Documentation

Sadeep Ariyarathna
sadeepari@gmail.com

Table of Contents

1. Wiring Diagram

SIM7600
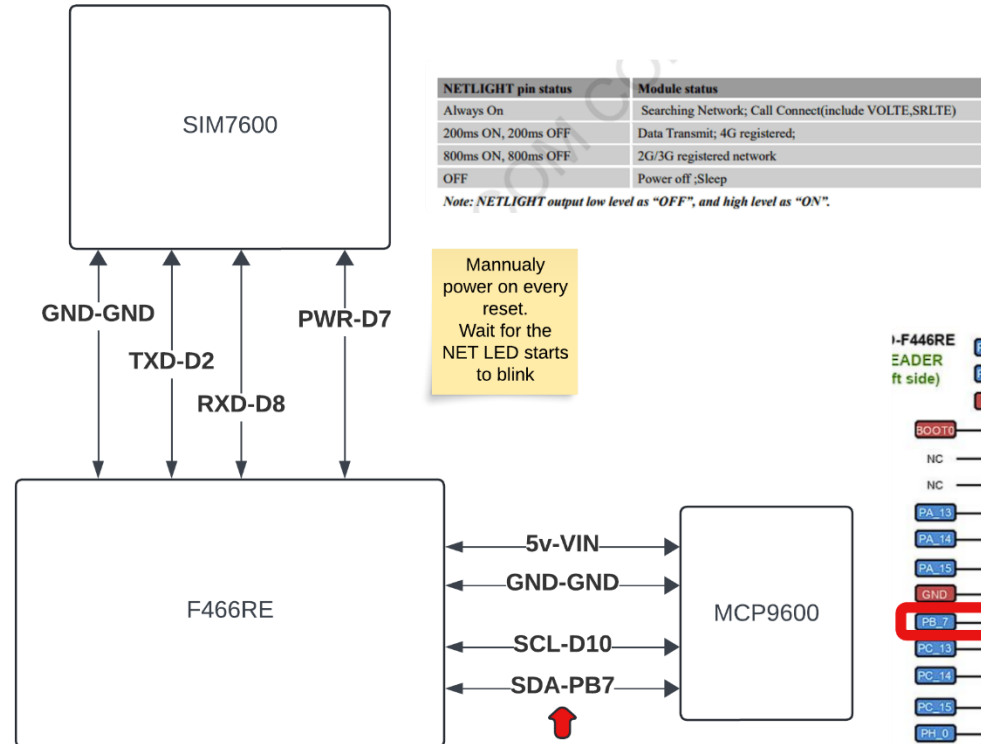
NETLIGHT pin status	Module status
Always On	Searching Network; Call Connect(include VOLTE,SRLTE)
200ms ON, 200ms OFF	Data Transmit; 4G registered;
800ms ON, 800ms OFF	2G/3G registered network
OFF	Power off ;Sleep

Note: NETLIGHT output low level as "OFF", and high level as "ON".

GND-GND   PWR-D7

TXD-D2

RXD-D8

Mannualy power on every reset.
Wait for the NET LED starts to blink

F466RE

5v-VIN

GND-GND

SCL-D10

SDA-PB7

MCP9600

refer to fig.1

I2C connection. Check the address configuration on the circuit

Address	J1	J2
0x67	-	-
0x66	X	-
0x65	-	X
0x64	X	X
0x60	(ADDR to GND)	

NUCLEO-F446RE
MORPHO HEADER CN7 (left side)

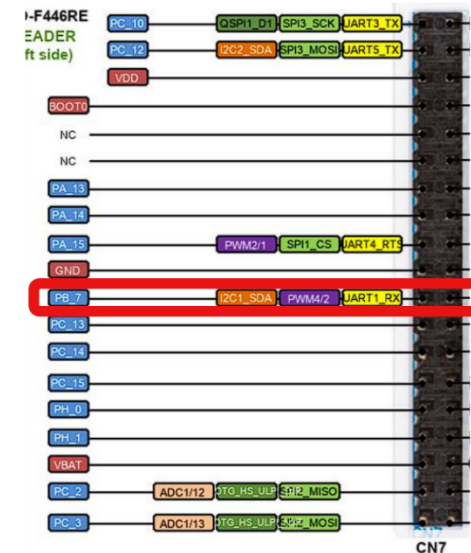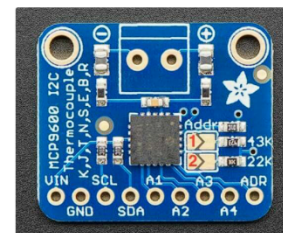| PC_10 | QSPI1_D1 | SPI3_SCK | UART3_TX |
PC_12	I2C2_SDA	SPI3_MOSI	UART5_TX
VDD			
BOOT0			
NC			
NC			
PA_13			
PA_14			
PA_15	PWM2/1	SPI1_CS	UART4_RTS
GND			
PB_7	I2C1_SDA	PWM4/2	UART1_RX
PC_13			
PC_14			
PC_15			
PH_0			
PH_1			
VBAT			
PC_2	ADC1/12	OTG_HS_ULPI	SPI2_MISO
PC_3	ADC1/13	OTG_HS_ULPI	SPI2_MOSI

CN7

fig.1

2. Description

The program is capable of collecting data from MCP9600 sensor and send that data over MQTT with the help of SIM7600 sim module. Data processing and sending is managed by utilizing DMA within the F446RE. Temperature is collected according to the pre-set timer function. Measuring is switched to a faster frequency whenever the measured temperature is above a given threshold.

3. Variables

- **MQTT Variables:** Stores configuration for the MQTT broker like APN, host, port, topics, and connection status flags.
- **SIM7600 Variables:** Stores AT command strings and response buffers.
- **Temperature Reading Variables:** Holds the I2C address of the temperature sensor and stores temperature readings.
- **I2C Callback Variables:** Manages data buffers for handling temperature data.

4. Core Functions

- **powerOn():** Powers on the SIM7600 module.
- **SIMTransmit(char cmd):** Sends AT commands to the SIM7600 module and captures the response.
- **startMQTT():** Initializes and starts the MQTT connection with the broker. It first checks if the module is ready by sending AT commands, then checks for network registration (AT+CGREG?), and finally starts the MQTT service.
- **transmitMQTT():** Publishes temperature data to the MQTT broker. It formats the data as a JSON-like string.
- **endMQTT():** (Not used yet) It is meant to properly disconnect and release the MQTT connection.
- **mainMQTT():** Manages the main MQTT loop, checking and maintaining the connection, and transmitting data.
- **readTemperature():** Reads temperature data from the sensor over I2C and stores it in a buffer.
- **HAL_I2C_MasterRxCpltCallback():** Callback function executed upon I2C data reception. Converts raw temperature data to a usable format and manages data buffers.
- **HAL_TIM_PeriodElapsedCallback():** Handles timer interrupts for periodic temperature data acquisition and manages the trigger for faster data collection if a temperature threshold is crossed.
- **printBuffer():** (For debugging) Prints the contents of the temperature data buffer.
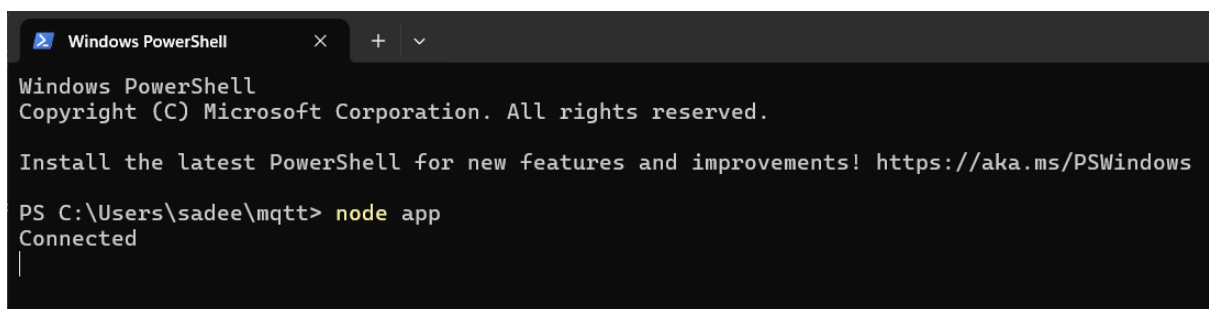
5. Program Flow

- **Main Function:**
    - Initializes the system and peripherals.
    - Starts a timer for periodic interrupts.
    - Continuously runs the mainMQTT() loop to maintain and transmit data over MQTT.

- Before running the C program, the python script should be run. (see next chapter)
- **Temperature Monitoring:**
  - Periodically (based on timer interrupts) reads the temperature.
  - If the temperature crosses a threshold, it triggers a faster sampling rate and expands the buffer size.
  - Data is transmitted via MQTT if the connection is established.

6. MQTT Python Script

- Open CMD at the directory.
- Run the command "node app".
- Wait for the confirmation "Connected".

After seeing "Connected", run the C program to start the Data transfer.