

Temperature Sensing and Display using SPI Communication with ATMEGA 328P

Nitin Kumar Maurya^{1*}, Khizar Ali², Mohd. Sazid³

^{1,2}*Student, Department of Electronics and Communication Engineering,
Noida Institute of Engineering and Technology, Greater Noida, India*

³*Professor, Department of Electronics and Communication Engineering,
Noida Institute of Engineering and Technology, Greater Noida, India*

***Corresponding Author**

E-Mail Id:- nitin27.rkt@gmail.com

ABSTRACT

This paper presents a simple and cost-effective temperature sensing and display system using SPI communication. The main components used are the LM35 temperature sensor, ATmega328P microcontroller, and a 16x2 LCD for display. The system reads analog temperature data from the sensor, converts it using the ADC (Analog to Digital Converter), and sends the information using SPI protocol. The temperature value is displayed in real-time on the LCD screen. This project demonstrates how SPI communication can be used effectively in small embedded systems for real-time applications, making it suitable for temperature monitoring in various fields like home automation and industry.

Keywords:- SPI, ATmega328P, LM35, LCD, embedded system, temperature monitoring

INTRODUCTION

Temperature monitoring is essential in various domains, including home automation, industrial processes, and agricultural systems. In many applications, it is crucial to accurately measure and display the temperature in real-time. This paper outlines the design and development of a temperature sensing and display system that uses SPI (Serial Peripheral Interface) communication for easy integration with other devices [1].

The main goal of this system is to continuously monitor the ambient temperature using an LM35 sensor, process the data using the ATmega328P microcontroller, and display it on a 16x2 LCD.

RELATED WORK

Various methods have been used for temperature sensing in embedded systems.

In many previous designs, protocols such as I2C or UART were used to interface between microcontrollers and sensors. Some systems relied on traditional analog thermometers.

However, compared to these protocols, SPI communication offers higher speed and is more suitable for applications requiring data exchange between multiple devices. SPI allows for faster and more reliable data transmission, which is why it has been chosen for this system. This paper demonstrates the use of SPI communication with a simple sensor setup.

SYSTEM OVERVIEW

LM35 Temperature Sensor

This analog sensor is used to measure temperature. It provides an output voltage that is directly proportional to the temperature in degrees Celsius [2].

ATmega328P Microcontroller

This microcontroller is used to process the analog signal from the LM35 sensor. It is capable of converting the analog signal to a digital value using its built-in ADC (Analog to Digital Converter).

16×2 LCD

The LCD is used to display the temperature reading in real-time.

SPI Communication Protocol

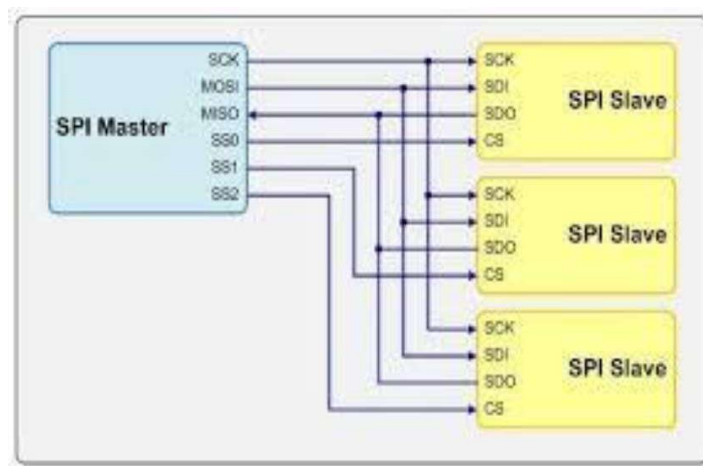
SPI is used to send data from the ATmega328P to other connected modules or microcontrollers, allowing for system expansion or integration with other devices [3].

CIRCUIT DESIGN AND WORKING

The LM35 sensor is connected to the ADC input pin (ADCO) of the ATmega328P. The output of the sensor is an analog voltage that is converted to a digital value using the ADC module of the microcontroller. The calculated temperature value is then passed to the 16x2 LCD for display.

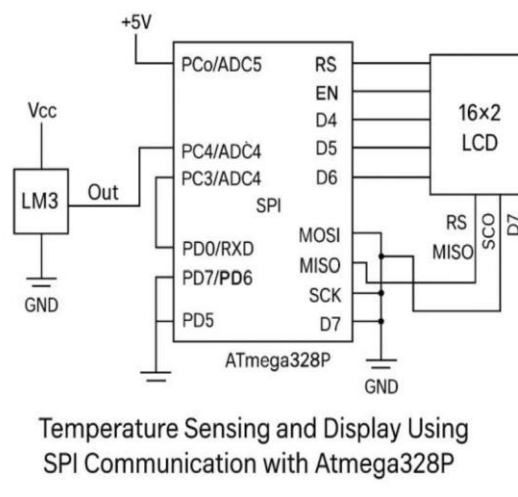
Circuit Connections

- Vcc: +5V to all components.
- GND: Common ground.
- Sensor Output: Connected to ADC0 (input of the ATmega328P).
- LCD: Connected to digital pins (RS, EN, D4—D7) in 4-bit mode.



The microcontroller reads the ADC value, applies the temperature calculation formula:

$$\text{Temp (in C)} = (\text{ADC Value} * 500) / 1024$$



SPI COMMUNICATION

SPI (Serial Peripheral Interface) is a full-duplex, synchronous communication protocol used for transmitting data between microcontrollers or other peripheral devices [4]. In this project, the SPI protocol is used to demonstrate how the temperature data can be sent to another microcontroller or device. The main lines used in SPI are:

MISO (Master in Slave Out)

- **MOSI (Master Out Slave In)**
- **SCK (Serial Clock)**
- **SS (Slave Select)**

This project utilizes the SPI protocol to enable data transfer for further processing or display, which is crucial for real-time applications in temperature monitoring systems.

```
#include <avr/io.h> #include <util/delay.h>
#include "lcd.h" // Custom LCD library
// Function to initialize ADC void init
ADC() {
ADMUX = (1<<REFS0); // Set reference
voltage to AVCC
ADCSRA = (1<<ADEN) | (1<<ADPS2) |
(1<<ADPS1) | (1<<ADPS0); // Enable ADC
and set pre scaler to 128
// Function to read ADC value
uint16_t read ADC (uint8_t channel) {
channel &= 0x07; // Make sure the channel
is between 0 and 7 ADMUX = (ADMUX &
0xF8) | channel; // Select ADC channel
ADCSRA |= (1<<ADSC); // Start
conversion
While (ADCSRA & (1<<ADSC)); // Wait
for conversion to finish return ADC; //
Return ADC value
int main(void) (uint16_t value; float
temperature;
LCD_init(); // Initialize the LCD
init_ADC(); // Initialize the ADC
while (1) {
value = read_ADC(0); // Read ADC value
from channel 0 temperature = (value *
500.0) / 1024.0; // Convert to temperature
LCD_cmd(0x80); // Move cursor to the top
left LCD rint("Temp: "); // Print label
```

```
LCD rint_float(temperature); // Print
temperature LCD rint(" C"); // Print Celsius
symbol
_delay_ms(1000); // Wait for 1 second
```

The Analog-to-Digital Converter (ADC) of the ATmega series microcontroller was configured using the in it ADC () function. In this setup:

- The reference voltage is selected as AVCC by setting the REFS0 bit in the ADMUX register.
- The ADC is enabled by setting the ADEN bit in the ADCSRA register.
- The prescaler is set to 128 by setting ADPS2:0 bits to 111', which ensures proper ADC timing by dividing the main clock.

```
value = read_ADC(0);
temperature = (value * 500.0) / 1024.0;
```

This formula converts the digital ADC reading to the corresponding temperature in Celsius.

```
/*LCD_cmd(0x80), LCD_print("Temp: "),
LCD_print_float(temperature), LCD_print("
C"); */
LCD_cmd(0x80) sets the cursor to the
beginning of the first line. LCD rint float()
displays the floating-point temperature
value.
```

```
/*while (1) (
_delay_ms(1000), 1 *
```

The system operates in an infinite loop, updating the temperature on the LCD every 1 second using delay ms (1000).

SERIAL PERIPHERAL INTERFACE TYPES Standard SPI

This is the most common type of SPI, featuring a single master device communicating with one or more slave devices using separate chip select lines. It operates in full-duplex mode, allowing simultaneous data transmission and reception.

Single SPI (SSI)

Single SPI, or SSI, involves a single master device communicating with a single slave device. Unlike standard SPI, SSI uses a shared chip select line for all connected slave devices. This simplifies the hardware configuration but limits communication to one slave at a time.

Multi-Master SPI

In multi-master SPI, multiple master devices share communication with one or more slave devices. Each master can initiate communication independently, enabling more complex network configurations. However, proper coordination is necessary to prevent data collisions.

Daisy chain SPI

Daisy Chain SPI allows multiple slave devices to be connected in a chain configuration, with data passing through each slave sequentially. This reduces the number of chip select lines required, making it suitable for applications with limited pins. However, it may introduce delays as data propagates through the chain.

Quad SPI (QSPI)

Quad SPI enhances standard SPI by increasing data transfer rates using four data lines (MOSI, MISO, and two clock lines). This parallel data transfer enables faster communication, making it ideal for high-speed applications such as flash memory and displays [5].

Serial Peripheral Interface bus (SPI Bus)

SPI Bus is a variation of SPI that employs a shared data bus, allowing multiple master and slave devices to communicate over the same set of data lines. It utilizes additional control signals for arbitration and synchronization, ensuring orderly communication in multi-device environments.

Enhanced Serial Peripheral Interface (eSPI)

eSPI is an advanced version of SPI designed for modern computing systems. It offers features like increased data rates, extended address space, and additional functionalities such as system management and security. It is commonly used in embedded systems, servers, and networking equipment [6].

Each type of SPI offers distinct advantages and is suited for different applications based on factors such as speed, simplicity, and system requirements. Understanding these variations allows developers to choose the most appropriate SPI configuration for their specific project needs, ensuring efficient and reliable communication between devices.

RESULTS AND DISCUSSION

The system was tested by varying the environmental temperature and observing the corresponding changes on the LCD. The temperature displayed on the LCD was accurate and updated in real-time. The sensor response was fast and consistent. Additionally, the SPI communication feature was successfully tested by connecting the ATmega328P to another microcontroller. The temperature data was transmitted without any errors, indicating the reliability of the SPI protocol in this context.

Example Output

Room Temp: 29.0°C

Hand Touch: 33.5°C

This data confirms that the system is capable of handling real-time temperature variations and transmitting data reliably [4].

APPLICATIONS OF SERIAL PERIPHERAL INTERFACE

The SPI protocol finds application in various electronic systems where efficient and high-speed communication between multiple devices is required. Here are some common applications:

Sensor networks

SPI is widely used in sensor networks where microcontrollers communicate with sensors to collect data. For example, in weather stations, SPI facilitates communication between the main controller and sensors measuring temperature, humidity, and pressure.

Memory Devices

SPI is used in memory devices like flash memory chips and EEPROMs to store and retrieve data. These devices can be accessed and controlled by microcontrollers using SPI, enabling efficient data storage in embedded systems [8].

Display Modules

SPI is employed in display modules such as LCD screens and OLED displays. Microcontrollers send display data to these modules via SPI, allowing for rendering text, graphics, and images on various electronic devices like digital clocks, MP3 players, and wearable devices.

Communication Modules

SPI is used in communication modules like Wi-Fi modules, Bluetooth modules, and RF transceivers. Microcontrollers communicate with these modules via SPI to establish wireless connectivity and exchange data with other devices or networks.

Motor Control

SPI is employed in motor control applications where microcontrollers communicate with motor driver ICs to control the speed and direction of motors. This enables precise and efficient motor control in robotics, industrial automation, and automotive systems [7].

Audio Interfaces

SPI is utilized in audio interfaces for digital audio transmission between microcontrollers and audio codecs or DACs (Digital-to-Analog Converters). This allows for high-quality audio playback in applications such

as musical instruments, audio players, and home entertainment systems.

Industrial Control Systems

SPI is employed in industrial control systems for communication between programmable logic controllers (PLCs), sensors, actuators, and other control devices. This enables real-time monitoring and control of industrial processes, enhancing efficiency and productivity.

Data acquisition Systems

SPI is used in data acquisition systems to interface analog-to-digital converters (ADCs) and digital-to-analog converters (DACs). Microcontrollers communicate with these converters via SPI to convert analog signals to digital data for processing or vice versa.

Automotive Electronics

SPI is employed in automotive electronics to communicate between microcontrollers, sensors, actuators, and electronic control units (ECUs). This enables various automotive functionalities such as engine control, vehicle diagnostics, and infotainment systems.

Embedded Systems

SPI is widely used in embedded systems for communication between microcontrollers and peripheral devices. Its simplicity, efficiency, and flexibility make it a preferred choice for interfacing diverse components in embedded applications across industries.

CONCLUSIONS

This project demonstrates how to build a simple yet efficient temperature sensing and display system using SPI communication with the ATmega328P microcontroller. The system is easy to implement, cost-effective, and suitable for a variety of applications, including home automation, industrial monitoring, and environmental systems. The flexibility of SPI allows for future upgrades, such as wireless communication or

integration with cloud-based platforms for remote monitoring. The system can be enhanced further by adding features like data logging, wireless transmission, or even integration with more advanced sensors for higher precision. This project serves as an excellent introduction to embedded systems, microcontrollers, and communication protocols like SPI.

REFERENCES

1. Microchip Technology. (n.d.). *ATmega328P datasheet*. <https://www.microchip.com/wwwproducts/en/ATmega328P>.
2. Texas Instruments. (n.d.). *LM35 temperature sensor datasheet*. <https://www.ti.com/product/LM35>.
3. Mazidi, M. A. (n.d.). *The AVR microcontroller and embedded systems*. Pearson Education.
4. Microchip Technology. (n.d.). *SPI communication protocol: Application notes*. <https://www.microchip.com>.
5. Kamal, R. (n.d.). *Embedded systems: Architecture, programming and design*. McGraw-Hill Education.
6. Barnett, R. H., Cox, S., & O'Cull, L. (n.d.). *Embedded C programming and the Atmel AVR*. Cengage Learning.
7. Arduino. (n.d.). *SPI library*. <https://www.arduino.cc/en/Reference/SPI>.
8. Circuit Digest. (n.d.). *Tutorial: Interfacing LM35 temperature sensor with AVR microcontroller*. <https://circuitdigest.com/microcontroller-projects/interfacing-lm35-temperature-sensor-with-avr-microcontroller>.