Design of Smart Bulb Controlling System

1. INTRODUCTION

1.1 OBJECTIVES AND SCOPE OF THE WORK

In this project work I am implementing an smart bulb controlling system based on LDR(light dependent resister)/ photo sensor, PIR sensor and push button. Automatic smart bulb control system needs no manual operation for switching ON and OFF when there is need of light it was ontrolled by photo sensor. It detects itself whether there is need for light or not and whether there is a person exists in room or not. When darkness rises to a certain value and the room is occupaid then automatically light is switched ON and when there is other source of light i.e. day time, the light gets OFF. Along with LDR, PIR sensors the system having one push button for manual operation, i.e. to control the button to turn on/off the light.

This project is used to measure the various parameters like Light, room occupancy and push button status and display them on a LCD and control the smart bulb.

The various functions of System :

I. Reading the digital input from ADC which is derived from Light sensor and PIR sensor.

II. Sending this data to LCD.

III. Controlling the light turning On/Off the respective flags based on sensor readings and button.

This proposed research work would be implemented using LPCXpresso LPC1769 hardware board based embedded system design methodology, and freeRTOS based LPCXpresso IDE Tool for firmware development.

Hardware requirements:

- LPC1769 Microcontroller based LPCXpresso Board
- LCD, Led
- LDR, PIR Sensors
- USB Cable
- Bread Board
- Connecting wires

Software requirements:

- freeRTOS
- LPCXpresso IDE
- Embedded C

2. HARDWARE IMPLEMENTATION OF DASH BOARD

The implementation of the Dash board can be divided in two parts.

1. Hardware implementation 2. Firmware implementation

The **Hardware implementation** deals in drawing the schematic on the plane paper according to the application, testing the schematic design over the breadboard using the various components.

The block diagram discusses about the required components of Dash board and working condition is explained using circuit diagram.

Block Diagram of System :

The block diagram of board is as shown in Fig 2.1. It consists of microcontroller, LCD module, LDR and PIR sensors, power supply unit, Led and push button (switch).

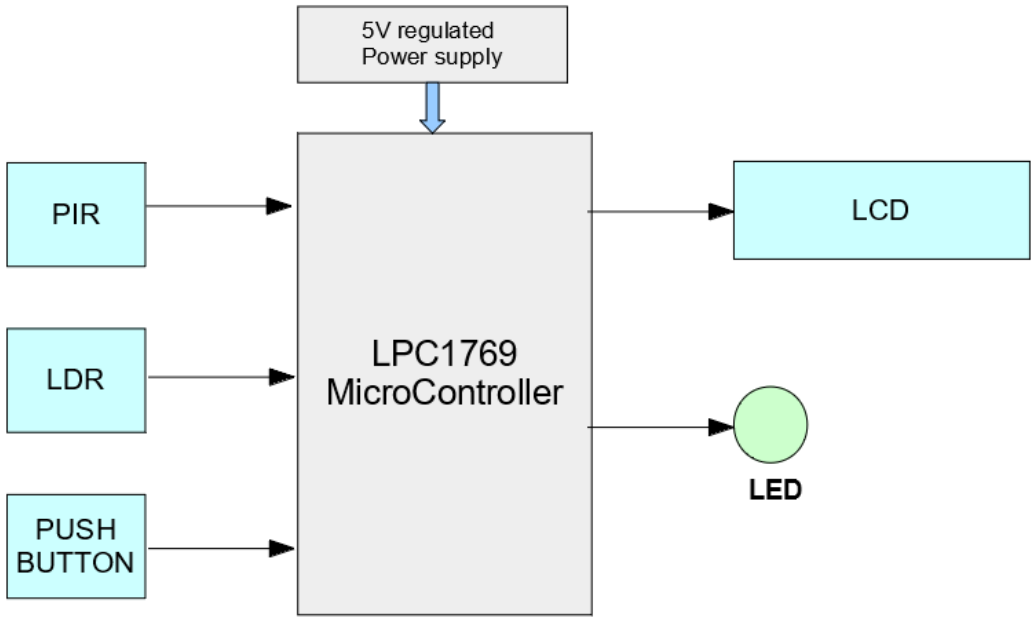


Fig. 2.1 : Hardware diagram of the system

Connections of Modules:

PORT-0 :

P0.9	P0.20	P0.23	P0.24				
EXT_INT	SMRT_BULB	PIR	LDR				

PORT-1:

P1.18	P1.19	P1.20	P1.21	P1.22	P1.23	P1.24	P1.25	P1.26	P1.27	P1.28
LCD_RS	LCD_RW	LCD_EN	LCD_D0	LCD_D1	LCD_D2	LCD_D3	LCD_D4	LCD_D5	LCD_D6	LCD_D7

Fig. 2.2 : Port Map for SmartBulb controlled Unit

Circuit Diagram of Dash board :

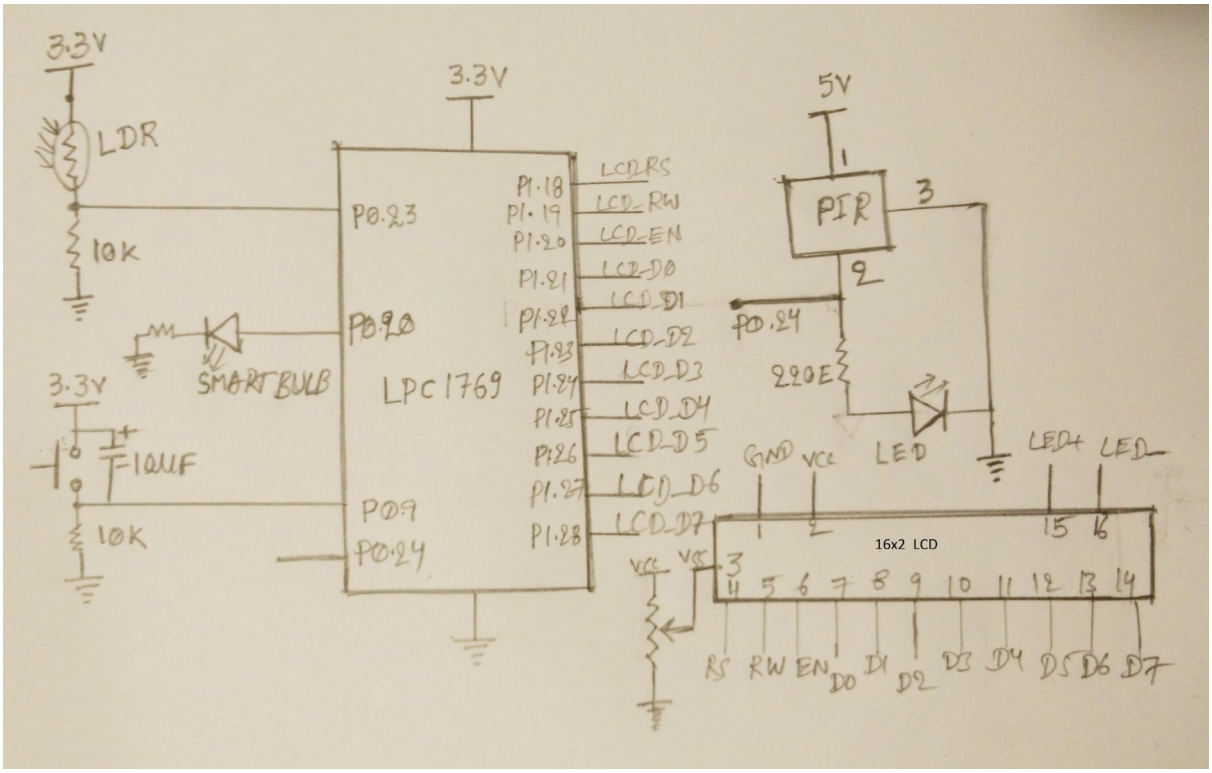
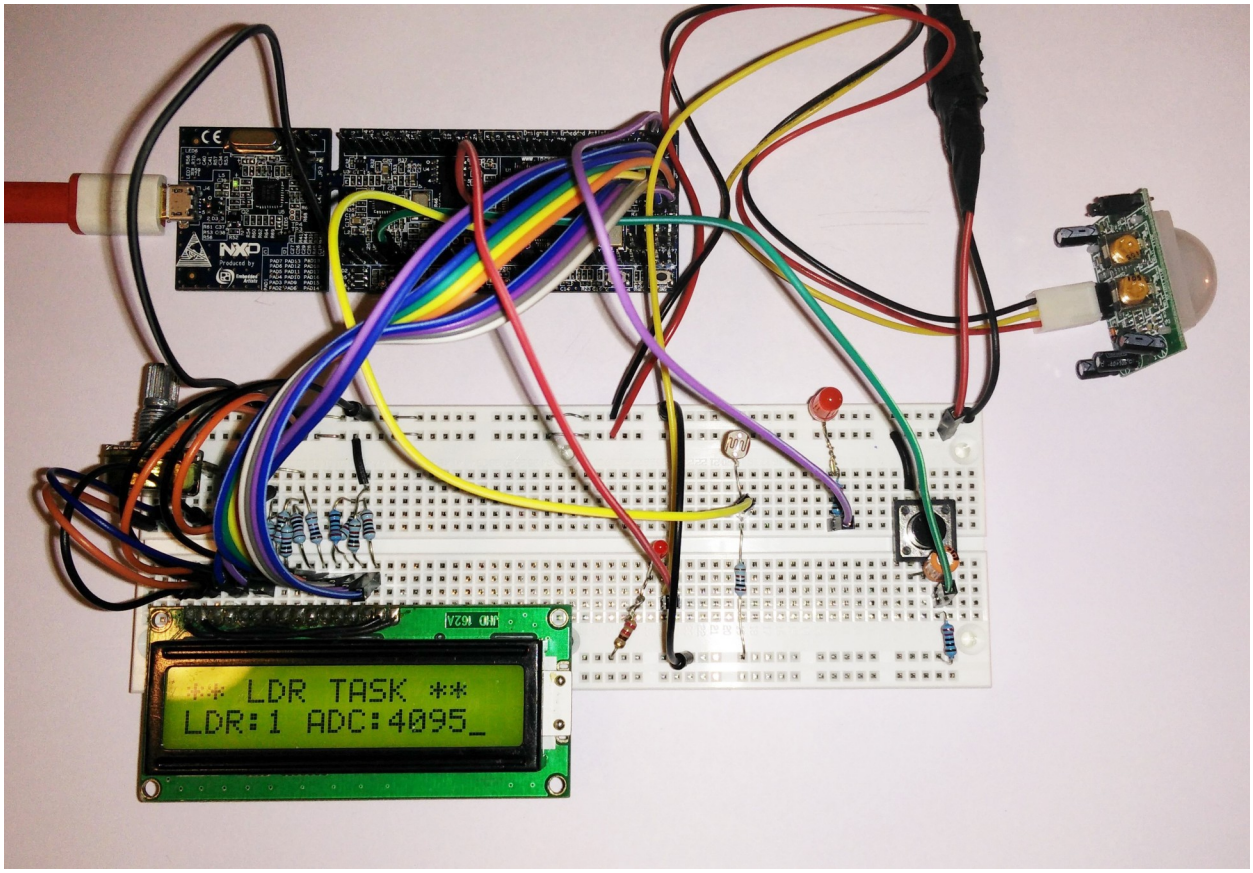


Fig 2.3: Circuit (Schematic) diagram of Dash board

Hardware setup :



3. SOFTWARE IMPLEMENTATION OF DASH BOARD

System initialisation :

Hardware Initialisation :

- Set Port0 pins 23, 24 as Analog input pins which is connected to LDR and PIR sensors.
- Set Port0 pin 9 as gpio interrupt pin which is connected to push button.
- Set Port0 pin 20 as gpio output pin which is smart bulb led through 220E resister.
- Set Port1 pins 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28 are gpio output pins which connected to LCD.

Software Initialisation :

- The ADC_Calibration() function used to calibrate light and pir sensors at system initialisation, it calculates the LDRsensorCutOff and PIRsensorCutOff (where CutOff = sensorLow + sensorHigh /2).
- The LCD_init() function used to initialise the LCD module.
- The vSetupTimerForRunTimeStats() function configures a timer to be used as the run time statistics time base.

The working of System :

This project uses 6 tasks as defined below:

1. **vTask_AnalogRead_PIR()** with Priority 2 : This task runs at periodic intervals of 34ms.

This task reads the ADC value if it is greater than the threshold (PIRsensorCutOff) sets the *b_Room_Occupancy* flag and gives the xSmartBulb_CountingSemaphore.

2. **vTask_AnalogRead_LDR()** with Priority 2 : This task runs at periodic intervals of 34 ms.

This task reads the ADC value if it is greater than the threshold (LDRsensorCutOff) sets the *b_LDR_State* flag and gives the xSmartBulb_CountingSemaphore.

3. **vTask_Button_Handler()** with Priority 3 : This is a bottom-half for the External Interrupt ISR.

This task runs when it acquires xButtonSemaphore from the External interrupt ISR. It toggles the *b_Button_State* flag and gives the xSmartBulb_CountingSemaphore.

4. **vTask_SmartBulb_Ctrl()** with Priority 3 :

This task waits for xSmartBulb_CountingSemaphore and if if get the semaphore sets the smart bulb on/ off based on b_Button_State, b_LDR_State and b_Room_Occupancy flags.

5. **vTask_16x2LCD_Print()** with Priority 0 :

This task uses the Gatekeeper Design methodology. This task waits for the messages to arrive, if message is received reads the message from queue and display the message on LCD.

6. **vTask_RunTimeStats()** with Priority 1 :

This task gathers run time stats for tasks in the system and prints on serial console.

FreeRTOS Features :

Mutex : This project uses 2 mutex smaphores.

1. xADCSemaphoreMutex for ADC convertion LDR and PIR tasks uses this mutex.
2. xLCDSemaphoreMutex for display data LCD task uses this mutex.

Message Queue : This project uses 1 message queue, LCDQueue Depth: 5. LDR and PIR tasks send messages to Queue regarding the current status, then the vTask_16x2LCD_Print() reads from the queue and display on LCD.

Binary Semaphore : This project uses 1 binary semaphore xButtonSemaphore. The external interrupt (push button) ISR gives the xButtonSemaphore to the bottom half when it ensures that the correct rising-edge interrupt has occurred on the interrupt pin. The bottom half, which is waiting indefinitely, for this semaphore then acquires the semaphore and toggles the ButtonState flag.

Counting Semaphore : This project uses 1 counting semaphore xSmartBulb_CountingSemaphore. The bottom half of external interrupt, LDR and PIR tasks gives the xSmartBulb_CountingSemaphore to the vTask_SmartBulb_Ctrl when they ensures that there is need to change the SmartBulb state. Then the vTask_SmartBulb_Ctrl task, which is waiting indefinitely, for this semaphore then acquires the semaphore and chages the SmartBulb state.

RunTime Statistics:

Task	Abs Time	%
RUNSTAT	94006	1%
IDLE	570857	6%
LCD DIP	1758214	21%
LDR SEN	2303686	34%
PIR SEN	2309390	34%
SMR BLB	68931	<1%
INT ISR	5399	<1%