Taneem Elmaasrawy

900163075

Embedded Systems

Final Project

Simple IoT Application

Dr. Mohamed Shalan

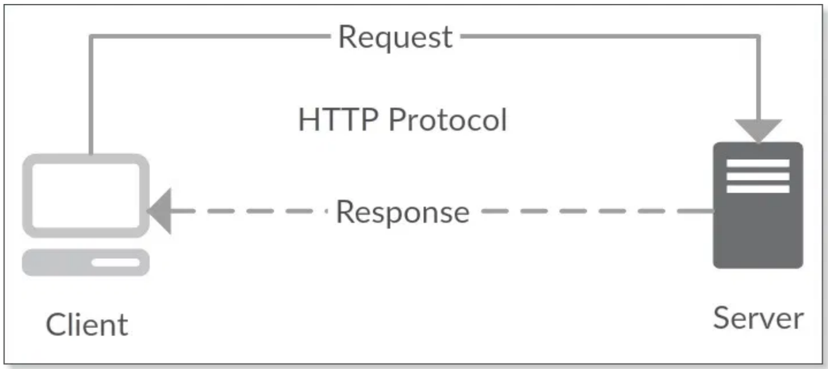
Content

Description

The project utilizes the ESP8266 module to create a small IoT application that can enable the user to perform I/O operations with the STM32 module through a web interface. The I/O operation includes retrieving date and time from the STM32 and controlling the LEDs.

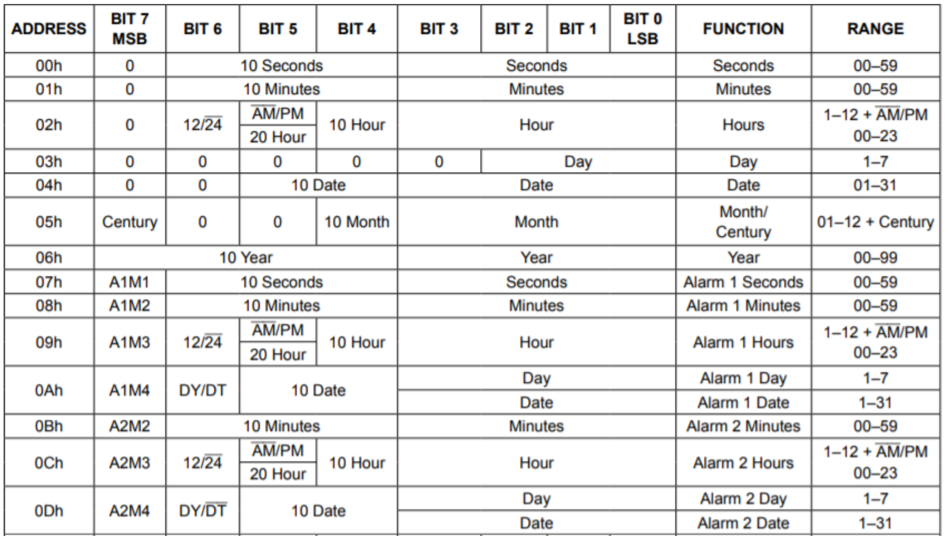
System Architecture

The system uses client-Server architecture. the web interface is the client as it is the one that initiates the request and the code uploaded on STM32 is the server as it responds when the request is initiated by the client and acts accordingly either by sending the date and time or by toggling the LED.



RTC module

- The module is a Real Time Clock that is used to provide date and time information to send it to the STM32 module by connecting SCL to PB6 and SDA to PB7.
- The Receive and Transmit operations is done using I2C synchronous communication where STM32 is the master and RTC is the slave.
- The RTC registers is set according to the following table.

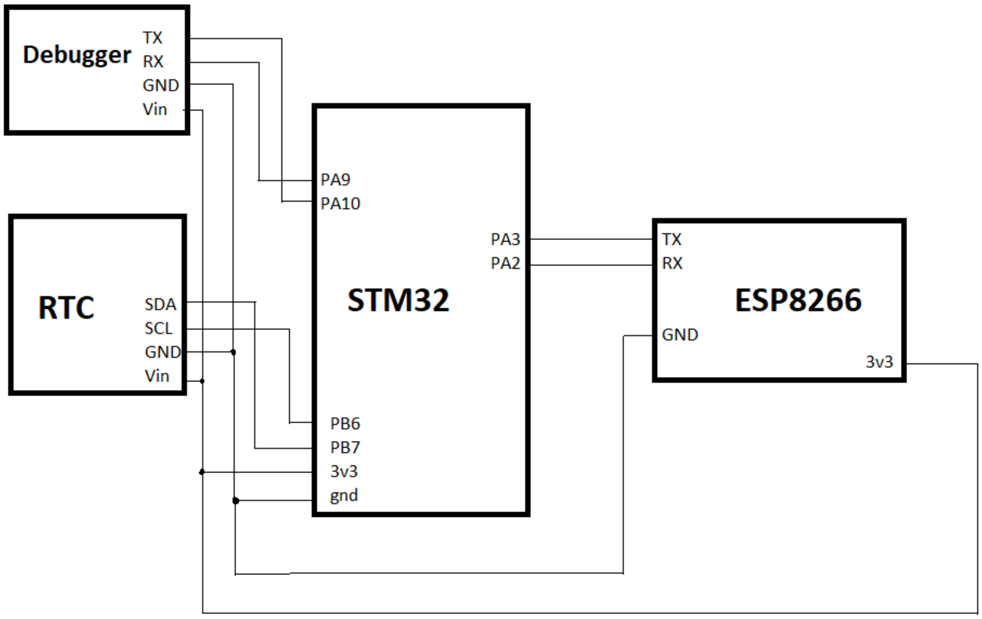


ADDRESS	BIT 7 MSB	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0 LSB	FUNCTION	RANGE
00h	0	10 Seconds			Seconds				Seconds	00–59
01h	0	10 Minutes			Minutes				Minutes	00–59
02h	0	12/$\overline{24}$	$\overline{AM}$/PM 20 Hour	10 Hour	Hour				Hours	1–12 + $\overline{AM}$/PM 00–23
03h	0	0	0	0	0	Day			Day	1–7
04h	0	0	10 Date		Date				Date	01–31
05h	Century	0	0	10 Month	Month				Month/ Century	01–12 + Century
06h	10 Year				Year				Year	00–99
07h	A1M1	10 Seconds			Seconds				Alarm 1 Seconds	00–59
08h	A1M2	10 Minutes			Minutes				Alarm 1 Minutes	00–59
09h	A1M3	12/$\overline{24}$	$\overline{AM}$/PM 20 Hour	10 Hour	Hour				Alarm 1 Hours	1–12 + $\overline{AM}$/PM 00–23
0Ah	A1M4	DY/DT	10 Date		Day				Alarm 1 Day	1–7
					Date				Alarm 1 Date	1–31
0Bh	A2M2	10 Minutes			Minutes				Alarm 2 Minutes	00–59
0Ch	A2M3	12/$\overline{24}$	$\overline{AM}$/PM 20 Hour	10 Hour	Hour				Alarm 2 Hours	1–12 + $\overline{AM}$/PM 00–23
0Dh	A2M4	DY/$\overline{DT}$	10 Date		Day				Alarm 2 Day	1–7
					Date				Alarm 2 Date	1–31

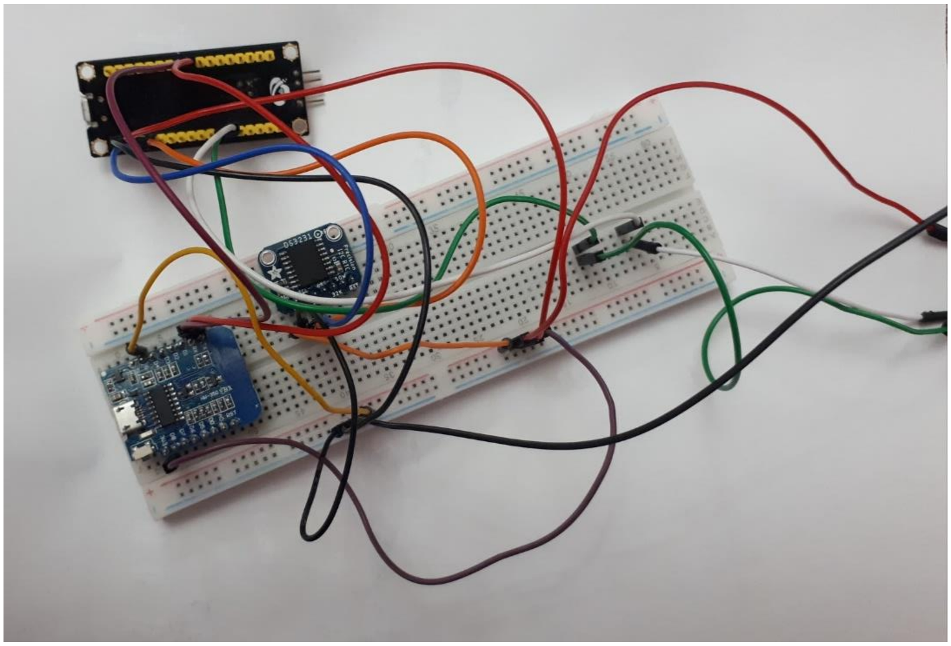
ESP8266 module

- It acts as a WiFi Access Point that connects the STM32 module with the Web interface. ESP8266 is connected with the STM32 module by connecting TX and RX pins to the MCU UART2 (PA3&PA2).
- In order to use it on Arduino, the ESP8266 package by ESP8266 community must be installed.

Block Diagram



Connections

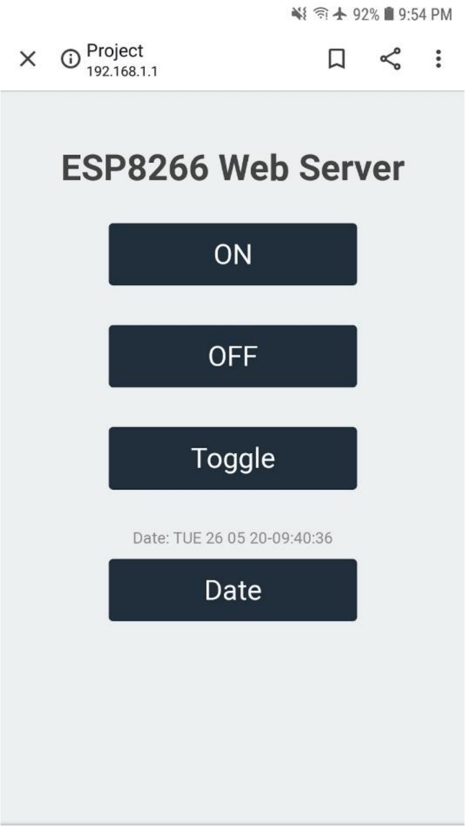


Inputs

The web interface sends commands to the STM32 according to the button pressed by the user:

Button	Command	Function
ON	‘n’	Turn the led on
OFF	‘f’	Turn the led off
Toggle	‘t’	Toggles the led
Date	‘d’	Retrieve date and time

Output



ESP8266 Web Server

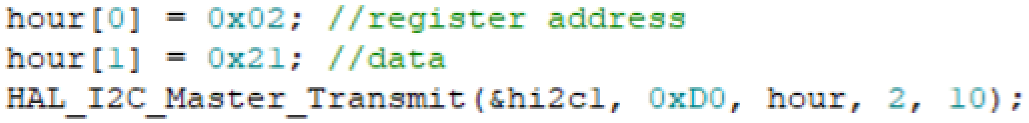
Date: TUE 26 05 20-09:40:36

Keil Code

- In order to set the RTC data, the buffer that contains the register address and the data has to be transmitted using HAL_I2C_Master_Transmit () function.
- Data is set according to the datasheet for each register.

  For example: 21 → 0010 0001

  Where bit 4 determines whether the data is greater than 10 hours or not and bit 5 determines if it is greater than 20. And the first 4 bits holds the remaining



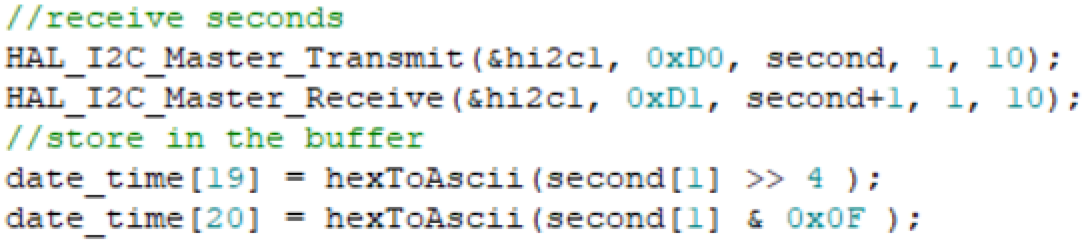
```
hour[0] = 0x02; //register address
hour[1] = 0x21; //data
HAL_I2C_Master_Transmit(&hi2c1, 0xD0, hour, 2, 10);
```

- In order to receive the updated time:

  1- transmit the address of the register to RTC module.

  2- receive data.

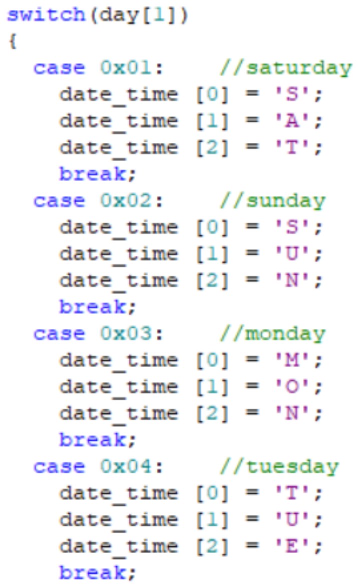
  3- convert data from hexadecimal to ASCII.



```
//receive seconds
HAL_I2C_Master_Transmit(&hi2c1, 0xD0, second, 1, 10);
HAL_I2C_Master_Receive(&hi2c1, 0xD1, second+1, 1, 10);
//store in the buffer
date_time[19] = hexToAscii(second[1] >> 4 );
date_time[20] = hexToAscii(second[1] & 0x0F );
```

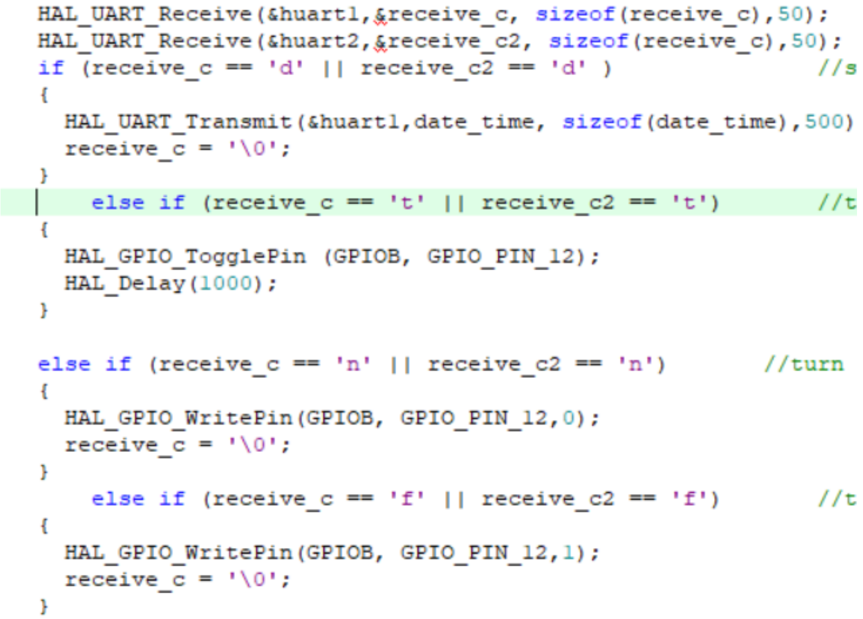
- This part converts the date from a numeric value to week day.

  Saturday: 1 & Friday: 7



```
switch(day[1])
{
  case 0x01:    //saturday
    date_time [0] = 'S';
    date_time [1] = 'A';
    date_time [2] = 'T';
    break;
  case 0x02:    //sunday
    date_time [0] = 'S';
    date_time [1] = 'U';
    date_time [2] = 'N';
    break;
  case 0x03:    //monday
    date_time [0] = 'M';
    date_time [1] = 'O';
    date_time [2] = 'N';
    break;
  case 0x04:    //tuesday
    date_time [0] = 'T';
    date_time [1] = 'U';
    date_time [2] = 'E';
    break;
```

- Receive command either from UART1 which is connected to the debugger and Tera Term or from UART2 which is connected to the ESp8266 module to receive command from the web interface.



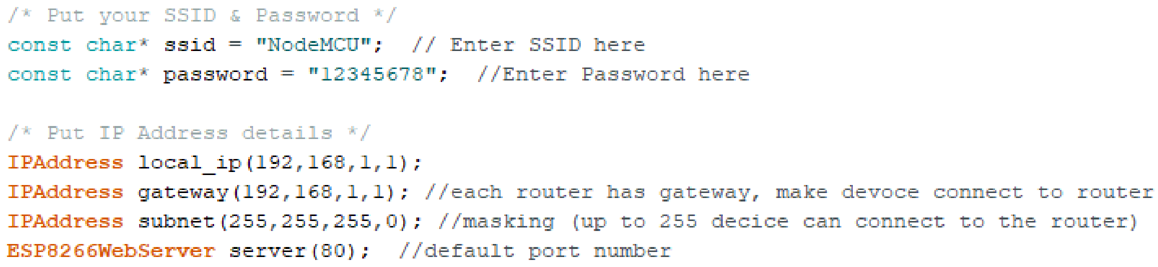
```
HAL_UART_Receive(&huart1,&receive_c, sizeof(receive_c),50);
HAL_UART_Receive(&huart2,&receive_c2, sizeof(receive_c),50);
if (receive_c == 'd' || receive_c2 == 'd' )           //s
{
  HAL_UART_Transmit(&huart1,date_time, sizeof(date_time),500)
  receive_c = '\0';
}
    else if (receive_c == 't' || receive_c2 == 't')       //t
{
  HAL_GPIO_TogglePin (GPIOB, GPIO_PIN_12);
  HAL_Delay(1000);
}

else if (receive_c == 'n' || receive_c2 == 'n')       //turn
{
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_12,0);
  receive_c = '\0';
}
    else if (receive_c == 'f' || receive_c2 == 'f')       //t
{
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_12,1);
  receive_c = '\0';
}
```

Arduino Code

This part creates WiFi network by setting:

- the SSID and its password.
- IP address, Subnet, and gateway.



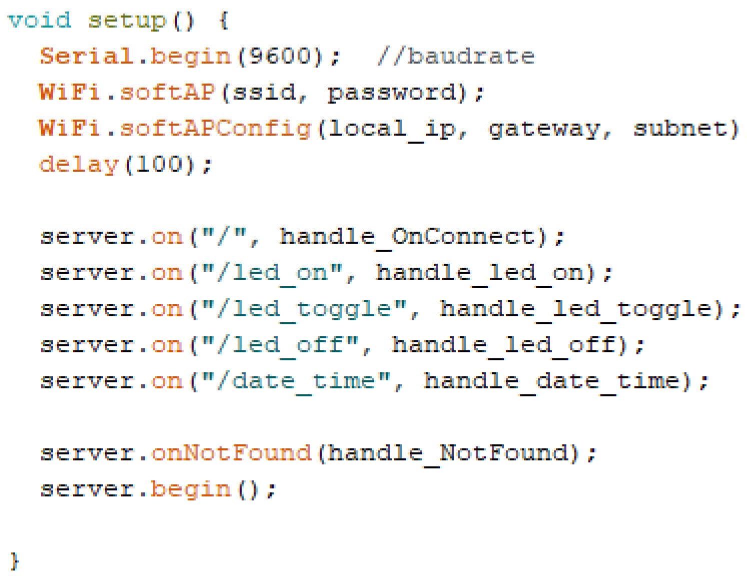
```
/* Put your SSID & Password */
const char* ssid = "NodeMCU";  // Enter SSID here
const char* password = "12345678";  //Enter Password here

/* Put IP Address details */
IPAddress local_ip(192,168,1,1);
IPAddress gateway(192,168,1,1); //each router has gateway, make devoce connect to router
IPAddress subnet(255,255,255,0); //masking (up to 255 decice can connect to the router)
ESP8266WebServer server(80);  //default port number
```

- Set Baud rate to 9600

- Set up a soft access point

- Link the functions to the URLs to send the command according to the button pressed.



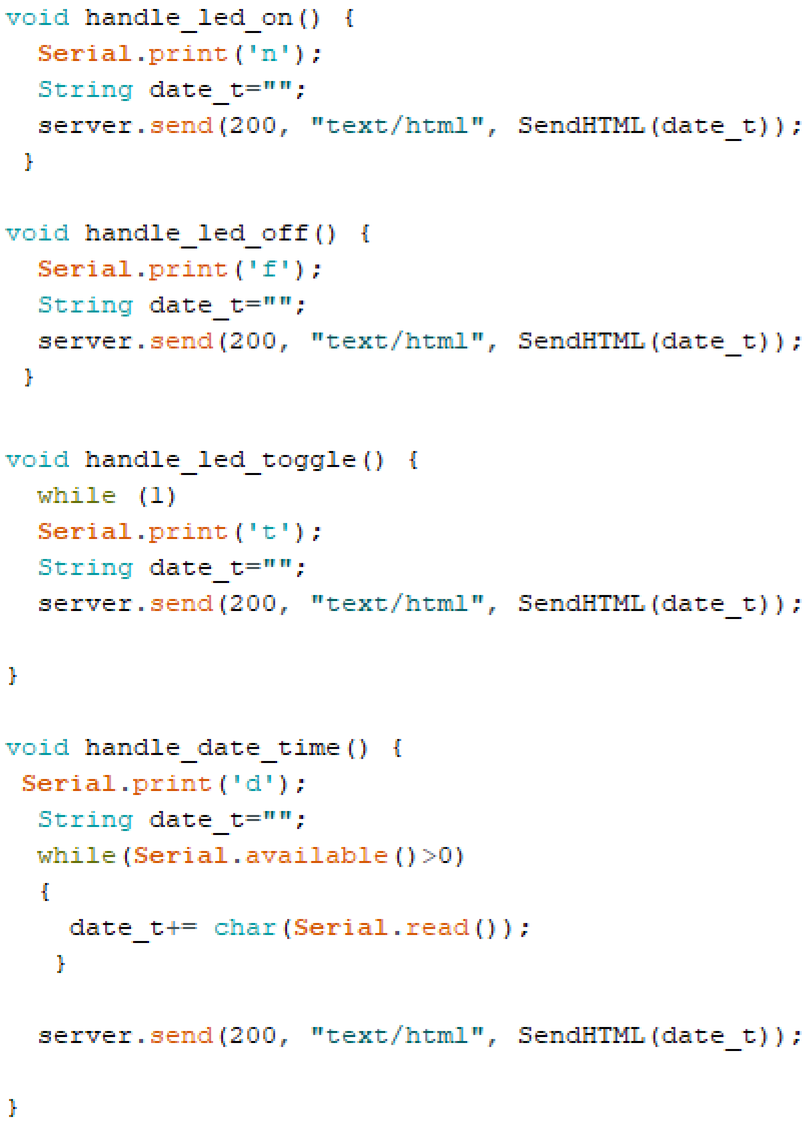
```
void setup() {
  Serial.begin(9600);  //baudrate
  WiFi.softAP(ssid, password);
  WiFi.softAPConfig(local_ip, gateway, subnet)
  delay(100);

  server.on("/", handle_OnConnect);
  server.on("/led_on", handle_led_on);
  server.on("/led_toggle", handle_led_toggle);
  server.on("/led_off", handle_led_off);
  server.on("/date_time", handle_date_time);

  server.onNotFound(handle_NotFound);
  server.begin();

}
```

- Functions used to send the commands to the STM32 using serial communication



```
void handle_led_on() {
  Serial.print('n');
  String date_t="";
  server.send(200, "text/html", SendHTML(date_t));
 }

void handle_led_off() {
  Serial.print('f');
  String date_t="";
  server.send(200, "text/html", SendHTML(date_t));
 }


void handle_led_toggle() {
  while (1)
  Serial.print('t');
  String date_t="";
  server.send(200, "text/html", SendHTML(date_t));

}

void handle_date_time() {
 Serial.print('d');
  String date_t="";
  while(Serial.available()>0)
  {
    date_t+= char(Serial.read());
   }

  server.send(200, "text/html", SendHTML(date_t));

}
```

References

RTC datasheet:

https://blackboard.aucegypt.edu/bbcswebdav/pid-1577849-dt-content-rid-11605847_1/courses/CSCE430201_2020Sp/DS3231%20datasheet.pdf

STM32F103C8:

https://blackboard.aucegypt.edu/bbcswebdav/pid-1558186-dt-content-rid-11337582_1/courses/CSCE430201_2020Sp/Description%20of%20STM32F1%20HAL%20and%20low-layer%20drivers%20%28en.DM00154093%29.pdf

ESP8266 Packages:

https://arduino-esp8266.readthedocs.io/en/latest/installing.html#instructions-windows-10

https://lastminuteengineers.com/creating-esp8266-web-server-arduino-ide/