**EMBEDDED SYSTEM PROJECT REPORT**

***In partial fulfilment of the requirements for the award of the degree of***

**Bachelor of Technology**

**in**

**Electronics and Communication Engineering**

V semester

*Submitted by*

**Student Name: Ankush Verma**

**Register Number: 200907058**

A logo with a brown logo

Description automatically generated

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**

MANIPAL INSTITUTE OF TECHNOLOGY

(A Constituent institute of MAHE, Manipal)

MANIPAL – 576104, KARNATAKA, INDIA

**PROJECT: DEVELOPING AUTOMATED TEMPERATURE-CONTROLLED FAN WITH LM35 TEMPERATURE SENSOR USING LPC1768 MICROCONTROLER**

**1. OBJECTIVES:**

The objective of this project is to design and implement an automatic temperature-controlled fan system using an LPC1768 microcontroller and an LM35 temperature sensor. The system should be able to automatically turn on a fan when the temperature exceeds a certain threshold to maintain a comfortable environment.

**2. ASSUMPTIONS**:

1. The LM35 temperature sensor provides an accurate measurement of the ambient

2. The fan can be controlled by a digital signal from the microcontroller to turn it on and off.

3. Power is always supplied to all components.

4. The temperature sensor is connected to pin PO.23 (ADCO.4) of the microcontroller.

5. The reference voltage for the ADC is 3.3V.

6. The temperature sensor provides a linear voltage output that is proportional to the temperature being measured.

7. The temperature sensor has a sensitivity of 10mV/0 C, as indicated by the line (V\_REF \* adc\_value / 4095 - V\_REF / 2) / 0.01 in the ISR.

8. The fan is connected to pin Pl.21 of the microcontroller.

9. The fan is turned on and off by setting and clearing the corresponding pin. 10. The temperature threshold for turning on the fan is 30 0 C.

11. The code uses interrupt-driven ADC conversion with software-controlled mode.

12. The code assumes that the ADC conversion will be completed before the next iteration of the loop.

13. The code assumes that the temperature sensor is working correctly and providing accurate readings.

14. Once the temperature rises it will turn on the fan and then it will cool down the temperature

**3. CONFIGURATIONS:**

The system consists of the following components:

1. LPC1768 Microcontroller: Reads temperature data from the LM35 sensor and controls the fan using GPIO pins.

2. LM35 Temperature Sensor: Outputs a voltage proportional to the measured temperature. Connected to an ADC pin of the microcontroller.

3. Fan: Controlled by a digital signal from a GPIO pin of the microcontroller to turn on and off. A relay or motor driver may be used to drive the fan from the microcontroller.

4. Power supply: A stable 3.3V DC power supply for the microcontroller and LM35 temperature sensor

**4. CONNECTIONS:**

1. LM35 temperature sensor:

VCC: 3.3V power supply.

VOUT: ADC 0.4 (depending on the connection/kit)

2. DC fan:

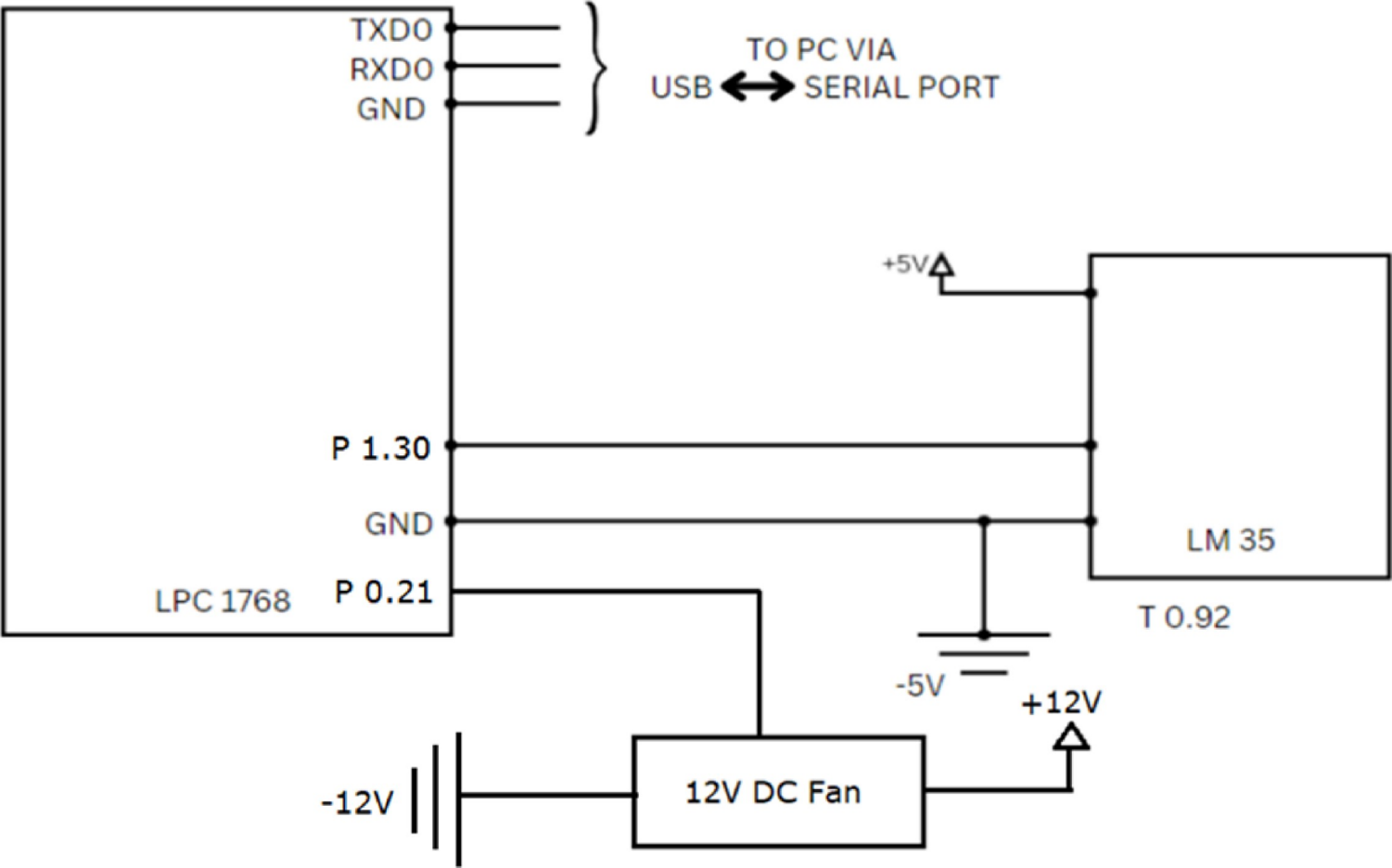
Positive terminal: FAN\_PIN (GPIO pin for controlling the fan) Negative terminal: Ground.

3. LPC1768 microcontroller:

Pl.30: ADC 0.4 (for temperature sensor)

PO.21: FAN\_PIN (for controlling the fan)

**4. BLOCK DIAGRAM:**



**5. PROGRAM:**

#include <LPC17xx.h>

#include<stdint.h>

#include<stdlib.h>

// Define constants for temperature calculation

#define T\_REF 100 // Reference voltage

#define ROOM\_TEMP 25.0 // Typical room temperature

#define TEMP\_THRESHOLD 30.0 // Temperature threshold for turning on the fan // Pin for controlling the fan

#define FAN\_PIN 21

// Global variable

volatile uint32\_t gpio\_pin\_state = 0;

int temperature;

int i;

// Function to initialize ADC

void init\_adc(void)

{

LPC\_SC -> PCONP |= (1 << 12); // Enable ADC power/clock

LPC\_PINCON -> PINSEL1 &= ~(3 << 28); // P1.30 is GPIO

LPC\_GPIO1 -> FIODIR &= ~(1 <<4); // Set P1.3 as input

LPC\_PINCON -> PINSEL1 |= (3 << 28); // P0.24 is ADC0.4

LPC\_ADC -> ADCR = (1 << 4) | (1 << 21); // Enable ADC0.4 channel

}

// Function to read ADC value

void read\_adc(void)

{

int adc\_value = rand();

LPC\_ADC -> ADCR |= (1 << 24); // Start ADC conversion

while (!(LPC\_ADC -> ADGDR & (1 << 31))); // Wait for conversion to complete.

temperature = (T\_REF \* adc\_value / 4096);

LPC\_ADC -> ADGDR = temperature >> 4;

// Calculate temperature in degrees Celsius

for (i = 0; i < 20000; i++);

}

// Function to initialize GPIO for controlling fan

void init\_fan(void)

{

LPC\_GPIO0 -> FIODIR |= (1 << FAN\_PIN); // Set fan pin as output

LPC\_GPIO0 -> FIOCLR |= (1 << FAN\_PIN); // Turn off fan initially

gpio\_pin\_state = 0;

}

//Main function

int main(void)

{

while (1)

init\_adc(); // Initialise ADC

init\_fan();

// Generate a random ADC value between 0 and 1023

read\_adc();

if (temperature >= TEMP\_THRESHOLD) {

LPC\_GPIO0 -> FIOSET |= (1 << 21); // Turn on the fan if temperature is above threshold

gpio\_pin\_state = 1;

// Add a delay to let the fan spin up

for (i = 0; i < 100000; i++);

} else

{

LPC\_GPIO0 -> FIOCLR |= (1 << 21); // Turn off fan if temperature is below threshold

gpio\_pin\_state = 0;

// Add a delay to let the fan spin down

for (i = 0; i < 100000; i++);

}

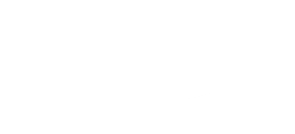
}

}

**5. OUTPUT:**

A screenshot of a computer

Description automatically generated



A screenshot of a computer

Description automatically generated