Standard Embedded Diploma - Mini Project 3

Project Title:

Smart Home Automation Using ATmega32, LCD, Temperature Sensor, LDR Sensor, Flame Sensor and Motor Control

Objective:

To design and implement a smart home system that automates the control of lighting and fan speed based on environmental conditions. The project uses an ATmega32 microcontroller, LM35 temperature sensor, LDR (Light Dependent Resistor), Flame sensor, and a PWM-controlled motor. An LCD provides real-time display, and an H-bridge circuit controls the fan motor. Additionally, LEDs indicate varying light intensity levels.

Project Overview:

This project involves developing a smart home system that adjusts lighting, fan speed, and safety alerts based on environmental conditions:

1. **Lighting Control**: The LDR detects ambient light intensity and controls three LEDs to indicate different levels of light.
2. **Fan Speed Control**: The LM35 temperature sensor adjusts the fan speed proportionally to the room temperature.
3. **Fire Detection**: A flame sensor detects fire, triggers an alarm, and displays a critical alert on the LCD.
4. **LCD Display**: Shows real-time temperature readings, fan status, light intensity, and alerts.

Features:

- **Automatic Lighting Control:** Based on the LDR readings, the system controls three LEDs according to light intensity thresholds:
    - **Intensity < 15%:** All 3 LEDs (Red, Green, and Blue) turn ON.
    - **Intensity 16–50%:** Red and Green LEDs turn ON.
    - **Intensity 51–70%:** Only the Red LED turns ON.
    - **Intensity > 70%:** All LEDs are turned OFF.

- **Automatic Fan Speed Control**: Fan speed is automatically adjusted based on room temperature, controlled by a PWM signal. The fan operates at different speeds based on temperature ranges:
    - **Temperature ≥ 40°C:** Fan ON at 100% speed.
    - **Temperature ≥ 35°C and < 40°C:** Fan ON at 75% speed.
    - **Temperature ≥ 30°C and < 35°C:** Fan ON at 50% speed.
    - **Temperature ≥ 25°C and < 30°C:** Fan ON at 25% speed.
    - **Temperature < 25°C:** Fan OFF.
- **Fire Detection and Alert:** The system raises an alarm and displays "Critical alert!" on the LCD if fire is detected. The system remains in alert mode until the flame is no longer detected.
- **LCD Display: Real-time feedback on system status:**
    - **First Row**: Displays "FAN is ON" or "FAN is OFF" based on the fan's state.
    - **Second Row**: Displays "Temp= xxC LDR= xxx%" showing temperature and light intensity.

Hardware Components:

- **Microcontroller:** ATmega32
- **Light Sensor:** LDR (Connected to PA0, ADC Channel 0)
- **Temperature Sensor:** LM35 (Connected to PA1, ADC Channel 1)
- **Display:** 16x2 LCD in 8-bit mode
    - **RS:** PD0
    - **Enable:** PD1
    - **R/W:** Ground (always write mode)
    - **Data Pins:** PORTC (8-bit mode)
- **Motor Control:** H-Bridge connected for motor speed control.
    - **IN1:** PB0
    - **IN2:** PB1
    - **Enable1:** PB3/OC0

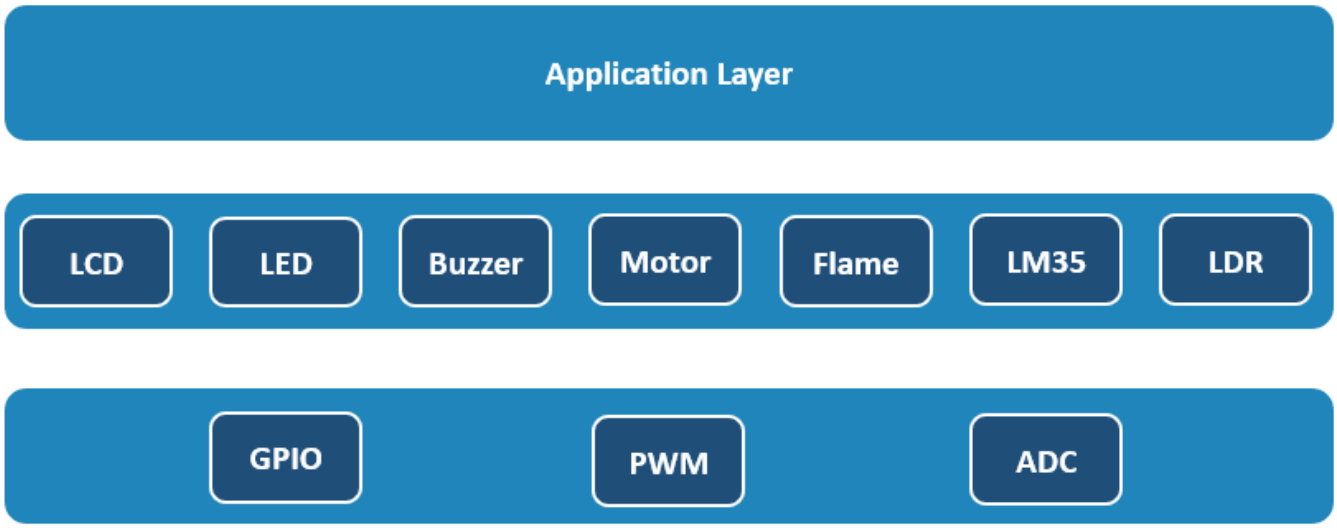
- **LEDs:**
    - **Red LED:** Connected to PB5
    - **Green LED:** Connected to PB6
    - **Blue LED:** Connected to PB7
- **Flame Sensor:** Connected to PD2 to detect fire**.**
- **Buzzer:** Connected to PD3, activated during fire alert**.**

Operation Description:

- **Lighting Control (LDR):** The LDR monitors light intensity, and the three LEDs are controlled based on the previous mentioned thresholds.
- **Temperature Control (LM35):** The LM35 sensor measures room temperature. The fan speed is controlled using a PWM signal adjusting speed according to temperature ranges specified above.
- **Fire Detection:** The flame sensor triggers an alert when fire is detected:
    - Displays "Critical alert!" on the LCD.
    - Activates the buzzer.
    - The system stays in alert mode until the flame sensor no longer detects fire.
- **LCD Display:**
    - **First Row:** Indicates the fan status ("FAN is ON" or "FAN is OFF").
    - **Second Row:** Displays temperature ("Temp= xxx") and light intensity ("LDR= xx%").

System Requirements:

1. **Microcontroller:** ATmega32
2. **System Frequency:** 16 MHz
3. **The Project should be implemented using the below layered model architecture**



Drivers Requirements

This section will list the specific requirements for the ADC, GPIO, LCD, temperature sensor, LDR, LED, DC motor, flame sensor, buzzer, and PWM drivers used in the project. Each driver must be implemented or reused from previous course materials as specified.

ADC Driver Requirements

The ADC (Analog-to-Digital Converter) will be used to read analog data from sensors such as the LM35 temperature sensor and the LDR. The ADC must be configured as follows:

1. **Reference Voltage**: Internal reference voltage 2.56V.
2. **Prescaler**: F_CPU/128, providing the necessary accuracy for the measurements.

ADC Functions:

- **void ADC_init(void)**
  - Initializes the ADC by setting the reference voltage and prescaler.
- **uint16 ADC_readChannel(uint8 channel_num)**
  - Reads analog data from a specific ADC channel and converts it to digital.

GPIO Driver Requirements

Use the same GPIO driver implemented during the course to handle basic pin configurations for inputs and outputs. This will control the direction of pins connected to sensors, actuators, and indicators like LEDs.

PWM Driver Requirements

Use the PWM driver to control the fan's speed based on the temperature. The driver uses **Timer0** in **PWM mode** and operates as follows:

PWM Function:

- **void PWM_Timer0_Start(uint8 duty_cycle)**
  - Initializes Timer0 in PWM mode and sets the required duty cycle.
  - Prescaler: F_CPU/1024
  - Non-inverting mode
  - The function configures OC0 as the output pin.
  - **Parameters**:
    - **duty_cycle:** Percentage (0 to 100%) representing the PWM duty cycle.

LCD Driver Requirements

Use the same LCD driver implemented in the course with 8-bit data mode. The LCD is used to display temperature and light intensity values, along with fan status.

- **Display**: 16x2 LCD in 8-bit mode
- **Pin Configuration**:
    - RS: PD0
    - Enable: PD1
    - R/W: Ground (always in write mode)
    - Data Pins: PORTC (8-bit mode)

Temperature Sensor Driver Requirements

Use the course-provided LM35 temperature sensor driver to measure the room's temperature. The sensor should be connected to **ADC channel 1**.

LDR Sensor Driver Requirements

This driver is responsible for reading the light intensity using an LDR (Light Dependent Resistor). Implement the following function:

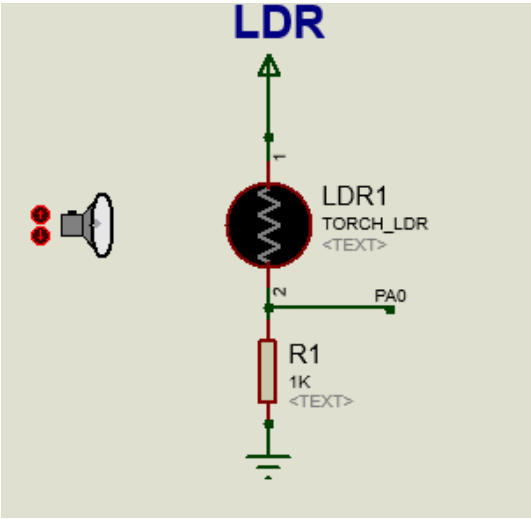
- **uint16 LDR_getLightIntensity(void)**
    - Reads the LDR sensor value and returns the light intensity.

The LDR should be connected to **ADC channel 0**.

Additional configurations:

- **LDR_SENSOR_CHANNEL_ID**: 0
- **LDR_SENSOR_MAX_VOLT_VALUE**: 2.56 Voltage
    - This represents the maximum voltage output of the LDR sensor when exposed to the lowest possible light intensity (darkness). In a voltage divider circuit, the LDR's resistance decreases with increasing light, resulting in maximum voltage in darkness. This value may vary based on the circuit configuration.
- **LDR_SENSOR_MAX_LIGHT_INTENSITY**: 100
    - This represents the maximum light intensity the sensor can measure (0% → 100%).

Circuit Schematic



LED Driver Requirements

The LED driver will control the red, green, and blue LEDs based on the light intensity. The required functions include initialization, turning LEDs on, and turning LEDs off.

LED Driver Functions:

- **void LEDS_init(void)**
  - Initializes all Leds (red, green, blue) pins direction.
  - Turn off all the Leds
- **void LED_on(LED_ID id)**
  - Turns on the specified LED.
- **void LED_off(LED_ID id)**
  - Turns off the specified LED.

Additional configurations:

- The Led pins should be configurable to be connected to any microcontroller ports and pins.
- The driver should support both Led connections Positive Logic and Negative Logic.

DC Motor Driver Requirements

The DC motor will control the fan, adjusting its speed based on temperature readings.

DC Motor Driver Functions:

- **void DcMotor_Init(void)**
  - Initializes the DC motor by setting the direction for the motor pins and stopping the motor at the beginning.
- **void DcMotor_Rotate(DcMotor_State state, uint8 speed)**
  - Controls the motor's state (Clockwise/Anti-Clockwise/Stop) and adjusts the speed based on the input duty cycle.
  - **Parameters**:
    - state: Enum or uint8 that indicates motor direction (CW, A-CW, Stop).
    - speed: Motor speed in percentage (0 to 100%).

Additional configurations:

- The motor pins should be configurable to be connected to any microcontroller ports and pins.

Flame Sensor Driver Requirements

The flame sensor will detect fire incidents. Implement the following functions using a polling technique:

- **void FlameSensor_init(void)**
  - Initializes the flame sensor pin direction.
- **uint8 FlameSensor_getValue(void)**
  - Reads the value from the flame sensor and returns it.

Additional configurations:

- The flame sensor pin should be configurable to be connected to any microcontroller ports and pins.

Buzzer Driver Requirements

The buzzer will serve as an alert system in case of fire detection by the flame sensor.

Buzzer Driver Functions:

- **void Buzzer_init(void)**
  - Initializes the buzzer pin direction and turn off the buzzer.
- **void Buzzer_on(void)**
  - Activates the buzzer.
- **void Buzzer_off(void)**
  - Deactivates the buzzer.

Additional configurations:

- The buzzer pin should be configurable to be connected to any microcontroller ports and pins.

Project Files and Videos

Flame Sensor Proteus Library:
https://www.mediafire.com/file/cnrdckfrdxun270/Flame-Sensor-Library-For-Proteus-main.zip/file

Flame Sensor Video:

https://youtu.be/cD3n-Bz7liM

Project Video:

https://youtu.be/Iyj7sOfwYLw

Thank You

Edges For Training Team