

Dual-Screen Arduino-ESP8266 Smart Mirror with Linux Terminal Control and IoT Data Integration

Arnav Sharma¹

¹Student, B.E. – Software Engineering, Chitkara University, Punjab, India

Abstract - This paper presents the design and implementation of a low-cost dual-screen smart mirror built around an Arduino Uno microcontroller and an ESP8266 Wi-Fi module. Two 1.8-inch TFT LCDs are mounted behind a partially reflective acrylic mirror and used to display time, date, weather conditions, indoor and outdoor temperature, a to-do list and a simple news headline widget. The ESP8266 connects to the internet, obtains network time and weather data from public web APIs, formats the information as serial messages and forwards it to the Arduino over a software serial link. The Arduino is dedicated to sensor acquisition and graphics rendering, driving both TFT displays using the ST7735 controller. A PIR motion sensor enables automatic activation of the screens when the user is present, while a DHT11 sensor provides indoor temperature measurements that are combined with the outdoor temperature from the weather API. A simple Linux command-line interface allows the user to choose, in real time, which information widget should be shown on each screen. The resulting system demonstrates how microcontroller-class hardware can be used to prototype an IoT-enabled smart mirror without requiring a full single-board computer such as a Raspberry Pi.

Key Words: Smart mirror, Internet of Things (IoT), Arduino, ESP8266, TFT display, PIR sensor, DHT11, Linux CLI.

1. INTRODUCTION

Smart mirrors combine a reflective surface with an embedded electronic display to present contextual information such as time, weather, calendar events or notifications while still functioning as a regular mirror. Most published designs rely on a single large display and a single-board computer, for example a Raspberry Pi running a web-based smart-mirror framework. In contrast, this project explores a different design point: a microcontroller-based smart mirror that uses two small TFT displays driven directly by an Arduino Uno, with a separate ESP8266 NodeMCU module responsible for all networking tasks.

The primary goals of the system are: (i) to demonstrate that dual-screen smart-mirror functionality can be achieved using modest 8-bit hardware, (ii) to separate network-facing code and API keys from the display

controller for better modularity and security, and (iii) to provide a flexible interaction model in which the user can dynamically assign different information widgets to either display via a simple Linux terminal interface. The finished prototype is shown in Fig. 2 and Fig. 3 (front view and internal wiring respectively), and a short demonstration video is available online. [9].

1.1 RELATED WORK

Numerous smart-mirror implementations have been reported in both hobbyist communities and academic literature. Many designs are based on the MagicMirror² framework running on a Raspberry Pi, where a web browser renders HTML widgets showing time, weather and news feeds behind a two-way mirror. Several research papers describe IoT-enabled mirrors that display date, time and weather while integrating features such as voice assistants, home-automation control or security cameras. However, these architectures usually assume the presence of an operating system and relatively high processing and memory resources.

Closer to the embedded domain, there are designs that employ an Arduino Uno with a single LCD or TFT display to show basic information, but these typically treat the mirror as a single combined display surface. The design presented in this paper differs in three ways: it uses two independent TFT panels, it delegates all Internet communication to a separate ESP8266 module, and it introduces a Linux command-line interface as a lightweight yet powerful control mechanism for selecting the content of each screen.

2. SYSTEM OVERVIEW

The smart mirror system consists of four main subsystems: (1) the sensing and display controller built around an Arduino Uno, (2) the Wi-Fi and web-API subsystem implemented using an ESP8266 NodeMCU board, (3) the mirror assembly containing two 1.8-inch ST7735-based TFT displays mounted behind a reflective acrylic sheet, and (4) a Linux PC that acts as a control console via the serial port.



Fig - 1: Dual Screen Output

The NodeMCU connects to the home Wi-Fi network and periodically obtains network time using an NTP client. It then calls a public weather API for the city of Chandigarh and parses the resulting JSON payload to extract the current temperature and a short textual weather description. It also generates static strings representing a to-do list and calendar information using the TimeLib library. All of this information is formatted as labelled lines, for example "Time:HH:MM:SS" or "Weather:clear sky", and transmitted over a UART link to the Arduino.

The Arduino Uno receives these labelled lines on a software serial port implemented with the AltSoftSerial library. It stores the latest values in string buffers and combines them with indoor temperature readings obtained from a DHT11 sensor. A PIR motion sensor connected to a digital input is used to decide when the mirror should be active. When the sensor detects motion, the Arduino restores the most recent content on both TFT displays and starts a thirty-second timer. If no further motion is detected before the timer expires, the displays are cleared to black to reduce power consumption.

3. HARDWARE DESIGN

The prototype is built in a cardboard enclosure sized to accommodate the mirror and electronics. Two 1.8-inch TFT modules based on the ST7735 controller are mounted behind cut-outs in an acrylic mirror sheet so that the displayed text appears to float within the reflective surface. Each TFT uses an SPI interface plus dedicated chip-select, data/command and reset lines. To avoid contention, the two displays share the SPI clock and data lines but have separate chip-select and reset pins.



Fig - 2: Front View

The Arduino Uno is responsible for driving both displays and reading the local sensors. A DHT11 module connected to a digital pin provides indoor temperature and humidity readings, while a PIR motion sensor mounted on the front panel detects user presence. The NodeMCU ESP8266 board is mounted inside the enclosure and powered from the same 5 V USB supply as the Arduino, with on-board regulation providing 3.3 V for the Wi-Fi module. A simple UART connection implemented using AltSoftSerial carries data from the NodeMCU to the Arduino. The Linux PC is connected to the Arduino's primary USB-serial interface.

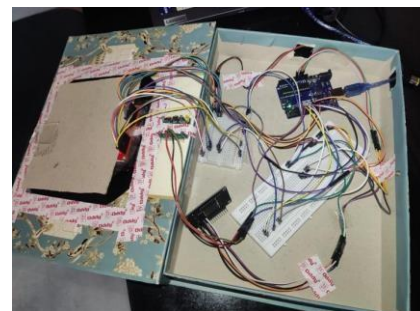


Fig - 3: Internal Wiring

4. SOFTWARE ARCHITECTURE

The software is divided into three main programs: firmware for the ESP8266, firmware for the Arduino Uno, and a lightweight Linux terminal client. The ESP8266 firmware is responsible for network connectivity and API access. After joining the configured Wi-Fi network, it uses an NTP client to synchronise its internal clock and queries a weather API endpoint exposed by OpenWeather. The HTTP response is parsed using ArduinoJson to extract the current outdoor temperature and a brief weather description string. The firmware also constructs a simple calendar string and a fixed to-do list.

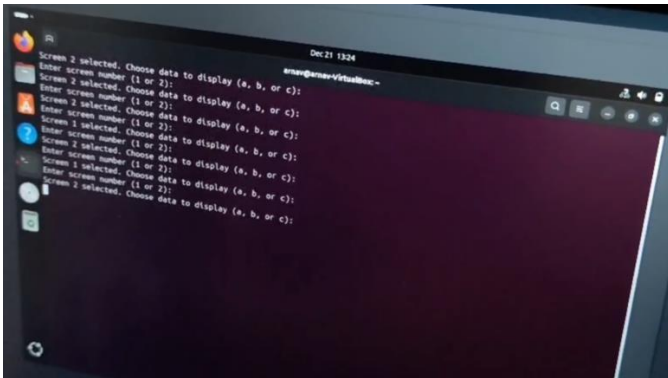


Fig - 4: CLI Output

Periodically, the ESP8266 prints a sequence of labelled lines to its serial port: current time, weather description, outdoor temperature, calendar string, to-do list header and a placeholder for news. On the Arduino side, these messages are read from the AltSoftSerial port, and a parser function updates the corresponding global string variables. Indoor temperature readings from the DHT11 are combined with the outdoor temperature string to create a composite message showing both values.

The main loop on the Arduino monitors the PIR sensor and the USB serial port connected to the Linux PC. When motion is detected, the loop enables the displays and processes any pending user commands from the terminal. The user is first prompted to select a screen number (1 or 2) and then to choose a widget option labelled "a", "b" or "c". For screen 1, option "a" selects the time widget, "b" selects the weather description and "c" selects the combined indoor/outdoor temperature view. For screen 2, option "a" selects date and day, "b" selects the to-do list and "c" selects the news string. A small rendering function clears the chosen TFT, sets font and colour parameters and prints the requested text.

5. RESULTS AND DISCUSSION

The prototype smart mirror successfully demonstrates dual-screen operation using an Arduino-class microcontroller. In normal use the PIR sensor reliably detects a person approaching the mirror, at which point both TFT displays wake from a blank state and restore their last contents. The Linux terminal interface provides a convenient way to experiment with different combinations of widgets on the two screens without needing to recompile firmware. Users can, for example, keep time and weather on the top display while cycling between calendar, to-do list and news on the lower display.

Because the ESP8266 handles all HTTP requests and JSON parsing, the Arduino firmware remains relatively simple and timing-deterministic, focused mainly on sensor handling and graphics. This separation of concerns also

keeps API keys and network credentials on the Wi-Fi module, which could, in a more advanced design, be replaced or updated independently of the mirror hardware. The main limitations of the current prototype are the limited resolution and viewing angle of the small TFT displays and the absence of advanced features such as user recognition or voice interaction. Nevertheless, the system serves as a useful platform for experimenting with IoT-connected ambient displays on constrained hardware.

6. CONCLUSION

This paper has described the development of a dual-screen smart mirror that combines an Arduino Uno, an ESP8266 Wi-Fi module, motion and temperature sensors and two 1.8-inch TFT displays. By delegating internet connectivity and API integration to the ESP8266 and reserving the Arduino solely for sensor management and display control, the design demonstrates an effective division of labor between microcontrollers. The Linux terminal interface provides a flexible mechanism for selecting which information is shown on each display without requiring a graphical configuration tool.

Future work could include migrating to higher-resolution IPS displays, integrating additional data sources such as calendar services or task managers, and exploring richer interaction techniques, for example voice control or gesture recognition. The present implementation, however, already achieves the core smart-mirror experience with inexpensive hardware and relatively simple firmware, making it well suited for educational settings and DIY home-automation projects.

REFERENCES

- [1] M. Teipen et al., "MagicMirror²: An open modular smart mirror platform," MagicMirror² Project, online: <https://magicmirror.builders/>
- [2] "IoT Based Smart Mirror Using Raspberry Pi," Int. J. Eng. Res. Technol. (IJERT), vol. 8, no. 6, 2019.
- [3] "IoT based Smart Mirror using Raspberry Pi," Int. J. Recent Adv. Sci. Eng. Technol. (IJRASET), 2018.
- [4] OpenWeather, "Weather API," <https://openweathermap.org/api>, accessed 2025.
- [5] NewsAPI.org, "News API Documentation," <https://newsapi.org/>, accessed 2025.
- [6] Arduino, "Arduino Uno Rev3," <https://docs.arduino.cc/hardware/uno-rev3>, accessed 2025.
- [7] Espressif Systems, "ESP8266EX Datasheet," <https://www.espressif.com/>, accessed 2025.

- [8] A. Sharma, "Smart Mirror Arduino–Linux CLI Project Report," SIT-111 Task 3.8HD, Chitkara University, 2023.

- [9] A. Sharma, "SIT 111 | 3.8 HD | Smart Mirror Implementation",
<https://www.youtube.com/watch?v=XgDFLYp4BZM>