模块 2

实验 2：连接输入和输出

实验 2：连接输入和输出

2.0 目标

本实验的目的是学习如何使用 TI 的 Launchpad 开发板将开关和 LED 连接到微控制器。

1. 您将了解 MSP432 Launchpad。
2. 您将编写基于 GPIO 输出控制三色 LED 的程序。
3. 您将编写基于 GPIO 输入和输出控制触摸控制模块和蜂鸣器模块的程序。
4. 您将在面包板上构建电路并执行明确的测量，以验证它们是否正常运行，并提高您对它们如何工作的理解。
5. 您将使用此知识设计自己的窗口入侵探测器警报系统。

小知识：实验中连接的开关将成为机器人上的撞击探测器。机器人将使用 LED 输出作为调试工具，以便您可视化软件正在执行的操作。

2.1 入门

2.1.1 从下面的软件工程起步

浏览以下工程：

InputOutput（LaunchPad 上的开关和 LED 的输入/输出）
GPIO（简单的输出到四个引脚）
Switch（输入开关的软件驱动程序）
Three_led（对应点亮三色 LED（GPIO 输出）视频中的程序）
GPIO_input（对应 GPIO 输入视频中的程序）
Lab_Switches_LED（本实验的入门项目）

2.1.2 参考资料

B3F-1052.pdf Switch Datasheet
HLMP-4700.pdf LED Datasheet
MSP432P4xx Technical Reference Manual (SLAU356)
Meet the MSP432 LaunchPad (SLAU596)
MSP432 LaunchPad User's Guide (SLAU597)
MSP432P401R Datasheet,msp432p401m.pdf (SLAS826)

2.1.3 阅读材料

Volume 1 Section 2.7, 4.1, 4.2.2, 4.3, and 4.6
Embedded Systems: Introduction to the MSP432 Microcontroller
Volume 2 Section 2.4, and 2.6
Embedded Systems: Real-Time Interfacing to the MSP432 Microcontroller

8.1.4 本实验所需组件

数量	组件描述	制造商	型号
1	MSP-EXP432P401R LaunchPad	TI	MSP-EXP432P401R
1	**触摸控制模块**	JiangNiu	JN-RSLK-Touch
1	**蜂鸣器模块**	JiangNiu	JN-RSLK-BUZZER
1	**V1. 1 转接板**	JiangNiu	JN-RSLK-v1.1-IF
1	Red 2mA 5mm diffused LED		
1	Carbon 1/6W, 5%, 470		
3	B3F tactile push button switches		
1	solderless breadboard		

2.1.5 所需实验设备

电压表
示波器（一个至少 10 kHz 采样的通道）
逻辑分析仪 （4 个至少 10 kHz 采样的通道）

实验 2：连接输入和输出

2.2 系统设计要求

在本实验，你将实现设计和测试 3 个案例，用于熟练掌握 GPIO 的输入和输出功能。
案例 1. 点亮 MSP432 Launchpad 板载三色 LED2。
案例 2. 通过检测触摸控制模块是否被按键下，来控制蜂鸣器发出声响。
案例 3. 将设计和测试一个窗口入侵探测器报警系统。如图 2.1 的框图所示，我们的简易窗口入侵探测器报警系统有三个输入和一个输出。输入是三个开关，用正逻辑实现，见图 2.2。第一个开关输入称为 **Activate**，用作布防/撤防控制。有两个窗口传感器，称为 **Window1** 和 **Window2**。当 **Activate** 按下或为 true 时，将激活安全系统。 当 **Activate** 没有按下或为 false 时，系统将被禁用，这意味着无论窗口传感器的状态如何，警报都将关闭。当窗口传感器被按下或为 true 时，窗口处于安全位置。如果未按下任一窗口传感器，则不安全。输出是一个名为 **Alarm** 的 LED，它以正逻辑实现。您将以 5 Hz（开启 100ms，关闭 100ms）闪烁 LED 以表示不安全状况。换一种说法，当传感器 **Window1** 或 **Window2** 检测到入侵者时，LED 应快速闪烁。您将把这些开关和 LED 连接到面包板，并根据表 2.1 中显示的真值表将它们连接到 MSP432 LaunchPad 开发板。

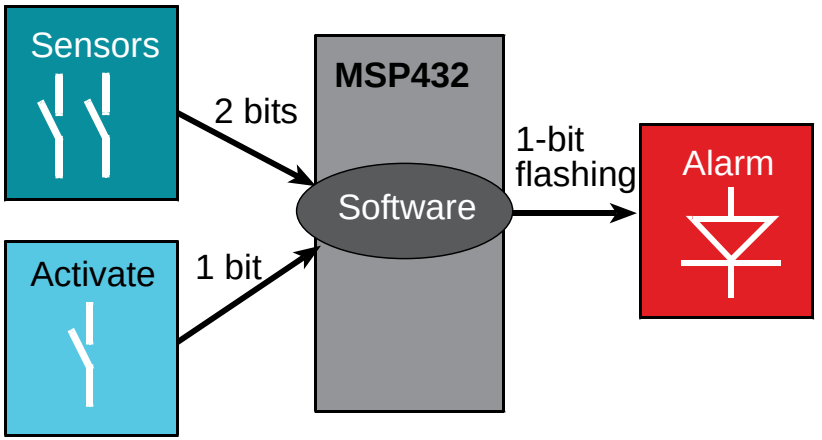

图 2.1. 窗口入侵探测器报警系统。

Activate switch	Window1 sensor	Window2 sensor	Alarm (LED)
OFF	X	X	OFF
ON	Not Pressed	Not Pressed	Flash at 5 Hz
ON	Not Pressed	Pressed	Flash at 5 Hz
ON	Pressed	Not Pressed	Flash at 5 Hz
ON	Pressed	Pressed	OFF

表 2.1. 窗口入侵探测器报警系统真值表

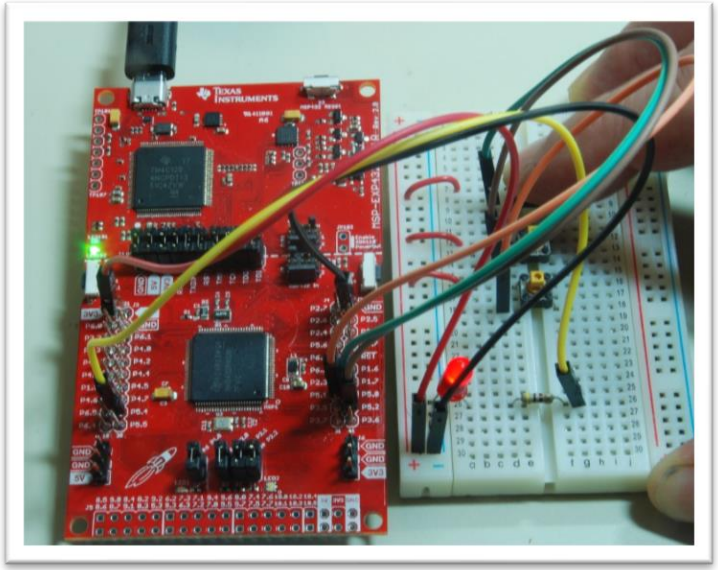

图 2.2. MSP432 LaunchPad 和外部电路

实验 2：连接输入和输出

2.3 案例 1 实验准备

MSP432 Launchpad 板载 LED2 为可控七彩灯 红绿蓝 三色全彩，可通过 R、G、B 的不同组合实现不同色彩。在 MSP432 Launchpad 的位置如图 2.3 所示，LED2 与 MSP432 内部引脚连接如图 2.4 所示。

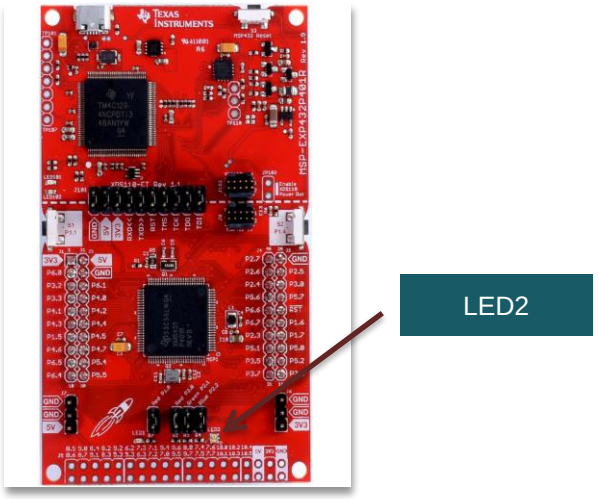

图 2.3 LED2 位置

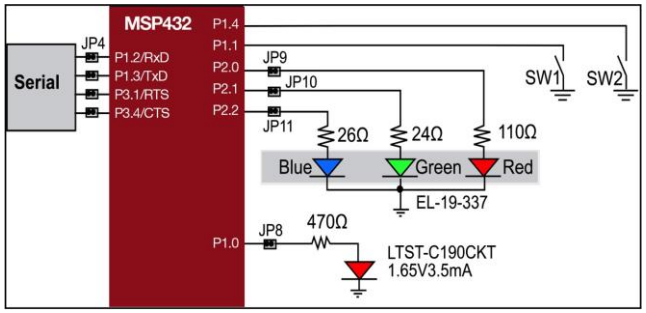

图 2.4 P2.0→红色; P2.1→绿色; P2.2→蓝色

2.4 案例 1 实验步骤

2.4.1 定义 LED 颜色对应值

红绿蓝三种颜色 LED 分别对应 MSP432 的 P2.0、P2.1、P2.2 三个引脚，引脚置高电平点亮对应的 LED。因此将颜色定义为如下程序：

```
#define RED        0x01  //R
#define GREEN      0x02  //G
#define BLUE       0x04  //B
#define yellow     0x03  //RG-
#define sky blue   0x06  //-GB
#define white      0x07  //RGB
#define pink       0x05  //R-B
```

2.4.2 将 MSP432 P2 端口配置为输出

要初始化通用 I/O 端口，将执行以下步骤。首先通过将 0 写入 PxSEL0 和 PxSEL1 寄存器来指定 GPIO。其次将设置寄存器的方向。初始化配置函数如下所示：

```
void Port2_Init(void){
  P2->SEL0 = 0x00;
  P2->SEL1 = 0x00; //configure P2.2-P2.0 as GPIO
  P2->DS = 0x07;   // make P2.2-P2.0 high drive strength
  P2->DIR = 0x07;  // make P2.2-P2.0 out
  P2->OUT = 0x00;  // all LEDs off
}
```

设置 P2 端口输出功能，函数如下所示：

```
void Port2_Output(uint8_t data){
  P2->OUT = data; // write all of P2 outputs
}
```

2.4.3 点亮 LED2

实现 LED2 每隔 1000ms 亮一种颜色。main 函数如下所示：

实验 2：连接输入和输出

```
void main(void)
{
    Clock_Init48MHz(); //Initialize clock to 48 MHz
    Port2_Init(); // initialize P2.2-P2.0 and make them
outputs (P2.2-P2.0 built-in LEDs)

    while(1){
        Port2_Output(RED);
        Clock_Delay1ms(1000);  //Delay about 1000 ms
        Port2_Output(GREEN);
        Clock_Delay1ms(1000);  //Delay about 1000 ms
        Port2_Output(BLUE);
        Clock_Delay1ms(1000);  //Delay about 1000 ms
    }
}
```

2.5 案例 2 实验准备

我们通过检测触摸控制模块被按下，来控制蜂鸣器发出声响。 将 V1.1 转接板安装到 MSP432 Launchpad 上，然后分别将触摸控制模块和蜂鸣器模块安装到 V1.1 转接板的③号位置和④号位置，连接引脚如图 2.5 所示：

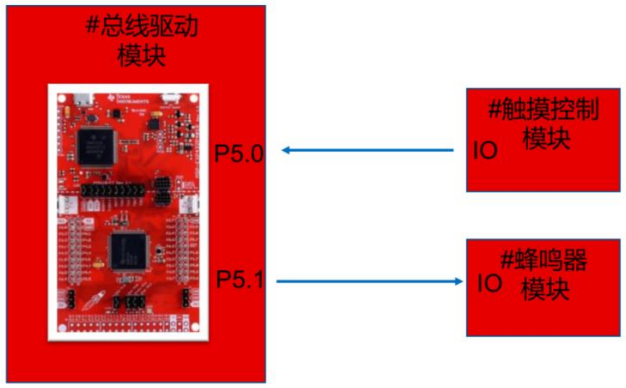

图 2.5 触摸控制模块和蜂鸣器模块使用引脚

2.6 案例 2 实验步骤

2.6.1 将 MSP432 P5 端口配置为输入和输出

触摸控制模块使用 P5.0 的输入功能，蜂鸣器模块使用 P5.1 的输出功能。我们要初始化 P5 端口为通用 I/O 功能，首先通过将 0 写入 PxSEL0 和 PxSEL1 寄存器来指定 GPIO,其次设置寄存器实现 P5.0 为输入，P5.1 为输出。
初始化函数如下所示：

```
void Port5_Init(void){
  P5->SEL0 = 0x00;
  P5->SEL1 = 0x00;//configure P5.0-P5.1 as GPIO
  P5->DIR = 0x02; //make P5.0 in, P5.1 out
  P5->REN = 0x01; //enable pull resistors on P5.0
  P5->OUT = 0x01; //P5.0 are pull-up
}
```

设置读取 P5.0 输入的函数：

```
uint8_t Port5_Input(void){
  return (P5->IN&0x01); //read P5.0 inputs
}
```

设置写入 P5.1 输出的函数：

```
void Port5_Output(uint8_t data){
  P5->OUT = (P5->OUT&0xFD)|data; //write P5.1 outputs
}
```

2.6.2 触摸控制模块控制蜂鸣器发出声响

MSP432 通过读取触摸控制模块的输入状态，判断是高电平还是低电平，如果是高电平,则控制蜂鸣器发出声响。main 函数如下所示：

实验 2：连接输入和输出

```
void main(void)
{
    uint8_t status;
    Port5_Init(); //initialize P5.0-P5.1 and make P5.0
input and P5.1 outputs
    while(1){
        status = Port5_Input(); //get P5.0 input status
        switch(status){
        case 0x01:
            Port5_Output(0x02); //control the buzzer module
to make a sound
            break;
        case 0x00:
            Port5_Output(0x00);
            break;
        default:
            Port5_Output(0x00);
            break;
        }
    }
}
```

2.7 案例 3 实验准备

2.7.1 开关接口

您最终将需要三个开关。 但是，您需要首先将一个开关连接到 TI LaunchPad 开发板。 图 2.6 显示了将开关连接到微控制器的一种可能方法。当未按下开关时，您可以使用内部或外部下拉电阻使引脚上的电压为零。

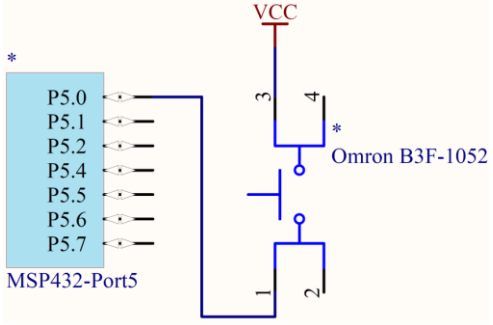

图 2.6 . 使用内部下拉将开关连接到 P5.0 的接口 (CircuitMaker)

为了标准化整个级别的时序，我们将激活外部晶振并以 48 MHz 运行。

Warning: *限制流入和流出端口引脚的电流小于 6 mA。构建开关接口的一个非常糟糕的方法是将开关的一侧置于+ 3.3V 而另一侧接地，每当按下开关时都会导致 3.3V 接地短路。*

提示：使用 P5.0 将一个开关与内部上拉接口连接的示例代码

```
uint8_t sensor;
int Program8_1(void){
  Clock_Init48MHz();  // makes bus clock 48 MHz
  P5->SEL0 &= ~0x01;  // configure P5.0 GPIO
  P5->SEL1 &= ~0x01;
  P5->DIR &= ~0x01;   // make P5.0 in
  P5->REN |= 0x01;    // enable pull resistor on P5.0
  P5->OUT &= ~0x01;   // P5.0 pull-down
  while(1){
    sensor = P5->IN&0x01; // read switch
  }
}
```

当 **Program8_1** 运行时，使用电压表测量未按下开关和按下开关时引脚上的电压。未按下开关时，您应该获得 0 V，按下开关时，您应该获得 3.3V。读取输入后，将引脚上的电压与软件中的值进行比较。使用调试器观察全局变量**传感器**。您最终将系统扩展为具有三个输入，并且软件将读取三个开关的状态。我们有意给这个例子用一个开关，知道你需要修改三个开关输入的硬件和软件。您将在项目中找到类似于 8_1 的 **Program8_2**，但会激活 **TExaS** 逻辑分析器。

实验 2：连接输入和输出

2.7.2 LED 接口

使用其数据手册查找用于本实验的 LED 的所需（Vf，If）工作点。假设微控制器引脚为 3.3V，则计算获得该工作点所需的电阻。选择接近该值的标准电阻值（例如，100，220，270，330，470，680，820 或 1000Ω）。确定将用于 LED 输出的引脚，并绘制 LED 的接口电路，类似于图 2.7。您必须重新绘制图 8.4，将 LED 移动到开关未使用的引脚。

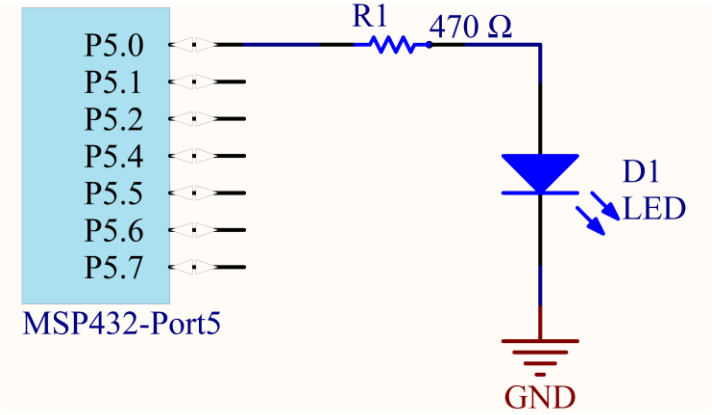

图 2.7. 将 LED 输出连接到 P5.4 的示例接口。 (CircuitMaker)

当软件输出一个高电平（假设为 3.3 V 输出）时，根据您构建的电路估算 LED 的电流和 LED 两端的电压。但是，在测量实际 LED 电流和电压时，您需要**单步**执行，因为如果运行，引脚将每秒振荡大约一百万次，并且 LED 看起来会变暗。

修改 **Program8_3**，使微控制器将相应的引脚作为输出。例如，如果你将 LED 连接到 P5.4，你必须编辑它，因此 Port5 bit 4 是一个输出，主循环振荡第 4 位。我们有意使用与上一个示例中输入相同的引脚编写 **Program8_3**，因为您知道需要修改此程序以输出到连接 LED 的特定端口引脚。程序应该只需打开和关闭 LED。

请注意，**Program8_3** 中的 LED 操作被写为**软件驱动程序**，这是一组便于 LED 操作的函数。此外，看看 LED 功能如何构成抽象，将它的作用（LED 初始化/开/关/切换）与其工作原理分开（P5 引脚 0）。

运行修改后的 **Program8_3** 并单步执行，直到 LED 亮起。测量电阻两端的电压和 LED 两端的电压。单步执行软件，直到 LED 熄灭并再次测量两个电压。在电阻器上使用欧姆定律来计算通过电阻器的电流。电阻器电流也将等于 LED 电流。将 LED 的实际（Vf，If）工作点与设计期间计算的预期值进行比较。当软件停止且微控制器的输出为高电平时，测量微控制器引脚上的电压。将此测量电压与 3.3V 的预期值进行比较。

提示：使用此程序测试 LED 接口

```
void LED_Init(void){
 P5->SEL0 &= ~0x01;  // configure P5.0 GPIO
 P5->SEL1 &= ~0x01;
 P5->DIR |= 0x01;    // make P5.0 output
}

void LED_On(void){
 P5->OUT |= 0x01;  // turn on
}

void LED_Off(void){
 P5->OUT &= ~0x01; // turn off
}

void LED_Toggle(void){
 P5->OUT ^= 0x01;  // change
}

int Program8_3(void){
 Clock_Init48MHz();  // makes bus clock 48 MHz
 LED_Init();         // activate output for LED
 while(1){
    LED_On();
    LED_Off();
 }
```

2.7.3 LED 切换

接下来，您将开发并测试闪烁 LED 5 次/秒的软件。基本上， 您需要去写 **Wait1ms()** 函数。在下一个实验中， 我们将使用 SysTick 计时器进行延时。但是在这个实验中，你可以使用两个嵌套的 for 循环。软件循环对于时间延迟非常不准

实验 2：连接输入和输出

确。因此，在这个实验里，延迟可以是任何值，使得 LED 每秒闪烁 4 到 6 次。使用秒表，逻辑分析仪或示波器验证 LED 是否以所需速率闪烁。
图 2.8 显示了在 Logic Analyzer 处于活动状态时运行的程序 8.4。 要激活逻辑分析仪以显示端口 5，请执行

TExaS_Init(LOGICANALYZER_P5);

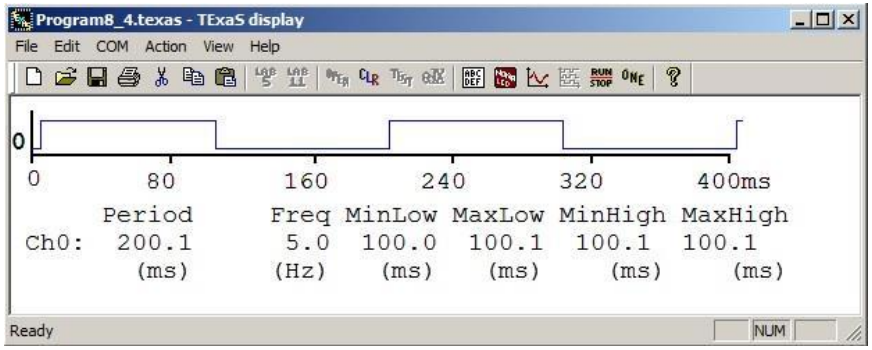

图2.8 . TExaS 逻辑分析仪运行程序 8.4，显示 P5.0 以 5 Hz 切换。

提示：使用此程序测试 LED 闪烁

```
int Program8_4(void)
  Clock_Init48MHz();   // makes bus clock 48 MHz
    TExaS_Init(LOGICANALYZER_P5);
    LED_Init();         // activate output for LED
    while(1){
    LED_Toggle();
    Clock_Delay1ms(100);  // approximately 100 ms
 }
}
```

2.8 窗口探测器报警系统

2.8.1 硬件实现

我们将使用 MSP432 上的四个引脚来实现报警系统。建议您不要使用已连接硬件的引脚。选择您想要使用的四个引脚，并绘制硬件电路图，如图 2.9 所示。

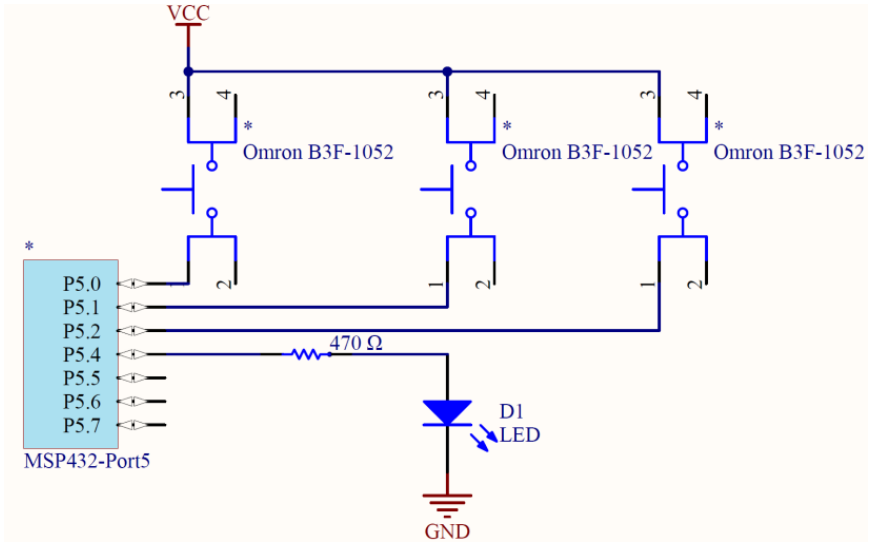

图2.9 硬件原理图接口输入和输出引脚。

实验 2：连接输入和输出

2.8.2 软件实现

这个伪代码描述了这个系统的基本方法，并将其绘制成图 2.10 中的流程图。

1. 将 LED 引脚作为输出并使三个开关引脚作为输入。
2. 系统启动时 LED 熄灭。
3. 等待 100 ms。
4. 着眼于这三个开关；如果 **Activate** 被按下，一个或两个 **Window1** 和 **Window2** 没有被按下，则切换 LED 一次，否则关闭 LED。
5. 一遍又一遍地重复步骤 3 - 5。

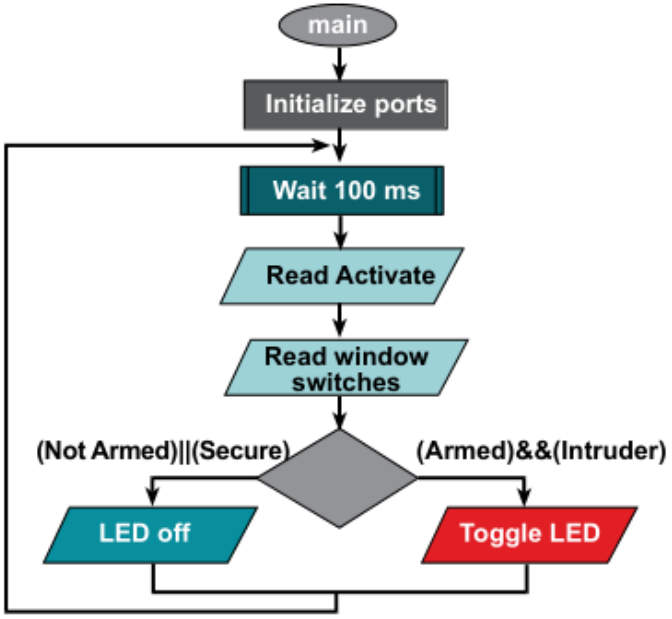

图 2.10. 系统可能的软件算法绘制为流程图

本实验的结构让您分别构建和测试每个子系统。在这个部分，我们结合开关和 LED 接口来实现窗口入侵探测器报警系统。要测试整个系统，首先应该单步执行软件，以验证它是否按照表 2.1 中列出的八种情况的预期运行。使用调试器同时观察三个输入和一个输出。首先，使用 step over debugger 命令验证正确的功能行为。
然后，你可以启动程序并测试系统全速运行。您可以使用示波器或逻辑分析仪来验证 LED 以 5 Hz 的频率闪烁，当按下 Activate 并且未按下一个或两个 Window1 和 Window2 时。

2.9 疑难解答

无法编程 LaunchPad：

- 检查 LaunchPad 开发板上的电缆，跳线。
- 检查 Windows 驱动程序以查看操作系统是否识别该板。
- 在此计算机上尝试另一个 LaunchPad。在另一台计算机上试用此 LaunchPad。

LED 没有打开：

- 检查 LED 的极性。
- 重复测量，参见第 8.3.2 节。与 LED 串联的电阻应介于 220 和 2000 欧姆之间。
- 如果 LED 上有 2 到 3V 且 LED 很暗，则它会断开（开路）或反向。

Switches 不工作：

- 许多开关有 4 个引脚，您可能会混淆开关所连接的引脚。
- 使用欧姆表测量开关上的电阻，以确定按下和非按下的条件。
- 调试器允许您可视化端口寄存器；因此，最好使用调试器检查初始化是否正确配置了方向和下拉模式。

实验 2：连接输入和输出

2.10 请思考

在这节中，我们列出了完成本实验后要考虑的思考问题。这些问题旨在测试您对本实验中概念的理解。

- 怎么让 LED 更亮？
- 如果你反向插入 LED 会发生什么？
- 如果颠倒 LED 和电阻会发生什么？即，将微控制器输出连接到 LED 的+侧，并将 LED 的 – 侧连接到电阻器的一端，将电阻器的另一端连接到地。它还能用吗？ 为什么？
- 每次你按下开关，开关就会打开/关闭/打开大约 1 - 2 毫秒。同样的，当开关被释放时，开关会关闭/打开/关闭。这个实验实际上对开关进行了去抖。主程序中的哪些操作会导致软件忽略触摸和释放时发生的开关弹跳？去抖意味着软件仅响应开关触摸事件一次，而不是在开关反弹时多次响应。

2.11 其它挑战

在这节中，我们列出了您可以做的其他活动，以进一步探索此模块的概念。您可以扩展安全系统或提出完全不同的东西。例如：

- 添加第二个 LED 以指示系统是否已激活
- 添加绿色 LED 指示一切正常
- 将这个实验实现为一个有限状态机
- 添加更多开关并实现数字门锁 （用户按特定顺序点击键，并通过 LED 模拟锁）
- 实现一个起搏器需求，用户按下开关来模拟心房传感器，并且通过 LED 模拟心室起搏
- 修改下面挑战函数，以便系统移除开关弹跳并正确计算按下开关的次数

```
int Challenge(void){ uint32_t Count=0;
  Clock_Init48MHz();  // makes bus clock 48 MHz
  Switch_Init();      // activate input from switch
  while(1){
    while(Switch_Input()==0){}; // wait for touch
    Count++;  // number of times switch is touched
    while(Switch_Input()!=0){}; // wait for release
  }
}
```

2.12 接来是哪些模块？

在这节中，我们列出了基于模块中所学概念的未来模块。

模块 3) 使用周期性中断在后台运行任务模块
模块 10) 使用 SysTick 实现延时，调暗 LED
模块 11) 使用开关为机器人添加碰撞传感器

2.13 您应该已经学会

在这节中，我们回顾了在这个模块中应该学习的重要概念是：

- 使用欧姆表、电压表和逻辑分析仪
- 使用 CircuitMaker 之类的程序绘制电路
- 使用面包板构建电路
- 编程方向寄存器
- 使用数字端口执行输入/输出，将函数编写为软件驱动程序
- 通过下拉连接一个正逻辑开关
- 连接一个正逻辑低电流 LED
- 使用 for 循环创建软件延迟
- 使用软件延迟切换 LED