liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the information contained in this work.



Introduction – IEEE/CETA Robotics Workshop

Robot design and fabrication is an extremely rewarding and exciting activity for students of all ages.

Targeting high-school computer engineering technology teachers, this hands-on workshop uncovers the basic principles involved in construction and programming of a simple autonomous line following robot. The activity satisfies many aspects of the Ontario's TEJ/ICS curricula, and it is hoped that the teacher will be equipped to integrate this activity into his/her TEJ/ICS curriculum activities and also to field a team to compete in the annual CETA (Computer Engineering Teachers Association – Dufferin-Peel) Regional Robotics Competition.

The robot platform is based on a modern 16-bit microcontroller (PIC24), and is programmed using standard tools and languages **used in industry**. The robot platform provides a cost effective system for exploration of a variety of topics in TEJ/ICS curricula, such as analog/digital interfacing, motor control, computer networking, embedded control & mathematical algorithms, as well as both C and assembly language programming.

Training Objectives:

After the workshop the attendee shall,

- be able to develop embedded control C programs for the PIC24 MCU,
- be able to fabricate the robot platform used in the CETA competition,
- be able to configure the MCU inputs and outputs required for the robot,
- be able to implement a basic line-following control algorithm used in the CETA competition

The training consists of 5, 1-day modules, each of which consist of a PowerPoint presentation, with a corresponding instructional lab manual and lab projects. All materials are provided on the student CDROM.

Workshop Outline (Summary):

- Day 1. How to Program in C – Day 1
- Day 2. How to Program in C – Day 2
- Day 3. How to Build the Robot Platform
- Day 4. How to Program the Inputs and Outputs
- Day 5. How to Control the Robot

Prerequisites:

You should ideally have some basic experience in electronics prototyping as well as microcontroller programming, and be aware of basic safety rules when working with electronics.



Objective (Day 4)

The objective of this module is to cover the fabrication and basic programming of the electronic controls for the robot.

Training Objectives:

After this module, the attendee shall,

- understand the PIC24 Architecture and Programmer's Model, and how to create embedded C programs for it, using Microchip's embedded development tools,
- be able to fabricate the basic electronic control circuit for the robot,
- be able to write a program to drive a small DC motor,
- be able to modify the program to accept optical sensor input to detect a line

Detailed Outline:

Robot Platform

Firmware Development

Lab 1 – Wiring the Board

16-bit PIC® Architecture & Programmer's Model

MPLAB® C for PIC24 Compiler Overview

C-Language Extensions for the PIC24

Lab 2 – Creating PIC24 C Projects in MPLAB

Working with Digital Input and Output Ports

Basics of Brushed DC Motors and Their Control

PIC24 Timer & Pulse-Width Modulation Peripherals

Lab 3 – Motor Control using the PWM Peripheral

Introduction to Sensors and Analog Signals

Optical Sensors used in Line Following Robots

Analog-to-Digital Converter Peripheral

Lab 4 – Line Detection Using the ADC



Release Notes

v001 Summary (11 Feb 2011):

- Initial release of Day 4 material (outline 004). Targeted for ACSE PIC24 workshop in Toronto on February 11, 2011. Coverage limited to Labs1-3.
- Schematic and bill of materials for this release are provided in Appendix A & B respectively.

v002 Summary (11 Nov 2011):

- Simplified robot platform: uses 5V PIC24F MCU, single battery pack and L293D motor driver.
- Platform is completely solderless.
- Targeted for ACSE/CETA workshop in Mississauga (Rick Hansen SS) on November 11, 2011
- Schematic and bill of materials for this design are provided in Appendices A & B respectively.



Initial Installation/Set-Up

Purpose:

Set-up your PC to run the labs.

Required Equipment & Software:

1. PC running Windows 7 or Windows XP Professional with Service Pack2 (SP2) or higher. The PC should have at least 2 available USB ports.
2. Adobe Acrobat PDF Reader
3. Line Follower Robot Component Kit (see **APPENDIX B**), includes the “PICkit 3” programmer/debugger.
4. Day 4 Student files (from CDROM or on-line)
5. Pliers & wire cutter/stripper
6. MPLAB 8 IDE v8.73a (www.microchip.com/mplab) & MPLAB C for PIC24 Compiler v3.30b “Lite” version (www.microchip.com/mplabc) - Both are available on the Student CDROM in the “/Development Tools” sub-folder
7. Basic multimeter (dc voltage/current + frequency counter function)
8. **(Recommended)** Oscilloscope.
9. **(Recommended)** Solder Station (suggest Weller WD1001)

Software Installation Procedure:

1. Using Windows’ standard program uninstallation procedure, uninstall any existing MPLAB IDE and/or MPLAB C for PIC24 compiler versions.
2. Copy all the files/folders from the student CDROM to **C:\IEEE\IEEECEA_WS**
3. Install (i) the MPLAB IDE and (ii) MPLAB C for PIC24 Compiler (use the default paths). Double-click on the self-extracting .EXEs found in \Development Tools



Class Folder Structure

Root Folder: "C:\IEEE\IEEEECETA_WS"

"C:\IEEE\IEEEECETA_WS\Day 4\Lab1..Lab4"

Contains the Class Labs – Solution projects/workspaces are provided in the **\Solution** sub-folder

"C:\IEEE\IEEEECETA_WS\Day 4\Presentation & Handouts"

Contains .pdf copies of the slides and lab manual.

"C:\IEEE\IEEEECETA_WS \Users Guides & Data Sheets"

Repository for helpful documents while doing the labs, such as MCU data sheets and development board schematics.

"C:\IEEE\IEEEECETA_WS \Development Tools"

Contains the MPLAB IDE and PIC24 C-Compiler, plus any other useful tools used in the training.



Lab 1 – Wiring the Board

Purpose:

To understand the basic fabrication procedure of the line follower robot's control circuitry.

First, you will wire up the basic circuit as indicated in the schematic (Appendix A) by carefully following the example fabrication photos in this lab manual. Next, you will insert the Motors, Sensors and battery pack. Finally, you will use MPLAB and PICKit 3 to program a working .hex file into the PIC24 device to test your completed circuit. You will confirm the operation of your circuit by controlling both motor's speed using the potentiometer. The LEDs will indicate line detection by the Left, Middle and Right opto sensors.

Procedure:

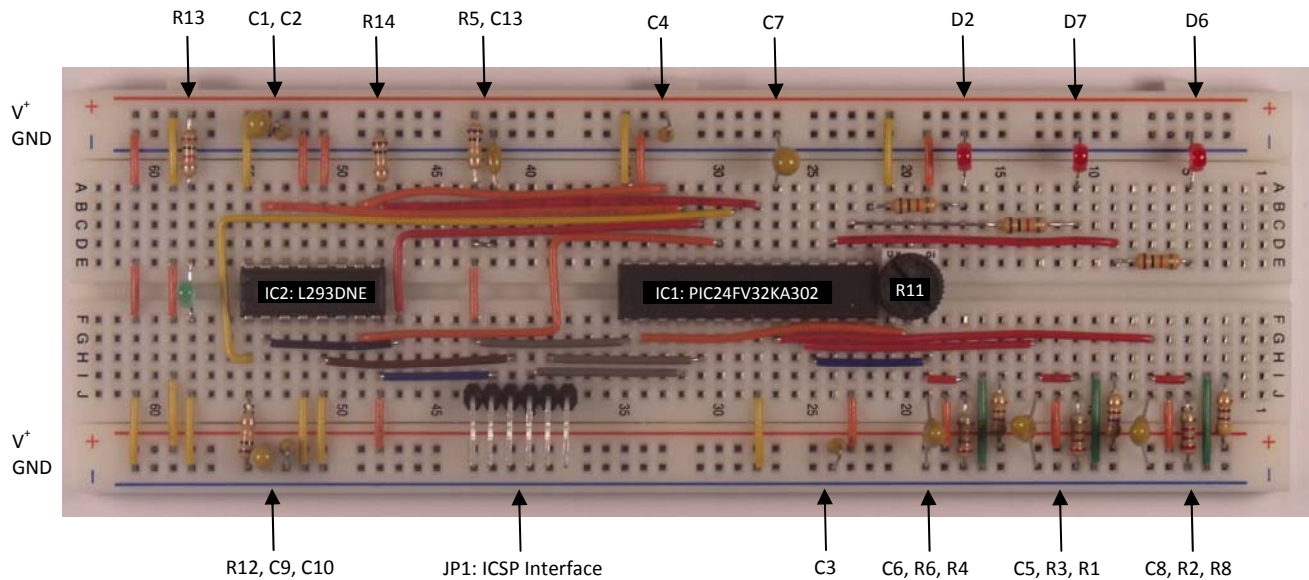
1. Using wire strippers/cutter, cut an 18cm piece of Red and Black #22AWG wire from the spools. Carefully strip off about 1cm of insulation from each end of the wires. Using a soldering iron, tin the exposed wire (get some help if you are new to soldering), twist the wires together and solder them to the DC motors as shown here (do not install the wheel):



2. Assemble the control circuit (see Appendix A for the schematic). Suggest you follow the fabrication in 3 steps, as outlined in the following sequence.



3. Insert the basic wiring and circuitry as shown. Key component designators are identified (refer to the schematic in Appendix A). Suggest you use the Acrobat viewer to view the .pdf version of the lab manual:



Design & Assembly Notes/Tips (Refer to schematic in Appendix A):

Wiring: Most of the wires shown are from the wiring kit (pre-cut and bent). Use these to save time. Some of these wire connections will need to be custom-made.

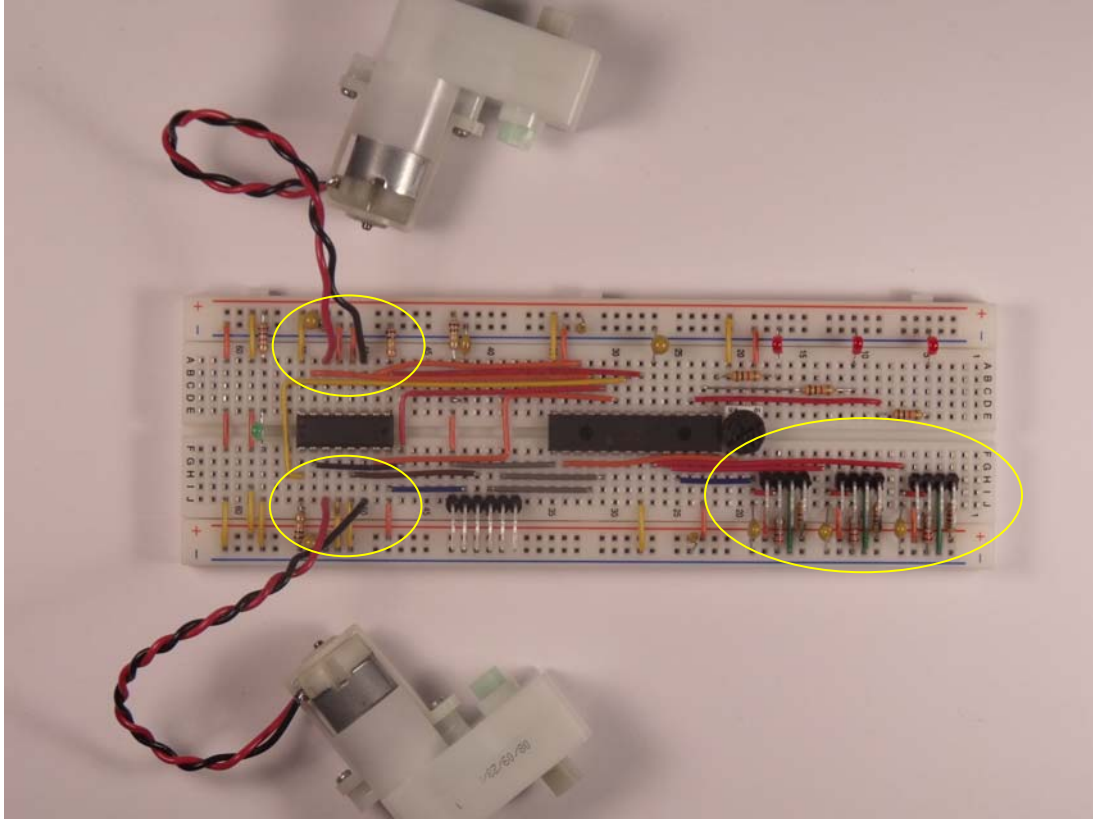
C1, C9, C7, C8, C5, C6: These capacitors are polarized (i.e. there is a “+” side and a “-” side). Follow the schematic carefully and connect the “+” lead to the correct node.

LEDs D8, D2, D7, and D6: The anode-side (positive side) of this component is the **longer** lead.

JP1 (ICSP Interface): You will need to break off a 6-pin piece from the 36-lead connector in your component kit.

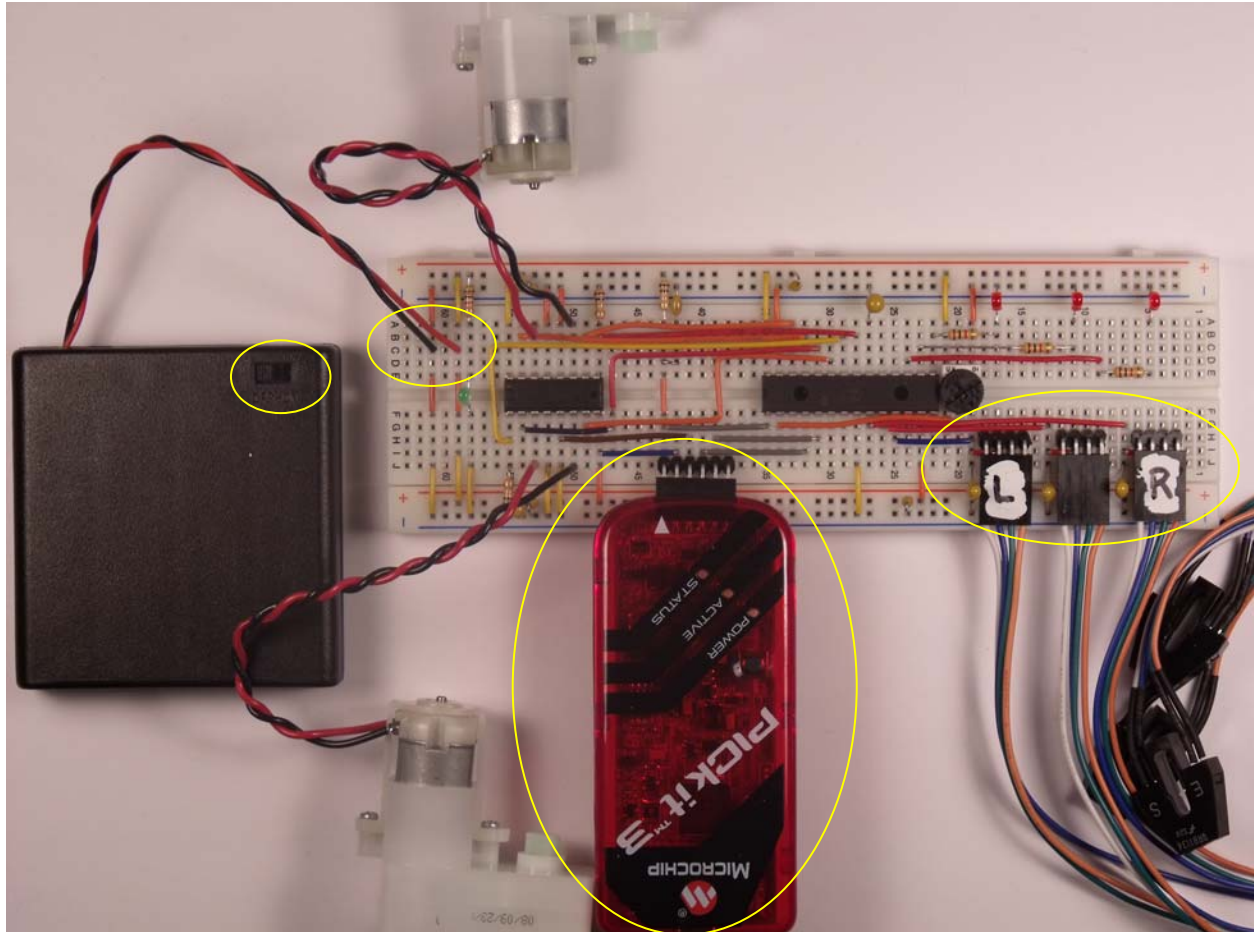


4. Next, break off 3, 4-pin headers, and insert these as shown. Also, insert the 2 motor leads as shown:





5. Finally, connect the Opto Sensors and battery pack (twist the battery pack leads as shown and **make sure the power switch is OFF**). Finish off by plugging in the PICkit 3 programmer as shown:

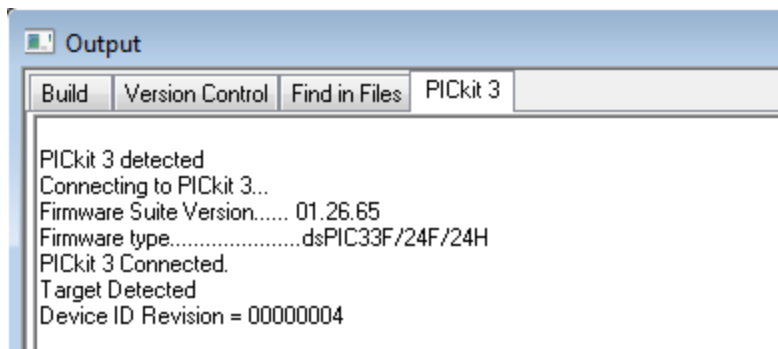


Design & Assembly Notes/Tips:

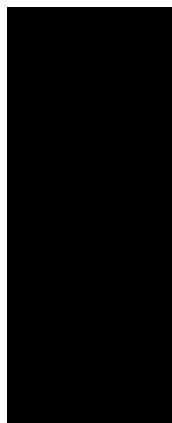
Why twist the leads? Wires are components! They have resistance, capacitance and inductance. The leads from the battery pack to the circuit and from the motor driver to the motor can carry up to 1A, and they require this current to change up/down rather rapidly. Twisting the wires reduces the inductance of these leads, making it easier for the current to change.



6. Test the completed assembly by importing/programming a .hex file into the MCU:
 - a. Turn on the battery pack's power switch. The Green LED (LED D8) should light.
 - b. Start MPLAB, and set the processor to PIC24FV32KA302 (**Configure ▶ Select Device ▶ PIC24FV32KA302**)
 - c. Select the PICkit 3 as the Programmer (**Programmer ▶ Select Tool ▶ PICKit 3**). If MPLAB detects that a firmware upgrade to your PICkit 3 is available and required, let it install any upgrades.
 - d. Successful connection to target will be indicated in the output window:



7. Import the .hex file (**File ▶ Import "C:\IEEE\IEEECEA_WS\Day 4\Lab 1\Solution\Lab 1.hex"**)
8. Program the device (**Programmer ▶ Program**). The program will start executing right after.
9. The motors should start to spin!
10. Using a screwdriver, rotate the potentiometer R11 to control the speed of the motors.
11. Move each opto-sensor ~1/2cm over the black line printed below and verify that it detects the line by viewing its status on the LEDs (Left-D2, Middle-D7, Right-D6). The LED should light when over the line. If it doesn't, this means it could require re-calibration, which you will do in Lab 4.





Lab 2 – Creating PIC24 C Projects in MPLAB

Purpose:

To learn how to create a C-based project from scratch for PIC24 using the MPLAB C Compiler for PIC24 (“MPLAB C30”). This project will blink LED D6.

A solution workspace is provided: C:\IEEE\IEEECEETA_WS\Day 4\Lab 2\Solution\Lab2.mcw

Procedure:

1. MPLAB IDE and C30 must be installed in their default directories. The hardware for this lab should already be constructed (see Lab 1). The student CDROM must be copied into C:\IEEE\IEEECEETA_WS
2. Using Windows Explorer, copy header file "p24FV32KA302.h" from "C:\Program Files\Microchip\mplabc30\v3.30b\support\PIC24F\h" to C:\IEEE\IEEECEETA_WS\Day 4\Lab 2
3. Using Windows Explorer, copy linker script file "p24FV32KA302.gld" from "C:\Program Files\Microchip\mplabc30\v3.30b\support\PIC24F\gld" to C:\IEEE\IEEECEETA_WS\Day 4\Lab 2
4. Start MPLAB IDE, and create a new c-source code file for the main() function:
 - a. **File ▶ New**
 - b. Type the following source code into the new file (leave out the comments to save time):



```
/** #include files *****/
#include "p24fv32ka302.h"

/** Symbolic Constants used by main() *****/
#define MS_DELAY_VAL 100          // Delay parameter

/** Local Function Prototypes *****/
void DelayMs(unsigned int ms);    // In-line delay of up to 65536 mS

/** Configuration Bit Macros *****/
_FOSCSEL(FNOSC_FRC & IESO_OFF)
_FOSC(POSCMOD_NONE & OSCIOFNC_OFF)
_FWDT(FWDTEN_OFF)
_FICD(ICS_PGx1)
_FDS(DSWDTEN_OFF)

/** Global Variable Declarations *****/
int count;

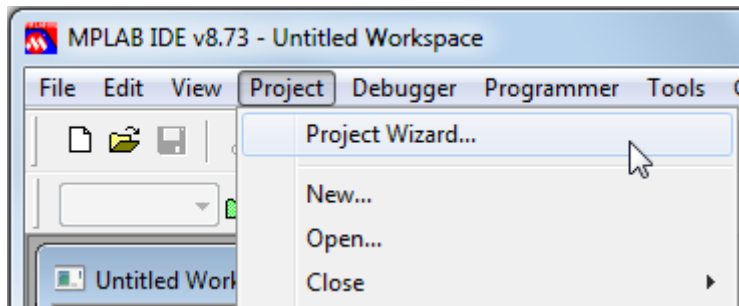
/** main() *****/
int main(void)
{
    // Initialize LED D6 (RB8)
    LATBbits.LATB8 = 0;    // Initial level to place on pin
    TRISBbits.TRISB8 = 0; // Make the pin a digital output

    while(1)
    {
        count++;
        LATBbits.LATB8 = 1;    // Turn on LED D6
        DelayMs(MS_DELAY_VAL); // Delay
        LATBbits.LATB8 = 0;    // Turn off LED D6
        DelayMs(MS_DELAY_VAL); // Delay
    }
}

/** DelayMs(); *****/
void DelayMs(unsigned int ms)
{
    while(ms--)
    {
        asm("repeat #4000"); // 4000 instruction cycles @ 250nS ea. = 1mS
        asm("nop");         // instruction to be repeated 4000x
    }
}
```

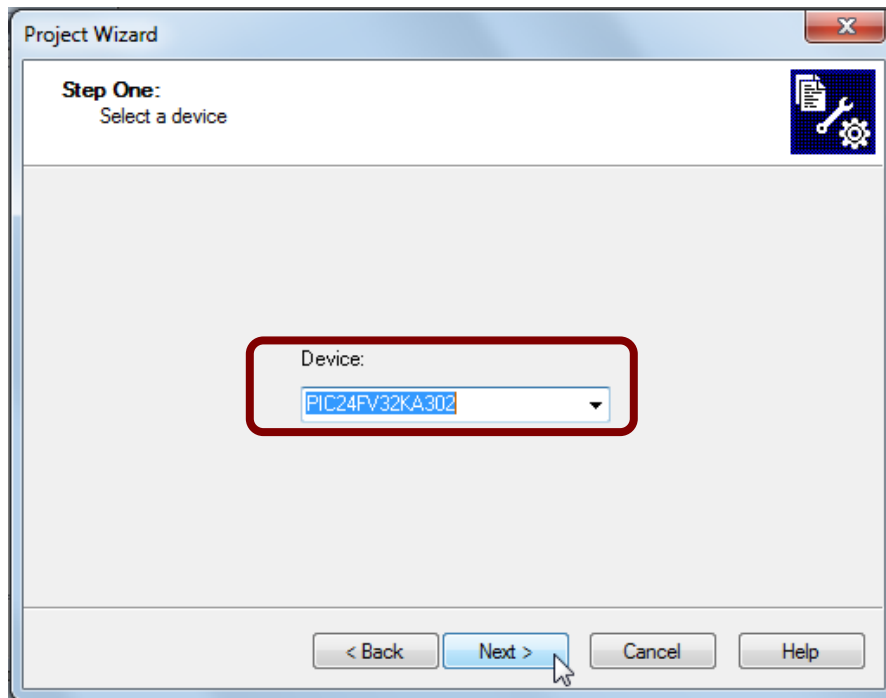


- c. Save the new file: (**File ▶ Save As**) “main.c” in C:\IEEE\IEEECEETA_WS\Day 4\Lab 2
- 5. Start the “Project Wizard” which will guide you through the remaining steps to create an MPLAB IDE project and workspace (select **Project ▶ Project Wizard**)



After the project wizard opens, click “Next” to continue

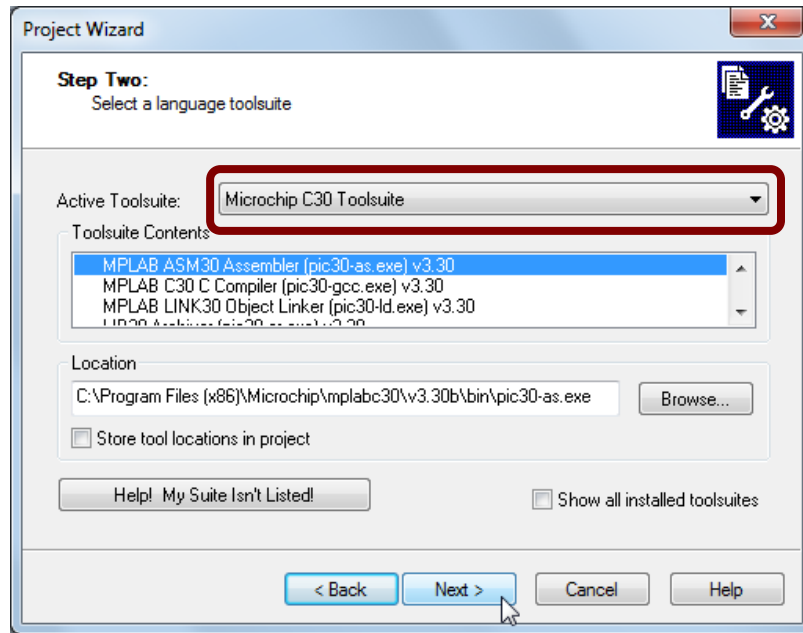
- 6. In the “Device” selection box, confirm that **PIC24FV32KA302** is selected as device



After you do this, click “Next” to continue



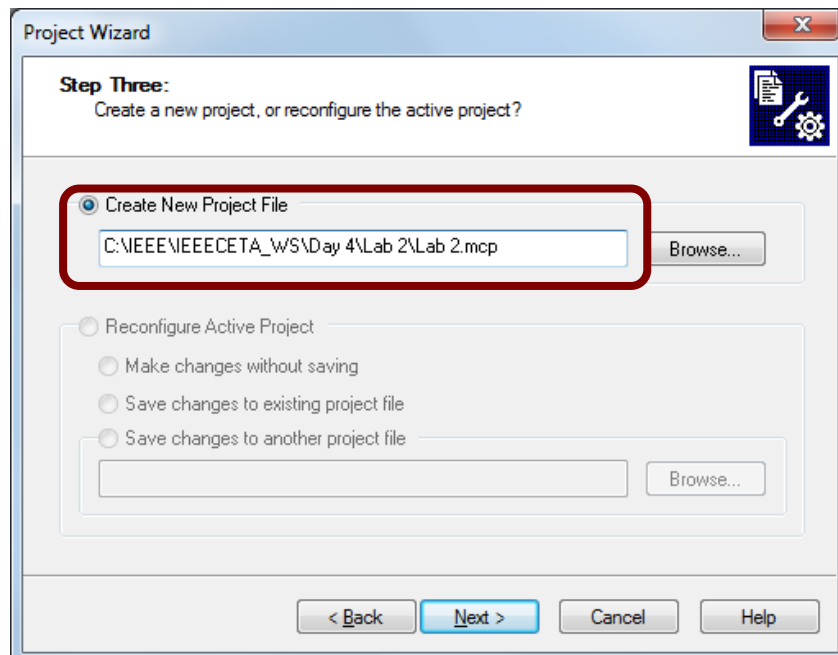
7. In the “Active Toolsuite” selection box, select **Microchip MPLAB C30** as the active tool suite.



If there are red X's beside any of the toolsuite files (compiler, linker etc), you will need to select the tool and “Browse” to the location where you instructed the MPLAB Compiler to install the files.

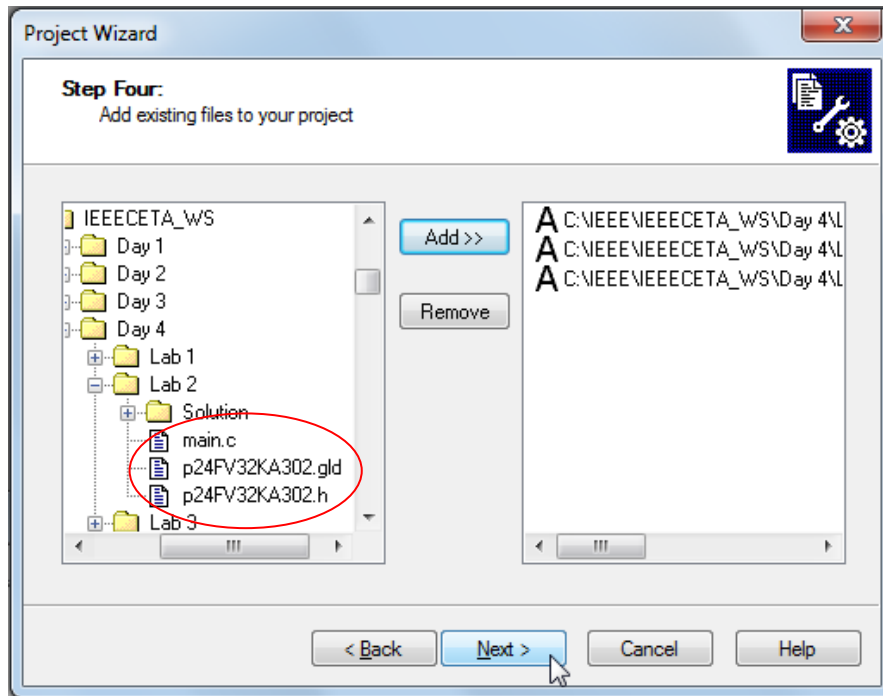
Click “Next” to continue

8. Enter the following project name in the box: **C:\IEEE\IEEECEETA_WS\Day 4\Lab 2\Lab 2.mcp**

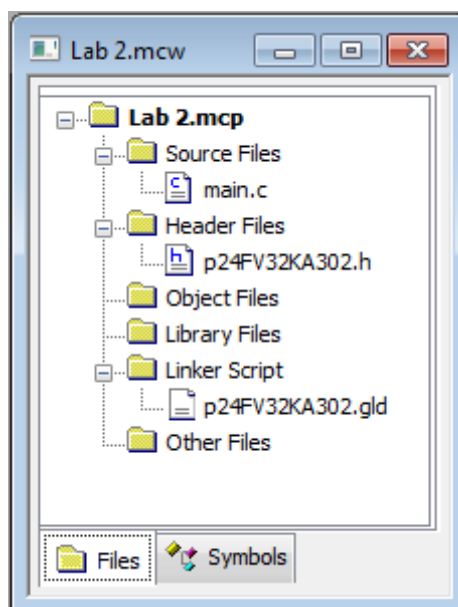




9. In the left pane, navigate to **C:\IEEE\IEEEECETA_WS\Day 4\Lab 2**. Select each one of the 3 displayed files and press “Add” to add them to the project (main.c, p24FV32KA302.h, p24FV32KA302.gld):

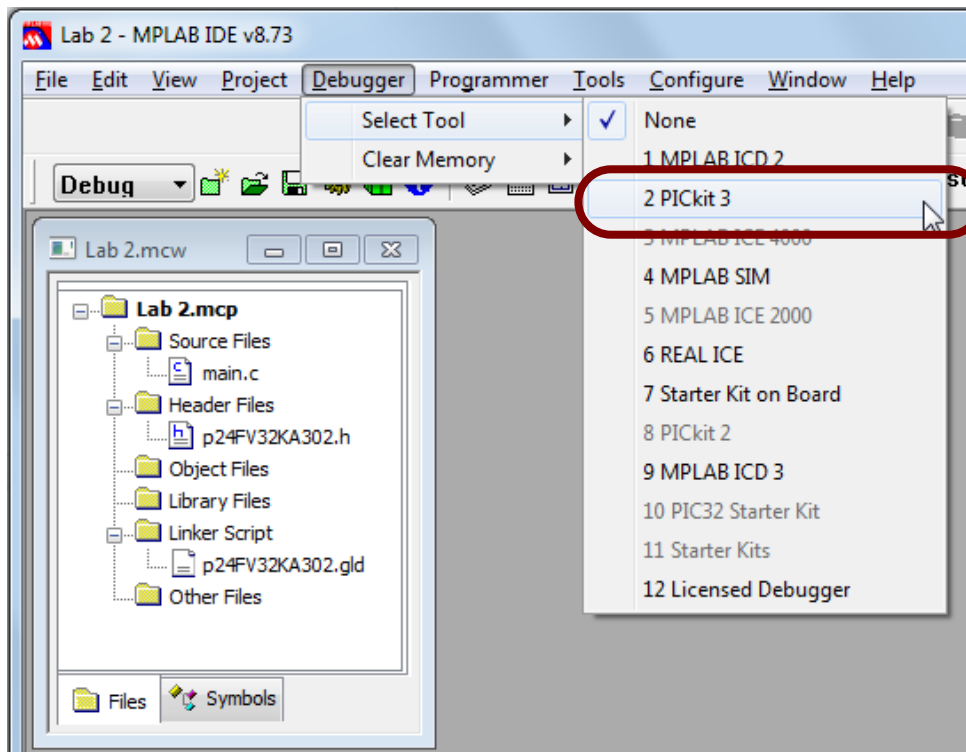


10. Click “Finish”. Direct MPLAB to save the workspace as “Lab 2.mcw”. If the project tree isn’t visible, select from the menu (**View ► Project**). You should see the three files used in this project.





11. Connect to the PICKit 3 as the debugger (**Debugger** ▶ **Select Tool** ▶ **PICKit 3**)




The output window (**View** ▶ **Output**) should show a successful connection to the target via the PICKit 3 (make sure to turn on the power switch on the battery pack).

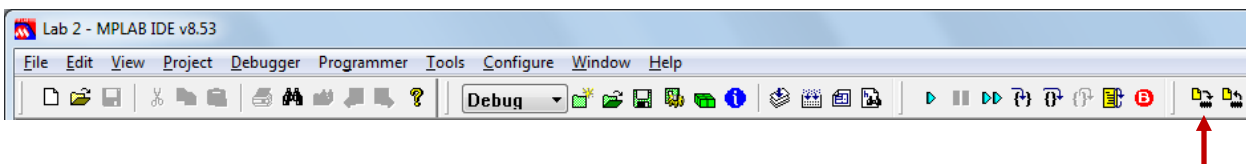
12. Select/Enable a “Debug” code build:




13. Click the **Build All** button 




14. If no build errors are reported, program the PIC24 by clicking the **Program** button 

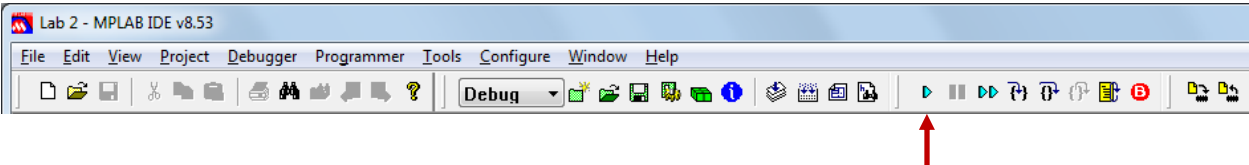





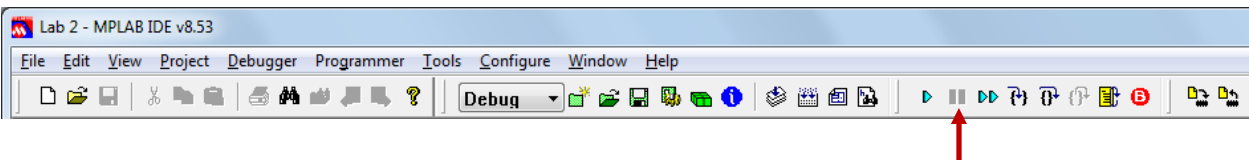
15. When programming completes, click the **Reset** button to reset the PIC24 



16. To run the program, click the **Run** button 



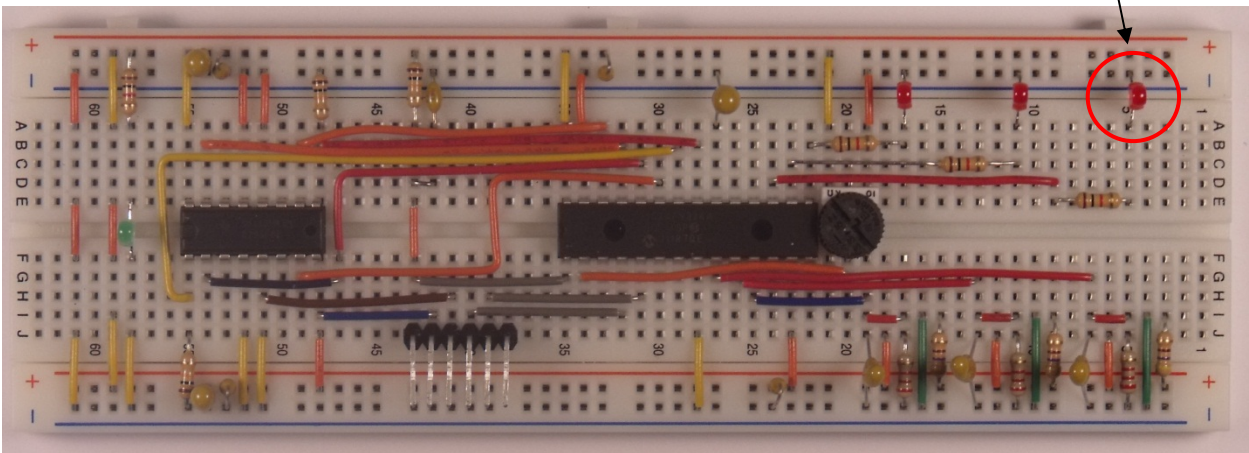
17. To stop the program, click the **Halt** button 



To startup again, click **Reset**, followed by **Run**.

Results:

LED Blinks!



To learn more about debugging in the MPLAB IDE, please refer to the document “16-Bit Language Tools Getting Started (DS70094E)” guide available in C:\Program Files\Microchip\mplabc30\v3.30b\docs



Lab 3 – Motor Control using the PWM Peripheral

Purpose:

Experiment with controlling the motor direction using a working project. The operating modes of the L293D are reproduced here:

1-2EN (PWM)	1A (Control)	2A (Control)	1Y (Output)	2Y (Output)	Motor Action
1	0	0	0	0	Brake (low-side)
1	0	1	0	1	Forward
1	1	0	1	0	Reverse
1	1	1	1	1	Brake (high-side)
0	x	x	High-Z	High-Z	Coast

Procedure:

1. Start MPLAB IDE
2. Open the solution workspace C:\IEEE\IEEECEETA_WS\Day 4\Lab 3\Solution\Lab 3.mcw
3. Build/Program and Run the project. Observe the rotation direction of the motor.
4. Open main.c and find the initialization function Initialize(). Find the 1A/2A and 3A/4A control outputs (RB13/RB12 and RB15/RB14). Change the setting per the table above and reverse the direction of the motor.
5. Build/Program and Run the project. Observe the rotation direction of the motor. It should be reversed.



Lab 4 – Line Detection using the ADC Peripheral

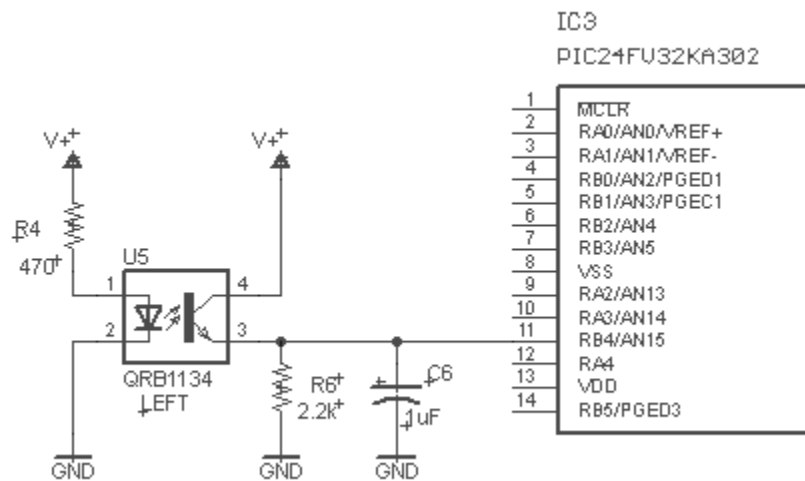
Purpose:

One of the main functions of the line-following robot firmware is to sense a black line path on a white board. This sensed input is then used to control the motor drive to the 2 rear wheels.

This task may be accomplished using a phototransistor reflective optosensors, such as the QRB1134. These sensors are typically positioned in front of the robot.

The reflective sensor consists of an infrared emitting diode and an NPN silicon phototransistor mounted side by side on a converging optical axis in a black plastic housing. The phototransistor responds to radiation from the emitting diode only when a reflective object passes within its field of view.

The following is a basic circuit using such a sensor:



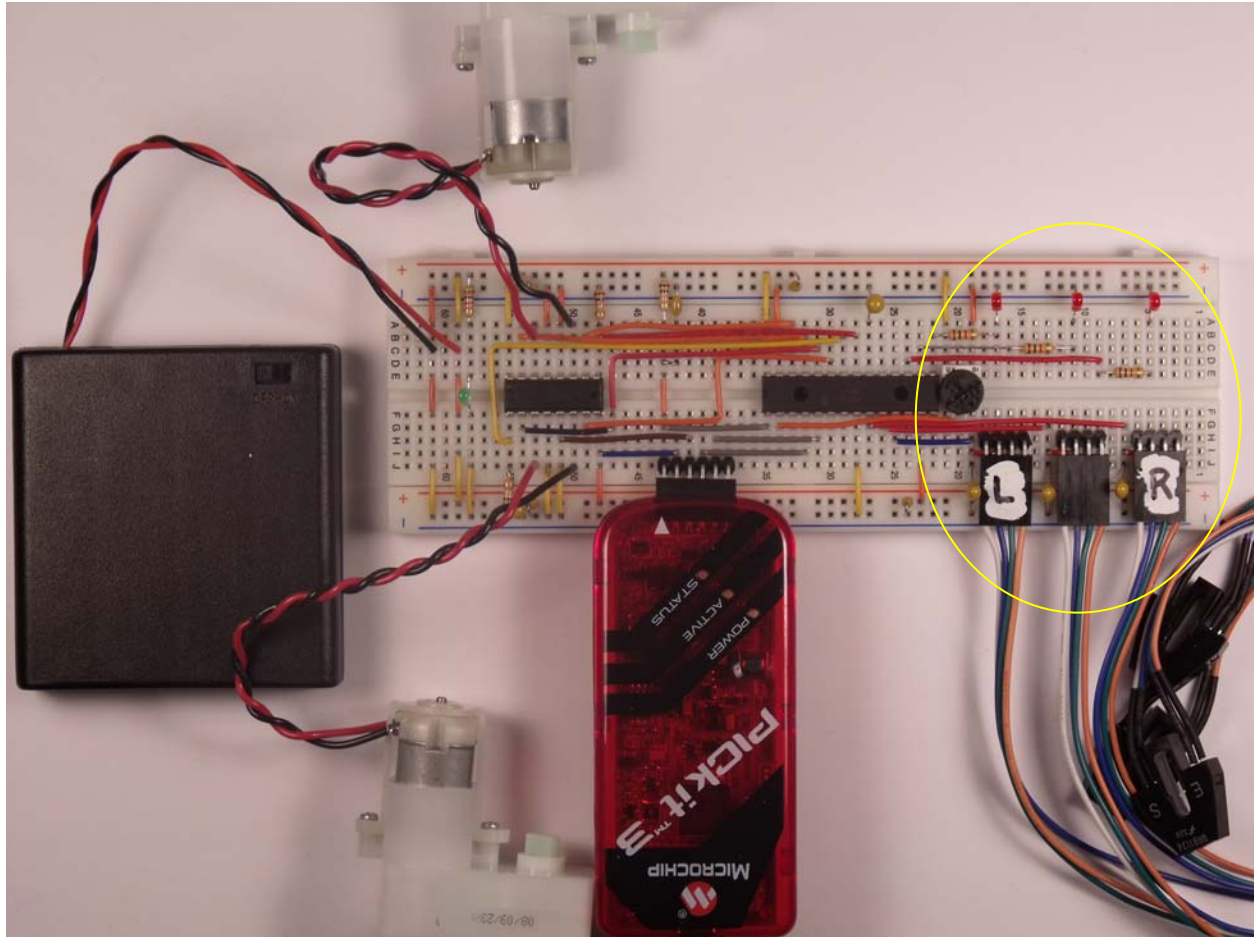
Here is a pin-out diagram for the QRB1134:

Schematic Pin Number	Function	Wire Color
1	(A) Anode	Orange
2	(K) Cathode	Green
3	(E) Emitter	Blue
4	(C) Collector	White

The voltage at the sensor output (Pin 3) represents the amount of reflected IR light. In this lab, you will learn how to use the built-in Analog-to-Digital Converter (ADC) in the PIC24F MCU to digitize this signal (i.e. turn it into a number that the firmware can use). Black surfaces will absorb light, while white surfaces will reflect, so we expect the signal to be mostly “high” while the sensor is hovering over a white background, and going “low” when it hovers over the black line.



A basic working line detection firmware is provided that emulates a comparator circuit: when the voltage at Pin 3 of the optosensor drops below a certain threshold, the optosensor's corresponding LED will be turned on.



The first part of the lab is a calibration step where you will use the PIC24 ADC to measure the sensor's voltage response to black/white surfaces (Pin 3) and calculate the optimum trip voltage.

Procedure:

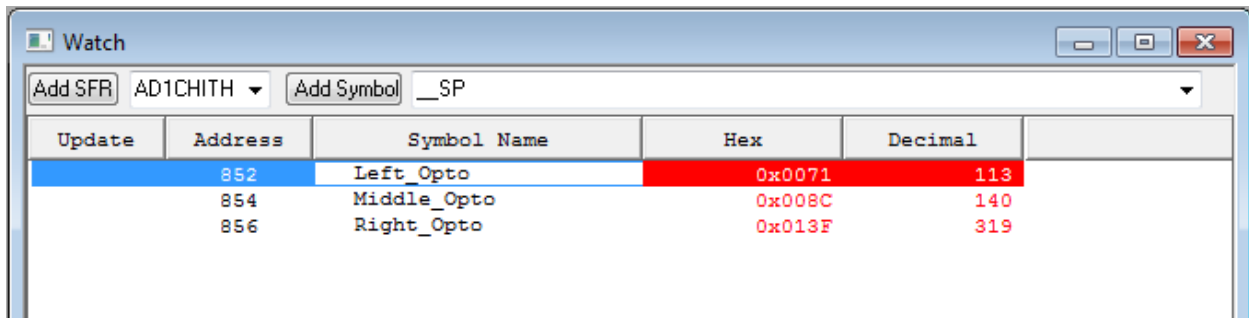
1. Close the previous MPLAB workspace. Turn off the battery to stop the motors, or remove their leads from the breadboard..
2. Connect the optosensors to the board as shown in in the picture above.



3. Start MPLAB and open the workspace “C:\IEEE\IEEECEA_WS\Day 4\Lab 4\Solution\Lab 4.mcw”
4. Open main.c in an editor window. Place a breakpoint on line 89 (the “Decide()” function) by double-clicking on that line.

```
85 //infinite loop
86 while(1)
87 {
88     Get_Inputs();
89     Decide();
90     Do_Outputs();
91     Timing();
92 }
```

5. This breakpoint is where the CPU will halt during debugging, allowing you to examine the ADC values of the optosensor signals value. A watch window has already been set up to view the 3 variables which will hold the optosensor readings:



Update	Address	Symbol Name	Hex	Decimal
	852	Left_Opto	0x0071	113
	854	Middle_Opto	0x008C	140
	856	Right_Opto	0x013F	319

6. Build/Program a “debug” build:

Select “Debug” Build configuration:
Select the PICKit3 as Debugger:
Build the Project:
Program the device:

**Project ▶ Build Configuration ▶ Debug
Debugger ▶ Select Tool ▶ PICKit3
Project ▶ Build All
Debugger ▶ Program**



- 7. For each optosensor:
 - a. Hold it ~1/2cm over the black line printed below. Run the code (**Debugger ▶ Run**). When the CPU halts at the breakpoint, record the value in the table below. Next, reset the CPU (**Debugger ▶ Reset ▶ Processor Reset**), move the sensor over the white background and repeat the measurement (run the code). Record the decimal values in the table below.



Sensor	ADC Value (Over Black)	ADC Value (Over White)
LEFT		
MIDDLE		
RIGHT		

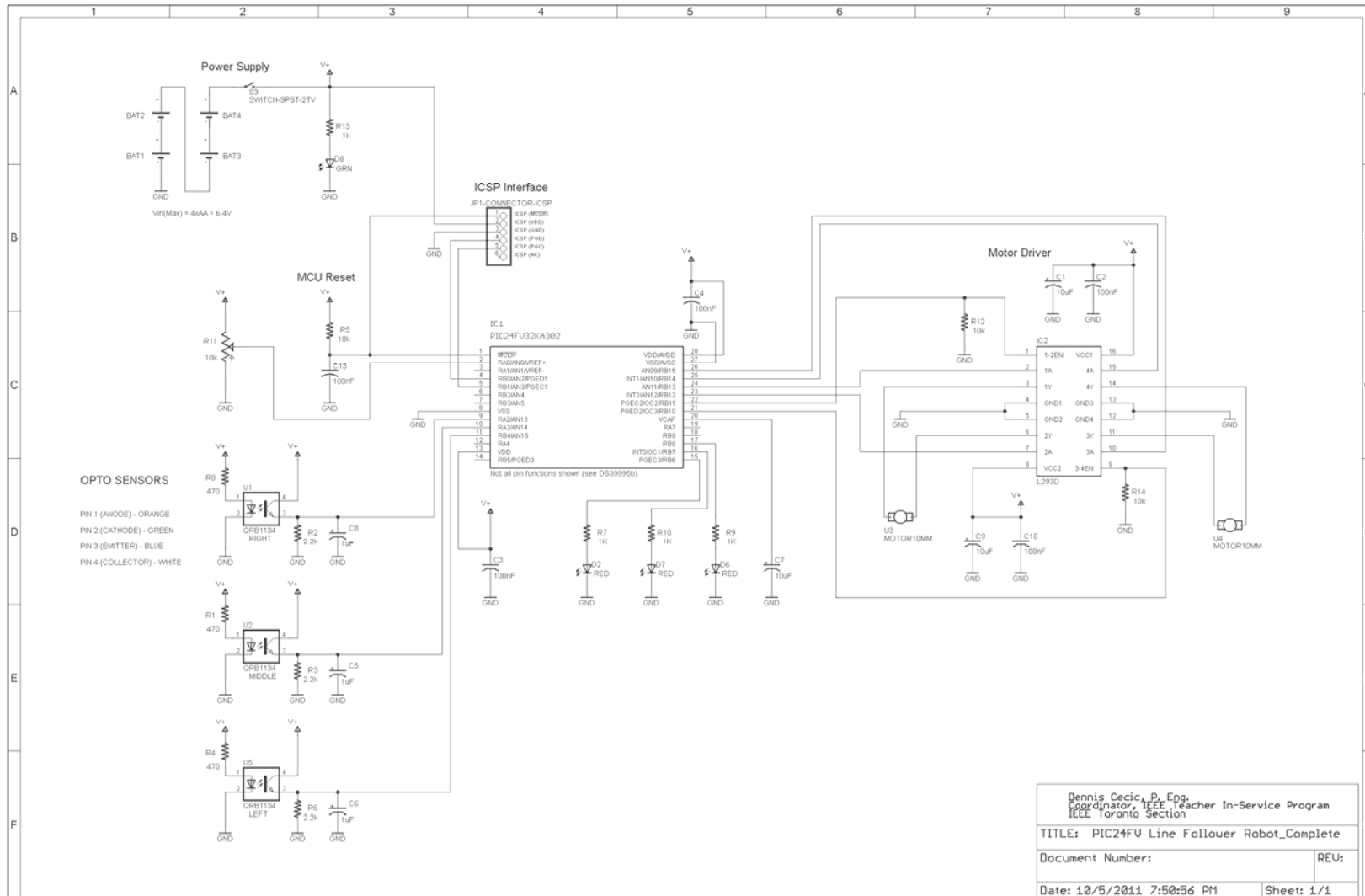
- 8. Use these values to define the appropriate threshold trip values on lines 38, 39 and 40 of main.c
- 9. Rebuild/Run the code in “Release Mode”

Select “Release” Build configuration: **Project ▶ Build Configuration ▶ Release**
 Select the PICKit3 as Programmer: **Programmer ▶ Select Tool ▶ PICKit3**
 Build the Project: **Project ▶ Build All**
 Program the device: **Programmer ▶ Program**

Does it work? Position the sensors as you did during calibration. Are there now shadows beneath the sensors that were not there during calibration? Are you positioning the sensor at the same height?



Appendix A – Line Follower Robot Electronic Control Circuit Schematic





Appendix B – Electronic Control Circuit Component Kit

Component Description	MFR Part Number	Digikey Part Number	Qty.	Notes
MCU 32KB FLASH 2KB RAM 28-SPDIP	PIC24FV32KA302-I/SP	PIC24FV32KA302-I/SP-ND	1	Academic pricing is available from Digikey
PROGRAMMER MCU PICKIT 3	PG164130	PG164130-ND	1	Academic pricing is available from Digikey
BREADBOARD 2.13x6.496 SLDLESS	TW-E40-1020	438-1045-ND	1	
WIRE SET 140PC FOR BOARD	TW-E012-000	438-1049-ND	1	
CAP CER .1UF 50V 10% AXIAL	C412C104K5R5TA7200	399-4484-1-ND	5	
CAP TANT 10UF 16V 10% RADIAL	TAP106K016SCS	478-1839-ND	3	
CAP TANT 1UF 25V 10% RADIAL	T350A105K025AT	399-3528-ND	3	
HOLDER BATT W/COVR 4AA ON/OFF SW	SBH-341AS	SBH-341AS-ND	1	
POT 10K OHM THUMBWHEEL CERM ST	3352E-1-103LF	3352E-103LF-ND	1	
BATTERY IND ALKALINE AA SIZE	EN91	N107-ND	4	
IC QUAD HALF-H DRVR 16-DIP	L293DNE	296-9518-5-ND	1	
143:1 GEAR MOTOR + WHEEL	GM8 + GMPW-B	N/A	2	"GM8PW Deal" (Solarbotics: www.solarbotics.com)
QRB1134 OPTO Emitter/Detector	IRROSC-2	N/A	2	2 sets gives 4 sensors total (Digilent: www.digilentinc.com)
CONN HEADER .100 SINGL R/A 36POS	PEC36SBCN	S1132E-36-ND	1	
RES 10K OHM 1/4W 5% CARBON FILM	CFR-25JB-10K	10KQBK-ND	5	
RES 1.0K OHM 1/4W 5% CARBON FILM	CFR-25JB-1K0	1.0KQBK-ND	5	
RES 470 OHM 1/4W 5% CARBON FILM	CFR-25JB-470R	470QBK-ND	5	
RES 2.2K OHM 1/4W 5% CARBON FILM	CFR-25JB-2K2	2.2KQBK-ND	5	
LED 3.1X2MM 650NM RED DIFFUSED	SLR-322VR3F	511-1228-ND	3	
LED 3.1X2MM 563NM GREEN DIFFUSED	SLR-322MG3F	511-1226-ND	1	
#22AWG Black Stranded (6")	3051 BK005	A2016B-100-ND	1	100' Price = \$41.23
#22AWG Red Stranded (6")	3051 RD005	A2016R-100-ND	1	100' Price = \$41.23