EECS 3215 Final Project: Controlling an RGB LED using a nodeMCU board

Michael Fahimi, 217382508, michaelfahimi777@gmail.com

Rynand Li, 218419721, rynandli@my.yorku.ca

Farnood Rashidzadeh, 216322380, farnoodr@yorku.ca

Kyle Truong, 218160317, kytrng@my.yorku.ca

Rikesh Boodhun, 216508673, panda2@my.yorku.ca

Department of Electrical Engineering and Computer Science, York University

EECS 3215 M: Embedded Systems

Professor Navid Mohaghegh

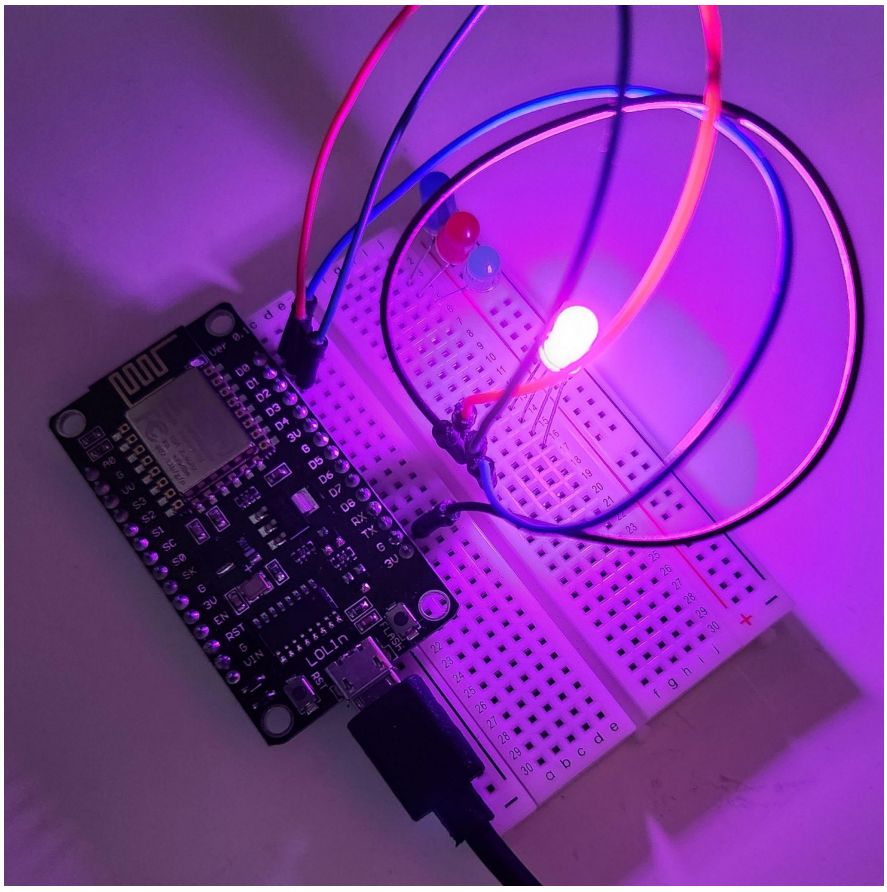
April 9, 2023

EECS 3215 Final Project: Controlling an RGB LED using a nodeMCU board

This report describes controlling an RGB LED light with nodeMCU and ESP-12 processor. The main objective of this project was to expand the knowledge of embedded systems. The team used a Lolin nodeMCU board with an ESP8266MOD processor. This project uses Arduino and Arduino IED for programming the board; however, the processor's firmware is based on Lua Script. Team members successfully implemented the project and controlled the device through an HTML-based interface in the local network.

Figure 1

An image of the final device



Introduction

Many tech-related aspects of daily life involve embedded systems, from the production level to the final products. IoT devices are among the most famous systems that use embedded systems. With the significant rise in their usage in recent years, one can only pay more attention to the different aspects of IoT devices.

The nodeMCU is an open-source board with an ESP82266 wifi chip, which makes it an excellent tool for this project. The RGB LED connects to the board's general input-output (GPIO) and ground pins and provides an opportunity for testing different modes throughout the project. After programming the board with Arduino, A simple HTML-based interface accessible in the local network, can control the LED.

This report describes the research and implementation of the described system, including details on the board, processor, used material, and software. The team will also provide the course instructor with a system representation.

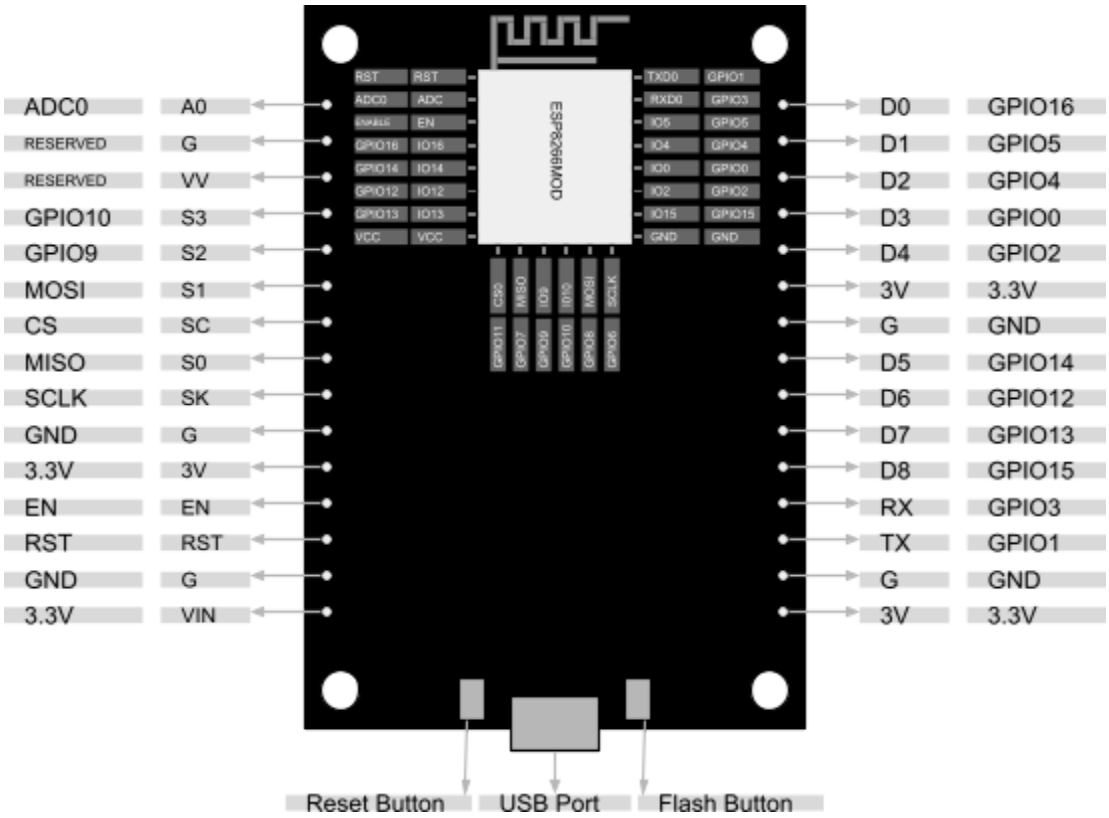
Materials and Methods

The materials used in this project, including the software and hardware, are described below.

Lolin NodeMCU Board v3: This board is produced by Lolin and features an ESP-12 (ESP8266MOD) wifi chip. In this project, the ESP-12 acts as the processor unit as well. Hysiri made the processor on this board. A map of pins is provided below (Hysiri, 2016).

Figure 2

Lolin NodeMCU based on the manual description.



- Useful System Specifications:
    - Operating Voltage: 3.3V(3.0 ~ 3.6V) - Temperature:-40 - 85°C
    - CPU Tensilica L106: RAM 50KB(Available) - Flash: 8 Mbit
    - System: 802.11 b/g/n (Frequency range: 2.4 GHz ~ 2.5 GHz(2400 M ~ 2483.5M))
    - IntegratedTensilica L106 ultra-low power 32-bit micro MCU, with 16-bit RSIC. The CPU clock speed is 80MHz (maximum value of 160MHz).
    - WIFI 2.4 GHz (supports WPA/WPA2)
    - Supports UART、I2C、GPIO、PWM、SDIO、SPI、ADC、PWM、IR
    - Integrated 10-bit high-precision ADC

- Supports TCP、UDP、HTTP、FTP
- Integrated TR switch, balun, LNA, Power amplifier and matching network
- Integrated PLL, Regulator and power source management components, +20 dBm output power in
- 802.11b mode
- Average working current 80mA, <Deep sleep current < 20uA, Power down leakage current < 5uA
- The rich interface on the processor: SDIO 2.0, SPI, UARTl
- Wake up, build the connection and transmit packets in < 2ms
- Standby power consumption < 1.0mW (DTIM3)Support AT remote upgrades and cloud OTA upgrade support Station / SoftAP / SoftAP+Station operation modesUltra-Small 20.2mm * 17.6mm * 3.05mm

- MCU Pins (Hysiri, 2016):
  - IO[0,4,5,9,10,16] - GPIO[0,4,5,9,10,16] Respectively
  - IO2: GPIO2; UART1_TXD
  - IO12: GPIO12; HSPI_MIS0 MIDO; HSPICS; UART0_RTS
  - IO13: GPIO13; HSPI_MOSI; UART0_RTS
  - IO14: GPIO14; HSPI_CLK
  - EN: Enable (Chip enable pin, cannot be floating, Active High), GPIO14; HSPI_CLK
  - IO15: GPIO15; MTDO; HSPICS; UART_RTS
  - TXD: UART0_TXD; transmit end in UART download, floating(internal pull-up) or pull-up; GPIO1

- VCC: 3.3 V power supply (VDD) (recommended: power supply max current be 500mA or above); GPIO2
- RXD: GPIO3 - UART0_RXD, receive end in UART download; GPIO3
- MOSI: MOSI | SCLK: CLK | GND: ground
- RST: Reset module
- TOUT: Tests the power supply voltage of VDD3P3 and the input power voltage of TOUT. These two functions cannot be used at the same time.
- CS0: chip Selection | MISO: MISO

Common Cathode RGB LED Light: This LED light provides three positive pins, one for each red, green, and blue colour, and a common negative pin.

Jumper Wires and Breadboard: To Connect the LED and the NodeMCU board.

Power Supply Cable: To connect the board to the power source and to programme the board.

Router: Provides the local network for controlling the board.

Arduino IDE: The IDE for programming the board.

ESP8266Wifi library: A premade library for ESP-12 wifi Connection.

The following section includes the steps taken to implement the final version of the project:

1. The board and the LED must be connected using a breadboard and jumper cables. In this project, GPIO5, GPIO4, and GPIO16 were chosen for the LED light's red, green, and blue pins. Finally, the common cathode must be connected to the board's GND pin.
2. Board is connected to a computer using the power supply cable.

3. Necessary Libraries and Arduino IDE must be installed on the computer.
4. The Arduino code is compiled and uploaded into the board.
5. The user face can be accessed through a device connected to the same router as the board to control the light.

Following the procedure must result in the final version of the device.

Discussion and Analysis

This section takes a deeper look at the details of this project, including the code, the software and the challenges the team had to resolve.

Our initial challenge faced was searching for reference material that we could use to help us develop our project. Finding an in-depth manual for the nodeMCU was difficult initially; however, we came across an open-source reference guide explaining the pins we needed for our specific use case.

Initially, we intended to program our project in C; however, the nodeMCU board interpreted the code in Lua or Arduino; thus, it was advised for us to choose between those two languages. The challenging part of this process was learning to use a different language, as none of our group members were familiar with this IDE. Arduino uses a variant of C++, and the use of source code found on the internet made it a simple process for implementing our project.

The project's main goal was to create a system that can be remotely accessed via a smartphone to turn the LED on and off. The first step was to establish a wireless connection with the board and create a function that would activate only one of the switches; an open-source library containing a program that establishes a network host on the nodeMCU was utilized. An HTML code was written to create a simple webpage with a button that turned on or off a single LED that the client controls.

The nodeMCU was attached to a breadboard where a LED was attached to a power rail corresponding to one of the GPIO pins. To turn on the LED, we needed to change the corresponding pin to output and the voltage to HIGH, which will send power to the LED. In Arduino, the code pinMode(pin, OUTPUT) sets the pin to output and digitalWrite(pin, HIGH) sets the pin to HIGH voltage.

The next step was to have three red, green and blue LEDs that could be turned on and off. Like the first step, we needed to change the corresponding pin to those colours to output a HIGH voltage to turn on the LED. To implement, we initialized an additional two buttons that corresponded to an LED pin and had a function listening to the button that would set the pin to high or low. The final step was to create an RGB light that can produce multiple colours by activating the red, green and blue in different sets. On the client page, the option to independently activate each LED was removed with a list of colours. For example, if purple were activated, the blue and red LED would be turned on to produce purple.

The Code: The code for this project is written in Arduino and using Arduino IDE. The file for the code is submitted alongside this report.

Our code uses the Wi-fi Library name <ESP8266WiFi.h> for a client to connect to our server. The Wi-fi SSID is nodeMCUX, and the Wi-fi password is ourGroup@3215. The Wi-fi server is set to 80.

GPIO16, GPIO5, and GPIO4 connect to the LED lights. GPIO16 connects to the blue LED, GPIO5 connects to the red LED, and GPIO4 connects to the green LED.

The function initLights() initializes the pin setup function. We set all the GPIO pins connected to the LED to OUTPUT using the function pinMode(GPIO, I/O). Then we use the function digitalWrite(GPIO, HIGH/LOW) to set the initial value to LOW.

The function wifiConnect() initiates the Wi-Fi connection. Serial printout “Connecting to nodeMCU.” It will then keep printing out “.” until it connects to the Wi-Fi, which will print "WiFi connected." The server will now start, as indicated by "Server started." It will also print out the IP address.

The function turnOnColor(int r, int g, int b) sets the RGB pins to HIGH or LOW.

The function setup() setup the serial and initializes the LED pins and the Wi-Fi.

The function loop() serves multiple functions.

- It checks if a client has connected. If a client has connected, the serial will print “new client.”
- It waits until the client sends some data.
- It reads the first line of the request by the client and matches the requested colour.

The requests that the client can make are LED=OFF, LED=ON, LED=RED, LED=GREEN, LED=BLUE, LED=CYAN, LED=MAGENTA, LED=YELLOW, LED=WHITE, and LED=AMAZING.

The request LED=AMAZING will switch colours from red, yellow, green, cyan, blue, and then magenta with a delay of 250 ms for each colour change.

At the end of the code, it will return the response and handle the HTML setup.

Conclusion

In conclusion, this project demonstrated the potential of embedded systems in the context of IOT devices and home automation. The nodeMCU board device can control an LED through an HTML-based user interface in the local network. Team members achieved their objectives in this project and expanded their knowledge of embedded systems.

In the future, this project can be expanded and include more advanced features, including dimming ability and using modules such as PWM and interrupts. Overall the project provided a practical example of implementing an embedded system for an IoT device.

References

Grokhotkov, I. (2017). *Reference — ESP8266 Arduino Core 3.1.2-7-g65579d29 documentation*. Read the Docs. Retrieved March 30, 2023, from https://arduino-esp8266.readthedocs.io/en/latest/reference.html

Hysiri. (2016). ESP82266 Wifi Module Datasheet. In *FCCID*. Shenzhen Hysiry Technology Co.,Ltd. Retrieved March 28, 2023, from https://fccid.io/2AKBPESP8266/User-Manual/User-Manual-3192780.pdf