TI-RSLK MAX

Texas Instruments Robotics System Learning Kit

Texas Instruments

Module 8

Introduction: Interfacing input and output

Introduction: Interfacing input and output

Educational Objectives:

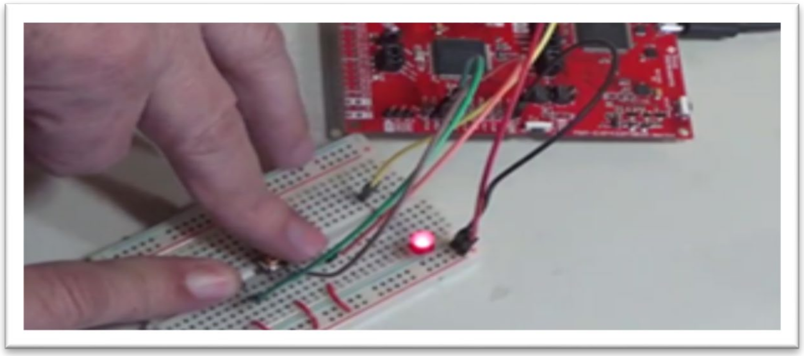
LEARN Switch & LED fundamentals,
BUILD interface circuits for switches and LEDs with TI's LaunchPad development board
WRITE CODE to configure switches as inputs and LEDs as outputs
DESIGN, TEST & DEBUG A SYSTEM
"Window Intruder Detect Security System"

Prerequisites (Modules 1, 2, 3, 4, 6)

- Basic circuits, Ohm's Law (Module 2)
- Running code on the Launchpad using CCS (Module 1,4)
- GPIO (Module 6)

Recommended reading materials for students:

- Chapter 8, **Embedded Systems: Introduction to Robotics**, Jonathan W. Valvano, ISBN: 9781074544300, copyright © 2019



The design, development and debugging of a robotic system involves many tasks. Building of any system involves knowledge of basic components and how they work and connect with other components commonly termed as **interfacing**. In fact, we can divide all of engineering into sub-systems and interfaces. Interfaces are used to combine multiple **sub-systems** together to form a more complex system. In many examples of interfacing with a microcontroller we use input devices to feed data into a computer and output devices to allow the computer to affect its surroundings.

Another name for sub-system is **module**, which can be a software module or hardware module. Another name for a software module is **device driver**. In order to build large systems, we need a method to manage complexity. Breaking a large system into modules, which are in turn broken into smaller modules, is the standard approach to dealing with complexity. There are two aspects of a module: what it does and how it works. Modular design provides an abstraction that allows us to separate what a device does from how it works. For example, consider the red LED on your LaunchPad. There are two prototypes in the **LaunchPad.h** header file that describe what this module does:

void LaunchPad_Init(void);
void LaunchPad_LED(uint8_t data);

How the module works can be found in **the LaunchPad.c** code file. Good modularity maximizes the number of modules while minimizing the **coupling** between modules. One quantitative measure of coupling is **bandwidth**, or the amount of data flowing from one module to another.

Again, to connect modules together we need an **interface**. In the software world, one module is connected to another by the public functions that can be called. This means the header files in C define the interconnection between software modules. In the hardware world, physical devices, e.g., electrical, mechanical, chemical, biological, allow modules to be interconnected. Furthermore, when connecting software modules to hardware devices, we use a combination of software and hardware components to affect the connection. For example, the IR sensor is a hardware device that uses optics to measures distance. We will use optical devices (e.g., the sensor), electrical circuits (e.g., the analog to digital converter (ADC), and software (e.g., the ADC routines) to connect the sensor to the robotic software running on the microcontroller.

Introduction: Interfacing input and output

A common example of an input device is the **switch**. Engineers use switches in a myriad of applications touching every industry such as aerospace, automotive, chemical, communication, marine, medical, military, petrochemical, and transportation. Switches are also found in the devices we use in our homes every day. We use switches in nearly all electrical and mechanical products.
We can categorize switches as:

- Momentary pushbutton
- Rotary
- Slide switch, and
- Toggle switch.

The state of the switch is either an “OPEN” or a “CLOSE”, which can be read as binary information by a microcontroller. A typical switch has a 100-MΩ resistance when “**not pressed**” and has a 0.1-Ω resistance when “**pressed**”. We can interface switches with either positive logic or negative logic. Most commonly, the switches require the use of internal pull-up resistors for negative logic switches and internal pull-down resistors for positive logic switch interfaces. The pull-up and pull-down functions are enabled by software during initialization.

A simple example of an output device is the light emitting diode Or LED. Like switches, LEDs are binary devices, in that they can be either “ON” or “OFF”. The software controls the state of LED explicitly by calling **the LaunchPad_LED** function with a 1 or a 0. This learning module will use LEDs to report binary diagnostic information. However, we can find LED interfacing in many applications, such as optical cables, solid-state relays, digital isolation barriers, and infrared transmitters. LEDs have a nonlinear voltage current relationship. Interfacing an LED requires understanding of Ohm’s Law in resistors.

Interfacing switches and LEDs to the microcontroller is an appropriate place to start because the process is simple and the proper behavior is obvious to the learner. However, wrapped into the simple activity of connecting switches and LEDs to the microcontroller, we can expose our students to the fundamental processes of design, assembly, coding, and testing. As part of the lab, students will design **a window intruder detect system** the knowledge they gained in this module. Ultimately, the robot will use similar switches to detect an object. The students will use LEDs to provide visualizing of where and what the robot software is doing and also help with debugging, as the robotic system is put together to accomplish the planned task of solving the maze.

ti.com/rslk