Preface

Texas Instruments Robotics System Learning Kit
The Maze Edition

Preface

The ultimate goal of the learning kit is to design, build, and test a robot system capable of solving complex tasks. One possible robot is shown in Figure 1. Example challenges include exploring a maze, racing autonomously, finding an object, and following a line. However, it is not the final robot that matters, but the educational journey that discovers a wide range of engineering principles along the way. Rather than just providing the robot kit and a challenge to solve, this curriculum follows an educational road map that intentionally exposes deep learning along the way.

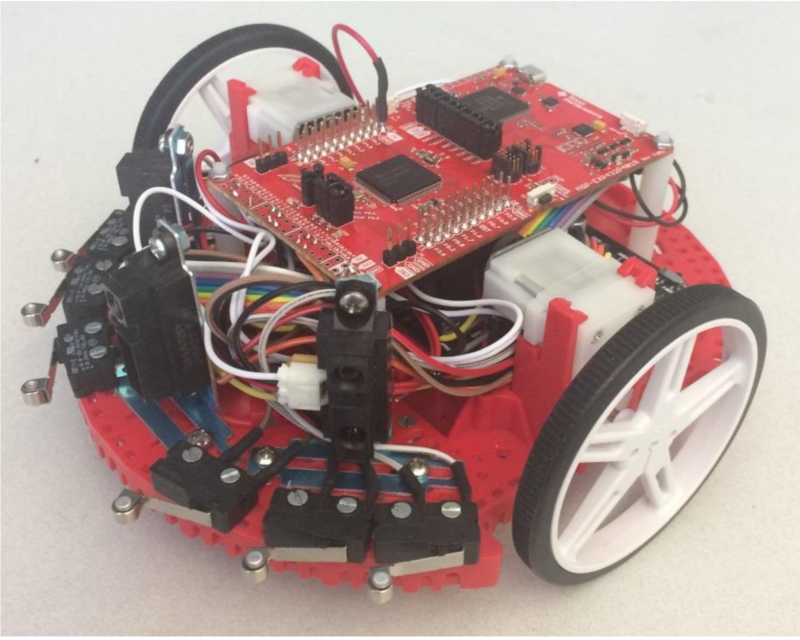

Figure 1: TI-RSLK Maze Robot

The **EE skills** you will learn include voltage, current, power, energy, batteries, resistors, capacitors, transistors, diodes, and DC motors. You will learn how to use a voltmeter, an ohmmeter, a current meter, and an oscilloscope.

This is an **embedded systems** curriculum; hence you will interface numerous devices to the MSP432 microcontroller. In particular, you will interface LEDs, switches, a line sensor, DC motors, tachometers, and an LCD. Your microcontroller hardware/software skills will include pulse-width modulation (PWM), flash read-only memory, periodic timers, edge-triggered interrupts, digital to analog converter (DAC), analog to digital converter (ADC), synchronous serial communication, and asynchronous serial communication.

A significant component of the curriculum involves software development. You will develop **software skills** in multithreading, data structures, debugging, linked lists, semaphores, and first in first out (FIFO) queues. You will learn how to use a logic analyzer for interface testing, and thread profiling.

This is a lab-based curriculum. However, there are numerous **fundamental concepts** to master, such as the Nyquist Theorem, the Central Limit Theorem, digital filtering, and Little's Theorem.

The overriding theme of this curriculum is to teach **systems design** in a bottom up fashion. We begin with simple components so that you learn fundamentals. A system is created by combining multiple components with the appropriate hardware and software interfaces. Once you master the fundamentals of one component, its operation can be abstracted into a set of high-level functions. Separating how a component works (low-level implementation) from what it does (high-level abstraction) is the key for developing complex systems. Obviously, the most important system in this curriculum will be the robot. However, there will be other systems like a security system, a traffic light control system using a finite state machine, Bluetooth Low Energy (BLE) communication system, and the Wi-Fi-based internet of things (IoT) system.

A system is comprised of subsystems connected together to solve a unified objective. An effective approach to teaching systems is to begin with very simple components. First, one completely understands how the component works. Second, one creates an abstraction that separates what is does from how it works. Third, components are interfaced together to create a new more complex system.

Preface

The terms system, subsystem, and component are used here interchangeably.

As you can see from Figure 2, there are twenty modules in the curriculum. Each module is relatively independent, and you can thread together modules to create a particular learning experience for your students.

Each module has:

- Introduction to module (1 page)
  - Overview
  - Educational objectives
  - Prerequisites, bullet list linking to other modules
  - References
- Class lecture PowerPoint slides (one to three files)
- Screen capture video with audio of PowerPoint (one to three videos)
- Class activity, homework exercises or practice problems.
- Lab document
- Hardware needed,
  - BOM excel file of parts
  - Circuit diagrams in CircuitMaker.

- Lab solution for faculty to access, not available to students.
- One to three videos of finished lab
- Quizzes
- Quiz solutions for faculty to access, not available to students.

The most important document is the lab manual. Performing labs results in the design, construction, and testing of the robot system. To find the circuit diagrams, create an account in Circuit maker. Launch the application, under projects select Tags, and search MSP432. You will find starter circuits for each lab that has hardware.

The robot challenge document lists some possible final projects for the course. Most users of this curriculum will pick and choose a subset of the modules, allowing the user to focus on which concepts they wish to learn (or teach). Challenges are sorted by the set of sensors and actuators that are required.

Robot Features (Full set, Advanced Kit) :

- Robot Chassis with 2 DC motors and wheels
- 6 AA NiMH batteries
- Motor driver and power distribution board (MDPD) with motor drivers and voltage regulator to power your system
- 3 IR distance sensors
- 6 touch/bump sensors
- 8 line sensors
- 2 tachometers
- Tachometer
- BLE or Wi-Fi

Course prerequisites:

- Algebra and college physics
- Basic knowledge of computers and architecture
- C programming

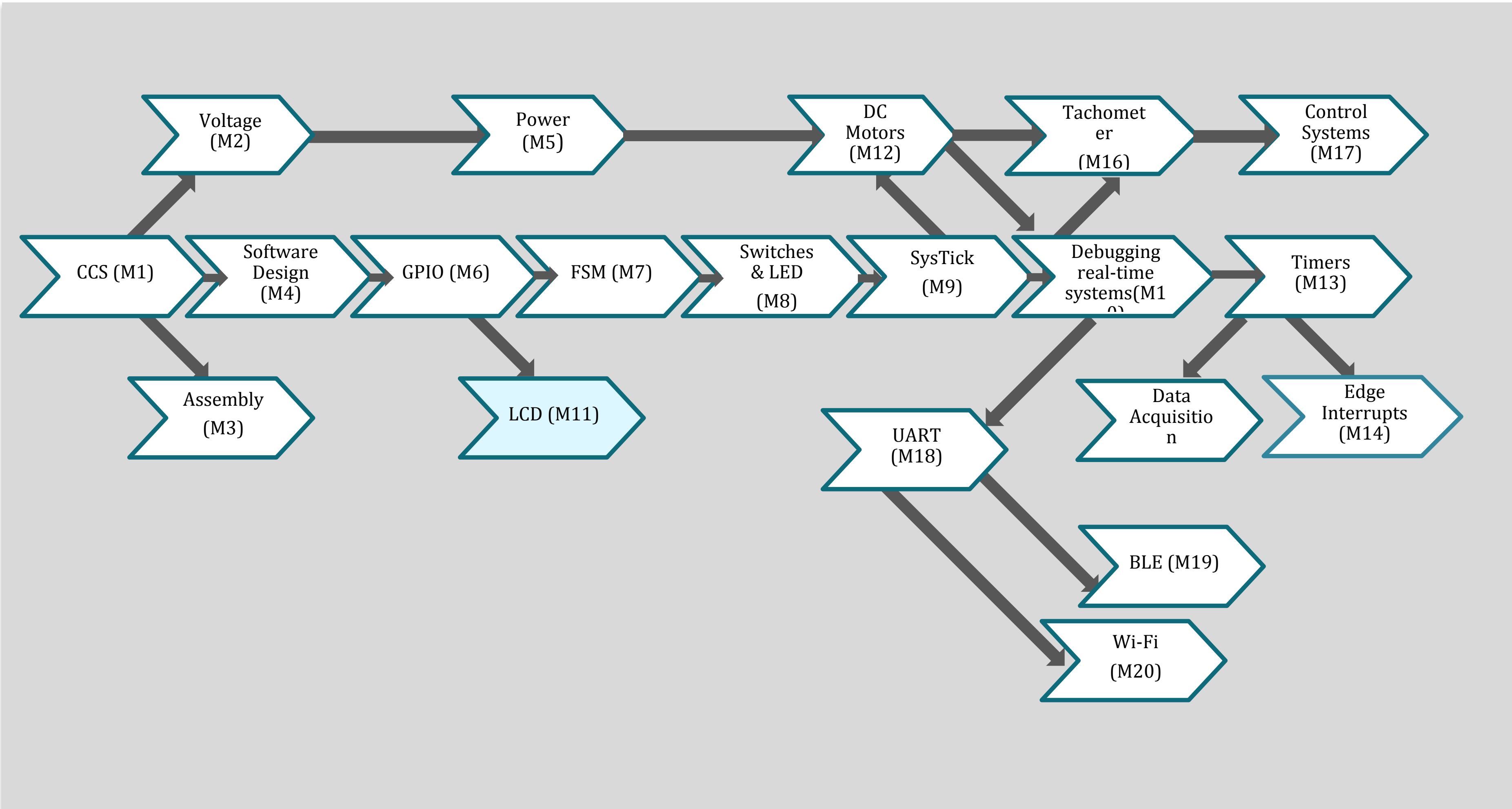

Figure 2. TI-RSLK: Learning Modules & Curriculum Pathways

Preface

Modules:

<CCS> 1. Running Code on the LaunchPad using CCS
Prerequisites: none
Equipment: LaunchPad
Theory: how to install and configure CCS for this class
Lab: installing CCS, MSP432 drivers, and running the TExaSdisplay logic analyzer

<Voltage_Current> 2. Voltage, Current and Power
Prerequisites: none
Motivation: The hardware interfaces for the robot involve voltage, current and power
Tools: Voltmeter, current meter, ohmmeter, signal generator, oscilloscope
Equipment: 220 470 22k 33k ohm resistors, 0.47uF 10uF capacitors, voltage supply, 10-mA LED and 2-mA LED
Theory: Resistor, Ohm's Law, LED voltage current relationship
Lab: Characterize LED, Measure Reactance

<Assembly> 3. ARM Cortex M
Prerequisites: <CCS>
Motivation: teaching assembly will help understand how it works, and how to debug
Tools: CCS
Equipment: LaunchPad (20-instruction subset of Cortex M)
Theory: machine code, registers, execution, bus, memory, simple I/O
Lab: assembly language programming
Installing and running assembly language

<SoftwareDesign> 4. Software Design using MSP432
Prerequisites: <CCS>
Motivation: most of the labs are C programmed on the MSP432, C programming is a prerequisite to the class, but an introduction to C code on the MSP432 is appropriate
Tools: CCS
Equipment: LaunchPad using built-in switches and LEDs
Theory: Using typical input parameters for the robot, perform, logical operations of AND, OR, EOR, shift, add, subtract, multiply, divide, variables, and functions
Lab: Simple C programming converting ADC-inputs to calibrated distance. Given three distance measurements, implement a classification algorithm to interpret the robot world

<Power> 5. Battery and Voltage Regulation
Prerequisites: <Voltage_Current>
Motivation: Robot is battery powered; robot motor controller board has power regulation
Tools: Voltmeter, current meter, ohmmeter
Equipment: Two power resistors 5W 10-ohm resistor and 22-ohm 5W resistor, rechargeable battery (4.8V/10ohm) is 0.48A, 2A-hr battery lasts 4hr Robot with power regulation/motor driver board
Theory: Resistor, Ohm's Law
Theory: total energy in battery voltage current relationship while V>regulator minimum
Lab: battery power, calculations, measurements

<GPIO> 6. GPIO – MSP432
Prerequisites: <Logic>
Motivation: robot line sensor is needed for line following
Tools: CCS Voltmeter, oscilloscope,
Equipment: LaunchPad, integrated line sensor
Theory: conversion light to voltage, direction registers, input, output, friendly (This connects to the maze robot).
Lab: input from line sensor, output to build-in LED
Input line sensor, detect position relative to a black line on a white field

<FSM> 7. Finite State Machines
Prerequisites: <Logic><GPIO>
Motivation: FSMs are an effective solution to robotic functions
Tools: CCS
Equipment: LaunchPad using built-in switches and LEDs
Theory: loops, decisions
Lab: Very simple C programming, 2-input, 2 output FSM

<Switches_LED> 8. Interfacing Input and Output
Prerequisites: <GPIO>
Motivation: robot touch sensors will be switches, LEDs provide debugging outputs for the robot
Tools: Voltmeter, current meter, ohmmeter, CCS
Equipment: switches, LEDs, resistors, LaunchPad
Theory: GPIO, LED, positive/negative logic, pullup/pulldown, input/output
Lab: Input from switches, output to LED

<SysTick>9. SysTick Timer
Prerequisites: <Switches_LED>
Motivation: introduction to time, introduce the concept of PWM that will be needed to drive the DC motor later; this is also introduces the need for interrupts, because this one task will require 100% processor utilization
Tools: CCS, logic analyzer,
Equipment: LaunchPad, LED
Theory: Introduce the need for the microcontroller to manage time. Define processor utilization. Use SysTick to create time delays. Use time delays to create a PWM signal. Use PWM to control power delivered to an actuator (LED).
Lab: A GPIO port is connected to an LED and the software controls brightness of the LED using duty cycle. The software varies the duty cycle sinusoidally (table look up) to make the LED appear to be “breathing”. Add a resistor and capacitor, and then observe the sinusoidal output on the oscilloscope.

<Debug>10. Debugging Real-time Systems
Prerequisites: <SysTick>
Motivation: system level design/debug; students need effective debugging skills. This will mimic the process students will use to design/develop/debug their robot. Eventually, students will place the robot in the maze and hit go. The robot runs for a while autonomously. After the run, students can reconnect the USB cable and upload parameters measured during the run. This module will introduce interrupts and use SysTick to perform the line-sensor measurements in the background.
Tools: CCS
Equipment: LaunchPad, line-sensor, bump sensors
Theory: RAM versus ROM, arrays, pointers, periodic interrupts
Lab: record bump and line sensor data into arrays in Flash

<LCD> 11. Liquid Crystal Display (optional)
Prerequisites: <GPIO>
Motivation: optional for robot, but makes the course accessible for other non-robotic applications; if used with the robot, the LCD can help with debugging during the stand-alone running.
Tools: Logic analyzer, CCS
Equipment: Nokia 5110, proto-board, LaunchPad
Theory: SPI interface, graphics
Lab: Display text and graphics

<Motors> 12. DC motors
Prerequisites: <Power><SysTick>
Tools: Voltmeter, current meter, oscilloscope, CCS
Equipment: LaunchPad, Motor driver board (MOSFET, resistors, diodes), DC motor
Theory: Brushed DC motor, PWM
Lab: Open loop DC motor output, measure speed versus duty cycle, extends the simple PWM built in <SysTick>

<Timers> 13. PWM and Periodic interrupts using Timers
Prerequisites: <Motors><Debug>
Motivation: periodic interrupts are a simple way to create PWM outputs to DC motors
Tools: Voltmeter, current-meter, oscilloscope, logic analyzer, CCS
Equipment: LaunchPad, DC motor on robot
Theory: timers, interrupts, frequency, PWM
Lab: software adjusts power to one DC motor

Preface

<EdgeInterrupts> 14. Real-Time Systems
Prerequisites: <Timers>
Motivation: edge-triggered interrupts is a good way to service bumper switches on robot
Tools: Oscilloscope, logic analyzer, CCS
Tools: CCS, logic analyzer,
Equipment: bumper switches on robot, LaunchPad
Theory: interrupt driven I/O, Input triggered interrupts
Lab: Input from four switches on robot to detect collision

<ADC> 15. Data Acquisition Systems
Prerequisites: <Timers>
Motivation: the robot uses IR distance sensors to detect walls or other robots
Tools: Oscilloscope, spectrum analyzer, logic analyzer, CCS
Equipment: sensor (IR distance sensor), LaunchPad
Theory: ADC conversion, sampling, periodic interrupts conversion distance to voltage, ADC signal averaging signal to noise ratio, central limit theorem, Nyquist, calibration
Lab: Input distance; detect distance and orientation to wall

<Tach> 16. Tachometer
Prerequisites: <Timers><Motors>
Motivation: The robot can have tachometers to measure wheel rotational speed
Tools: Voltmeter, oscilloscope, CCS
Equipment: Tachometer with digital inputs, DC motor, LaunchPad
Theory: period measurement interrupts
Lab: Measure motor speed

<Control> 17. Control Systems
Prerequisites: <Tach>
Comment: Assume students do not have a lot of control theory. However, they still could implement an incremental and an integral controller.
Motivation: If we have a tachometer or encoder, we can implement a digital controller.
Tools: Voltmeter, current meter, oscilloscope, CCS
Equipment: Robot with tachometer on the motors
Theory: Input capture, incremental control, integral control
Lab: Closed loop DC motor control, spin at constant speed

<UART> 18. Serial communication
Prerequisites: <Debug>
Motivation: Students could use a long USB cable to debug and control the robot in a tethered fashion while the robot is running.
Tools: Voltmeter, oscilloscope, logic analyzer, CCS
Equipment: LaunchPad connected with UART to a PC
Theory: Modulation, encoding, transmission, decoding, error detection, synchronization, FIFO queues
Lab: stream data from robot to PC, build an interpreter so student can manually control the robot from the laptop keyboard.

<BLE> 19. Bluetooth Low Energy
Prerequisites: <UART>
Motivation: Students could use a cell phone to debug and control the robot.
Tools: Logic analyzer, CCS
Equipment: LaunchPad connected with UART to CC2650BP (SNP)
Theory: characteristics, services, advertising
Lab: stream data from microcontroller to phone

<Wi-Fi> 20. Wi-Fi
Prerequisites: <UART>
Motivation: Students stream data from the robot onto a web page.
Tools: Logic analyzer, CCS
Equipment: LaunchPad connected with UART to CC3120 Booster
Theory: UDP TCP DNS, wireless router, creating a web server
Lab: stream data from microcontroller to web server