TI-RSLK MAX

Texas Instruments Robotics System Learning Kit

TEXAS INSTRUMENTS

Module 4

Introduction: Software Design Using MSP432

Introduction: Software Design Using MSP432

Educational Objectives:

REVIEW C programming
UNDERSTAND conditional statements and loops in C
DEVELOP logic and arithmetic functions
LEARN how to debug simple programs in C
DESIGN, BUILD & TEST A SOFTWARE COMPONENT

As the robot explores its world, it must make decisions. In the lab associated with this module, you will write software that takes input from three distance sensors and determines if one of eight possible scenarios is present, see Figure 1. The actual sensors will be interfaced in Lab 15, but in this lab you will write software to be used in the robot later.

Prerequisites (Module 1)
- Running code on the LaunchPad using CCS (Module 1)

Recommended reading materials for students:
- Chapter 4, **Embedded Systems: Introduction to Robotics**, Jonathan W. Valvano, ISBN: 9781074544300, copyright © 2019

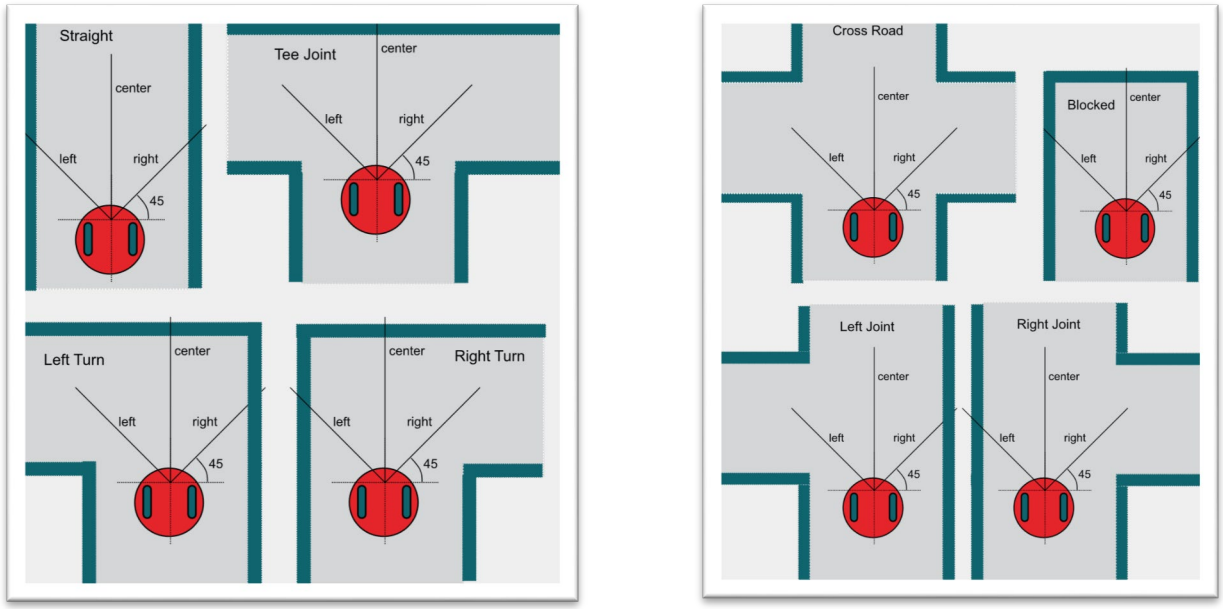

Figure 1. As the robot is exploring the maze, it encounters these eight possible scenarios. When the robot approaches an intersection, it first determines what alternative paths exist, and then it chooses which way to go

Introduction: Software Design Using MSP432

This module serves as a brief introduction to C. C is a general-purpose programming language initially developed by Dennis Ritchie between 1969 and 1973 while at AT&T Bell Labs. In 1999, a professional standard version of C, called C99, was defined. In this class we will write our software in C99, because it is prevalent in industry.

A **compiler** is system software that converts a high-level language program (human readable format) into object code (machine readable format). It produces software that is fast but to change the software we need to edit the source code and recompile.

The Project Explorer in CCS shows us the various components used for each project. A **linker** builds a single software system by connecting (linking) software components. In CCS, the **build** command performs both a compilation and a linking.
In an embedded system, the **loader** will program object code into flash ROM. We place object code in ROM because ROM is retains its information if power is removed and restored. In CCS, the **Debug** command performs a load operation and starts the debugger.

A **debugger** is a set of hardware and software tools we use to verify system is operating correctly. The two important aspects of a good debugger are control and observability.

Before we write software, we need to develop a plan. Software development is an iterative process. Even though we list steps the development process in a 1,2,3,4 order, in reality we cycle through these steps over and over. I like to begin with step 4), deciding how I will test it even before I decide what it does.

1) We begin with a list of the inputs and outputs. This usually defines what the overall system will do. We specify the range of values and their significance.
2) Next, we make a list of the required data. We must decide how the data is structured, what does it mean, how it is collected, and how it can be changed.
3) Next we develop the software algorithm, which is a sequence of operations we wish to execute. There are many approaches to describing the plan. Experienced programmers can develop the algorithm directly in C language. On the other hand, most of us need an abstractive method to document the desired sequence of actions. Flowcharts and pseudo code are two common descriptive formats. There are no formal rules regarding pseudo code, rather it is shorthand for describing what to do and when to do it. We can place our pseudo code as documentation into the comment fields of our program. Next we write software to implement the algorithm as define in the flowchart and pseudo code.
4) The last stage is debugging. Learning debugging skills will greatly improve the quality of your software and the efficiency at which you can develop code.

In the lab associated with this module, you will develop and test some software functions that will be used later in the explorer robot. In particular, the first function will convert ADC measurements from a sensor into distance to the wall, and the second function will take three distance measurements and classify the situation into the most likely scenario.

ti.com/rslk

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale (www.ti.com/legal/termsofsale.html) or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265