DOCUMENTATION
(DHT11(Temperature & Humidity))

INDEX:

1.Introduction : DHT11 Temperature & Humidity Sensor features a temperature & humidity sensor complex with a calibrated digital signal output. By using the exclusive digital-signal-acquisition technique and temperature & humidity sensing technology, it ensures high reliability and excellent long-term stability. This sensor includes a resistive type humidity measurement component and an NTC temperature measurement component, and connects to a high-performance 8-bit microcontroller.

DHT11 module detailed documentation

1.0. Module Overview:

Each DHT11 element is strictly calibrated in the laboratory that is extremely accurate on humidity calibration. The calibration coefficients are stored as programmes in the OTP memory,which are used by the sensor's internal signal detecting process.

The single-wire serial interface makes system integration quick and easy. Its small size, low power consumption and up-to-20 meter signal transmission making it the best choice for various applications, including those most demanding ones. The component is 4-pin single row pin package. It is convenient to connect and special packages can be provided according to users' request.

1.1. Key Features:

1. **Single-Wire Digital Interface :** The sensor communicates with a microcontroller or other devices through a single-wire digital interface, making it easy to integrate into electronic projects.
2. **Temperature Range:** It can measure temperature in the range of 0°C to 50°C with a resolution of 1°C.
3. **Digital Output:** The sensor provides digital output, which simplifies the interface with microcontrollers. The data is transmitted in the form of a 40-bit digital signal that includes both temperature and humidity information.

Usage :

1. **Weather Stations:** The DHT11 sensor is often used in DIY weather stations to measure and display real-time temperature and humidity data.
2. **Education:** The DHT11 is often used in educational settings to teach students about sensor technology, data acquisition, and basic programming.
3. **Greenhouses:** In agriculture, the sensor can be utilized in greenhouses to monitor and control the temperature and humidity for optimal plant growth.

1.2.Connection and Integration:

1. **Microcontroller Compatibility:**-The module can be interfaced with microcontrollers like Arduino or Raspberry Pi to control the lighting effects programmatically.
2. **Power Requirements:-**Follow the provided documentation to understand the power requirements and connect the module to a suitable power source.
3. **Input Pins:-**Depending on the module, there may be input pins that allow external control or interaction with the lighting patterns.

1.3. Module Features :

- Product Type : DHT11 Temperature & Humidity sensor module
- Product Model : DHT11
- Product Shape : Compact rectangle design
- Humidity Measurement Range: 20% to 80%
- Temperature Measurement Range: 0°C to 50°C
- Product standared Forward voltage : 3.0 to 5.0 V

1.5.Module connection:

To use the DHT11 module,connect it to your microcontroller or development board as follows:

- Connect the VCC pin to the 5V output of your microcontroller or a separate 5v power source.
- Connect GND pin to ground of your microcontroller.
- Connect data pin to your GPIO pins of your microcontroller.

1.6. How to use DHT11:

The DHT11 module is designed to measure temperature and humidity in the surrounding environment. It incorporates the DHT11 sensor, which is a basic, low-cost digital sensor capable of providing accurate readings for these two parameters. To use the DHT11 module, you will typically need a microcontroller like micro controller or another compatible platform.
Here are the general steps to use the DHT11:

Connections: First, connect the DHT11 module to your microcontroller. The exact pinout may vary slightly depending on the manufacturer, so consult the module's documentation or

any markings on the PCB for the pin configuration. Typically, you will need to connect power (VCC and GND) and the data pin to the appropriate pins on your microcontroller.
Power Supply: Ensure you provide the correct voltage (usually 5V) to the VCC pin and connect the GND pin to the ground of your microcontroller or power supply.
Program Your Microcontroller: Write a program for your microcontroller (e.g., Arduino) to control the DHT11 module. Depending on the module's design.This code will vary depending on the specific module you have, so consult the module's documentation or the DHT11's datasheet for guidance.
Upload Code: Upload your microcontroller code to the board. Make sure your microcontroller is properly connected to your computer, and use the appropriate development environment (e.g., Arduino IDE) to upload the code.
Power On: Power on your microcontroller and the DHT11 module.The Temperature and humidity should start getting values or following the programmed pattern.

2. WE10 CELLIUM WIFI MODULE:

2.0.Overview:

- WE-XX series of WiFi modules are controlled by a simple,intutive serial ASCII command interface.There are three different types of data sets that are exchanged between the module and host controller,Commands, Responses and Events.
- Wifi module is used to send and recieve data over Wi-Fi. They can also accept commands over the Wi-Fi.Wi-Fi modules are used for communications bewtween devices.They are most commonly used in the field of Internet of Things.

2.1.Features of wifi module:

- 100mW transmit power
- 11Mbps data rate
- 802.11 b/g/n compatibility
- Integrated antenna

2.2.Commands:

- These are used to command the WiFi module to perform a particular function. All the commands follow the structure described.

- CMD+<command>=<parameter 1>,<parameter 2><CR><LF>
- CMD – ‘Op code’ to identify the data stream as a ‘Command’.
- <command> - Name of command @refer command list (table x)
- <parameters> - Parameters expected by the command @refer command list (table x)
- <CR><LF> - Every command ends with “<CR><LF>” characters.
- There are 2 types of commands; SET commands and GET commands.
- (a) SET commands: These commands are used to set a parameters for a particular function.
- eg. “CMD+NAME=test123<CR><LF>”. This command sets the name of the device to ‘test123’.
- (b) GET commands: These commands are used to get default/stored parameters from the module.
- eg. “CMD?NAME<CR><LF>” This command fetches the name of the module.WE-XX modules respond to all the commands with a status code.
- Total length of any ATCMD should not exceed 1600 bytes for WE10.
- Total length of any ATCMD should not exceed 3200 bytes for WE20D.

2.3. Response

- WE-XX modules send out data packets in response to commands sent by the host controller. All the reponses follow the structure described.
- RSP=<status>,<paramter 1>,<parameter 2><CR><LF>
- RSP – Op code to identify the data stream as a ‘Response’.

- <status> - Execution status of a particular command that was sent by the host controller. <parameters> - Any parameters requested by the host controller @refer command list (table x).
- <CR><LF> - Every response ends with "<CR><LF>" characters.

2.4. Events

- WE-XX module generate events to notify the host controller of special conditions like "peer connection" , "disconnection", "data reception" etc. All the events follow the structure described.
- EVT+<event>=<parameter 1>,<parameter 2><CR><LF>
- EVT – Op Code to identify the data stream as an 'Event'
- <event> - Name of the event @refer event list (table x)
- <parameters> - Any parameters that would be needed by the host controller @refer event list (table x)
- <CR><LF> - Every event ends with "<CR><LF>" characters.
- © 2021, Celium Devices Private Limited2.
- Default UART Settings
- Baud rate 38400
- HWFC Disabled
- Parity None
- Data bits 8
- Stop bits 1

2.5. System commends :

- CMD+RESET command to soft reset module
- CMD+FACRESET Command to reset factory data, uart, ota and fast connect details set to default.
- CMD+UARTCONF Command to configure UART.
- CMD+GPIO Command to control GPIO
- WIFI b/g/n Commands:
- CMD+WIFIMODE Command to set the WiFi mode
- CMD+CONTOAP Command to connect to an Access Point (AP)
- CMD+DISCONN Command to disconnect from the AP
- CMD+SCANAP Command to scan for nearby Access Points (Aps)
- CMD+SETAP Command to configure AP
- CMD+DHCP Command to set IP Type (DHCP or Static IP)

- CMD+STAIP Command to set static ip in STA mode
- CMD+GATEWAYIP Command to set the Gateway IP in AP mode
- CMD+AUTOCONAP Command to enable auto connection.
- CMD+MAC Command to set the mac address of the device.

2.6.TCP/UDP/SSL commands:

- CMD+SETSERVER Command to set TCP/UDP/SSL server.
- CMD+SETCLIENT Command to set TCP/UPD/SSL client.
- CMD+CLOSESKT Command to close TCP/UDP/SSL connection.
- CMD+SKTSENDDATA Command to send data via TCP/UDP/SSL connections.
- CMD+SKTRCVDATA Command to get received data from TCP/UDP/SSL connections
- CMD+SKTAUTORCV Command to enable/disable auto receive data from SKT connection.

2.7. HTTP Commands:

- CMD+HTTP Command to push data to a web server using HTTP/HTTPs protocol
- CMD+HTTPDATA Command to make multiple POST and PUT requests in single session.
- CMD+HTTPCLOSE Command to close current HTTP session.

3.MQTT:

3.0. OVERVIEW:

MQTT, which stands for Message Queuing Telemetry Transport, is a lightweight and open messaging protocol designed for small sensors and mobile devices with high-latency or unreliable networks. It is a publish/subscribe messaging transport protocol that is well-suited for the Internet of Things (IoT) and other scenarios where low bandwidth, high latency, or an unreliable network connection is expected. MQTT was invented by Dr. Andy Stanford-Clark of IBM and Arlen Nipper of Arcom (now Eurotech) in the late 1990s.

3.1. Here are some key features and concepts associated with MQTT:

1. **Publish/Subscribe Model**: MQTT uses a publish/subscribe model where clients can publish messages to a topic or subscribe to receive messages on a specific topic. This decouples the senders (publishers) of messages from the receivers (subscribers), allowing for a flexible and scalable communication pattern.
2. **Broker:**The MQTT broker is a server that acts as an intermediary between publishers and subscribers. It is responsible for receiving messages from publishers and delivering them to subscribers based on the topics they have subscribed to. The broker plays a central role in the MQTT architecture.
3. **Topics:** Topics are hierarchical strings used to categorize and route messages. Publishers publish messages to specific topics, and subscribers express interest in receiving messages from specific topics. Topics allow for a flexible and organized way of routing messages within the MQTT system.
4. **QoS (Quality of Service) Levels:** MQTT supports different levels of message delivery assurance through its Quality of Service levels. There are three QoS levels:

1. QoS 0: The message is delivered at most once, and delivery is not confirmed.

2. QoS 1: The message is delivered at least once, and delivery is confirmed.

3. QoS 2: The message is delivered exactly once by using a four-step handshake.

5. Retained Messages: Retained messages are special messages that the broker retains evenafter sending them to all current subscribers. When a new subscriber subscribes to a topic with a retained message, it immediately receives the last retained message for that topic.

6. Lightweight Protocol:MQTT is designed to be lightweight both in terms of the protocol overhead and the resources required for implementation. This makes it well-suited for resource-constrained devices and networks.

7. Security: While MQTT itself does not define security mechanisms, it can be used over secure transport layers such as TLS/SSL to provide encryption and authentication.Additionally, some implementations support username/password authentication.

8. Persistent Sessions: Clients can establish persistent sessions with the broker, allowing them to resume communication after a disconnect. This is useful for maintaining state and ensuring message delivery in intermittent network conditions.

MQTT has become a popular choice for IoT applications, home automation, and other scenarios where efficient, low-overhead communication is essential. It has a wide range of client libraries available for different programming languages and is supported by many IoT platforms and devices.

3.2.Commands :-

1. CMD+MQTTNETCFG= <host add>,<port> <CR> <LF> command to set MQTT network configuration .
2. CMD+MQTTCONCFG :- command to set MQTT connection configuration.
3. CMD+MQTTSTART = <param> <CR> <LF> :- command to start /stop to mqttoperation.
4. CMD+MQTTSUB = <sub_topic> <CR> <LF> command to subscription is required.
5. CMD+MQTTCONCFG Command to set MQTT connection configuration.
6. CMD+MQTTNETCFG command to set MQTT network configuration.
7. CMD+MQTTPUB Command to publish data to broker.
8. CMD+MQTTSUB Command to subscribe to MQTT topic.
9. CMD+MQTTUNSUB Command to unsubscribe to mqtt topic.
10. CMD+MQTTSTART Command to start mqtt communication.
11. CMD+MQTTSSL Command to set MQTT SSL and CRT conif.

4.Righttec iot:

4.0.Overview :

Rightech IoT Cloud is an end-to-end real-time cloud IoT platform that connects any device with any protocol, provides data normalisation, web-based SCADA-layer for complex automation scenarios without writing code, visually designed logic patterns utilise real and virtual events and are automatically transpilated into code, complex m2m communication via finite automatons, not requiring any coding or low level device programming, allowing even non-IoT folk like web and mobile developers or corporate IT depts to wrap up the coolest IoT cases.

Rightech is a full-scale end-to-end cloud platform for the Internet of Things. Our platform is aimed to make IoT technology affordable to everyone and allow even non-IoT people like web and mobile developers or corporate IT departments to wrap up the coolest IoT cases.

We are achieving this goal by virtue of unique platform's features:

- Any device with any protocol can be used.
- Old devices even without internet capabilities can be connected.
- Visual logic editor that anyone is fit to use to automate business-processes without writing any code .
- Data normalisation based on virtual models allows to design single uniform automation scenarios including complex m2m communication.

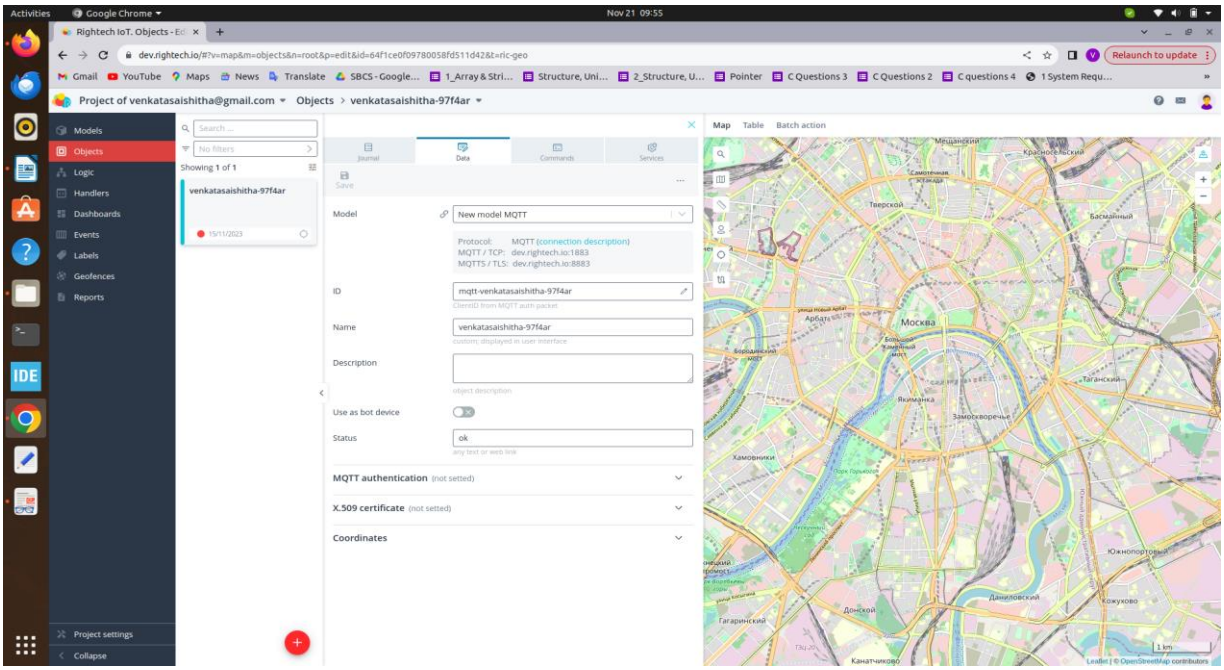
As a result we're drastically bringing down the cost and integration time frame of any IoT project giving our clients some crucial advantages:

- All client's resources can be dedicated to end applications generating the actual value
- No need for expensive programmers with rare IoT skills
- No low level device programming
- Various dashboards and predictive/real-time analytic tools already available
- Integration takes from 3 to 5 times less time compared to existing solutions
- Cost is determined only by the number of connected devices and is fully transparent.

Rightech is your universal framework for IoT applications development.Forget about device connectivity problems and focus on your applications that generate value.

4.1 HOW TO REGISTER IN RIGHTETC:

- **Registration:** Look for a "Register" or "Sign Up" option on their website. Click on it to start the registration process.
- **Fill Out Registration Form:** Provide the required information, such as your name, email address, password, and any other details that the platform requests.
- **Account Verification:** Some platforms may require you to verify your email address by clicking on a verification link sent to your email. Complete the verification process if required.
- **Log In**: Once your registration is complete and your account is verified, log into your "RightTech` IoT" account using your credentials.
- **Explore the Platform:** Navigate through the platform to understand its features, dashboard and settings. You should look for an option related to creating or managing parameters,which are typically settings or values used to configure and control IoT devices or data.



5.STM32 Microcontroller :

5.0.overview

The STM32 microcontroller series is a family of 32-bit ARM Cortex-M based microcontrollers developed by STMicroelectronics. These microcontrollers are widely used in various embedded systems, ranging from simple applications to complex projects in industries like automotive,industrial automation, consumer electronics, and more.

5.1. Here are some key features and aspects of STM32 microcontrollers:

- **ARM Cortex-M Architecture:**STM32 microcontrollers are built around the ARM Cortex-M processor architecture. The Cortex-M processors are known

for their low power consumption, high performance, and efficient architecture, making them well-suited for embedded applications.

- **Variety of Series**: The STM32 family is quite extensive and is categorized into different series based on their features and capabilities. Common series include STM32F0, STM32F1,STM32F3, STM32F4, STM32F7 and STM32H7among others.
- **Peripheral Integration:**STM32 microcontrollers come with a rich set of peripherals, such as GPIO (General Purpose Input/Output), USART, SPI, I2C, ADC (Analog-to-Digital Converter),timers, PWM controllers, and more. These peripherals provide a wide range of capabilities for interfacing with various sensors, actuators, and communication devices.
- **Development Tools:** STMicroelectronics provides a comprehensive set of development tools for STM32, including the STM32CubeMX configuration tool and the STM32CubeIDE integrated development environment. These tools simplify the process of configuring peripherals, generating initialization code, and developing applications.
- **Hardware Abstraction Layer(HAL) and Low-Level (LL) Libraries:**STM32 microcontrollers are supported by HAL and LL libraries, which provide a standardized way to access the hardware features. The HAL provides a high-level API, while the LL library offers a lower-level, more direct access to the hardware.
- **RTOS Support**:STM32 microcontrollers are often used in applications that require real-time operating systems (RTOS). The ARM Cortex-M architecture supports various RTOS implementations, and STMicroelectronics provides support for popular RTOS like Free RTOS.
- **Energy Efficiency:** Many STM32 microcontrollers are designed with a focus on low power consumption. This makes them suitable for battery-powered or energy-efficient applications.
- **Community and Ecosystem:** The STM32 community is active, and there are various forums,documentation, and resources available for developers. This community support can be valuable when working on STM32-based projects.

Overall, STM32 microcontrollers are widely used in the embedded systems industry due to their performance, flexibility, and the extensive ecosystem surrounding them. Developers can choose from a wide range of STM32 microcontrollers based on the specific requirements of their projects.

5.3. STM32 IDE tool:

Overview:

1. Integrated Development Environment (IDE):

- The IDE is the primary interface for software development. It includes code editing, debugging, and project management tools.

2. STM32CubeMX:

- STM32CubeMX is a graphical configuration tool that allows you to initialize the peripherals and configure the pin assignments for your STM32 microcontroller.

3. HAL (Hardware Abstraction Layer) Library:

- STM32CubeIDE integrates the HAL library, which provides a set of high-level functions to interact with the microcontroller peripherals.

4. CMSIS (Cortex Microcontroller Software Interface Standard):

- CMSIS provides a standardized interface to the core functions of a Cortex-M microcontroller, including the NVIC (Nested Vector Interrupt Controller) and SysTick.

5 .Debugger:

- The debugger allows you to set breakpoints, inspect variables, and step through code during the debugging process.

6. Project Configuration:

- You can configure various project settings, including compiler options,linker scripts, and build configurations.

7. Console Output:

- The console provides feedback from the compiler and debugger,displaying messages, warnings, and errors.

8. Peripheral Configuration:

- STM32CubeMX generates initialization code based on your peripheral configurations, helping to set up the microcontroller's features.

9. Project Explorer:

- The Project Explorer organizes your project files and allows you to navigate through the source code and configuration files.

10 Workflow:

Project Initialization:

- Create a new project in STM32CubeIDE.
- Configure the microcontroller and peripherals using STM32CubeMX.Code

Development:

- Write and edit your C code in the IDE.

Build and Compilation:

- Compile your code to generate the binary file.

Debugging:

- Use the debugger to find and fix issues in your code.

Flash and Run:

- Flash the compiled code onto the STM32 microcontroller and run the application.

5.4. MINICOM :

Minicom is a text-based serial communication program that is commonly used to connect to and communicate with devices over a serial port. It is often used 15for debugging and configuring devices, especially in embedded systems and projects involving microcontrollers or other hardware components. Below is an explanation of how minicom can be used in a project:

- Before using minicom, you need to install it on your system. You can typically install it using your system's package manager. For example, on a Debian-based system, you can use:

- bash
- sudo apt-get install minicom

2. Connecting to a Serial Port:

- Minicom is primarily used for serial communication, so you need to connect it to the serial port of the device you want to communicate with. Use the following command to open minicom:

1. bash

2.minicom -D /dev/ttyUSB0

3. Here, /dev/ttyUSB0 is the path to the serial port. The actual port may vary depending on your system and the connected device.

3. Configuration:

- Once minicom is open, you may need to configure the serial port settings such as baud rate, data bits, stop bits, and parity. This is often necessary to match the settings of the device you are communicating with. You can access the configuration menu by pressing Ctrl-A followed by Z.

4. Interacting with the Device:

- After configuring the serial port, you can interact with the device.Minicom allows you to send commands and receive responses. This is particularly useful for debugging purposes and for configuring devices that have a serial console.

5. Exiting Minicom:

- To exit minicom, you can use the Ctrl-A followed by X shortcut.

6. File Transfer:

- Minicom also supports file transfer using protocols like Xmodem or Ymodem. This can be useful for updating firmware or transferring files between your computer and the connected device.

7.Connect to Device:

- Open minicom and connect to the serial port of your device.
- Configure Settings:
- Configure the serial port settings to match the requirements of your device.

8. Interact and Debug:

- Send commands and receive responses for debugging or configuring the device.

9. File Transfer (if needed):

- Use minicom's file transfer capabilities if you need to transfer files between your computer and the device.

10. Exit Minicom:

- When you are done, exit minicom.

6.Rugged Board :

6.0.overview

Rugged Board is an Open Source Hardware & Software initiative to align with the fast-growing Semiconductor technologies with a switch from classic to modern product development strategy & process.

7.Stage 1 : Connection between stm32 and sensor module:

To interface the KY-034 module with an STM32F446RE board, you'll need to control the LEDs using the STM32 microcontroller's GPIO (General Purpose Input/Output) pins.

7.0.Required parameters:

1.Microcontroller
2.DHT11
3. Jumpers

7.1.Connection from sensor to STM32 :

1. Data -> PA1 pin (any gpio pins use)
2. VCC - > +5v.
3. GND -> GND.

7.2.Sample code for stm to sensor:

```
while(1)
  {
                        char buffer[64];
                              DHT11_Start();
                              Presence = DHT11_Check_Response();
                              Rh_byte1 = DHT11_Read ();
                              Rh_byte2 = DHT11_Read ();
                              Temp_byte1 = DHT11_Read ();
                              Temp_byte2 = DHT11_Read ();

                              TEMP = Temp_byte1;
                              RH = Rh_byte1;

                              Temperature = (int) TEMP;
                              Humidity = (int) RH;

                               snprintf(buffer, sizeof(buffer), "Temperature: %d°C, Humidity: %d%%\r\n", Temperature,
Humidity);

                                 HAL_UART_Transmit(&huart2, (uint8_t*)buffer,
strlen(buffer), HAL_MAX_DELAY);

                                 HAL_Delay(1000);

                               sprintf(&buffer[0], "%d\r\n", Humidity);
                               HAL_UART_Transmit(&huart1, (uint8_t *)buffer, strlen(buffer), 1000);
                               HAL_Delay(1000);
                               sprintf (&buffer[0],"%d\r\n",Temperature);
                               HAL_Delay(1000);

  }
```

8. Stage 2: Connection between stm32 and sensor module by using wifi module:

To interface the KY-034 module with an STM32F446RE board, you'll need to control the LEDs using the STM32 microcontroller's GPIO (General Purpose Input/Output) pins. transmit the status of led values into rightetch using wifi module.

8.1.Required parameters:

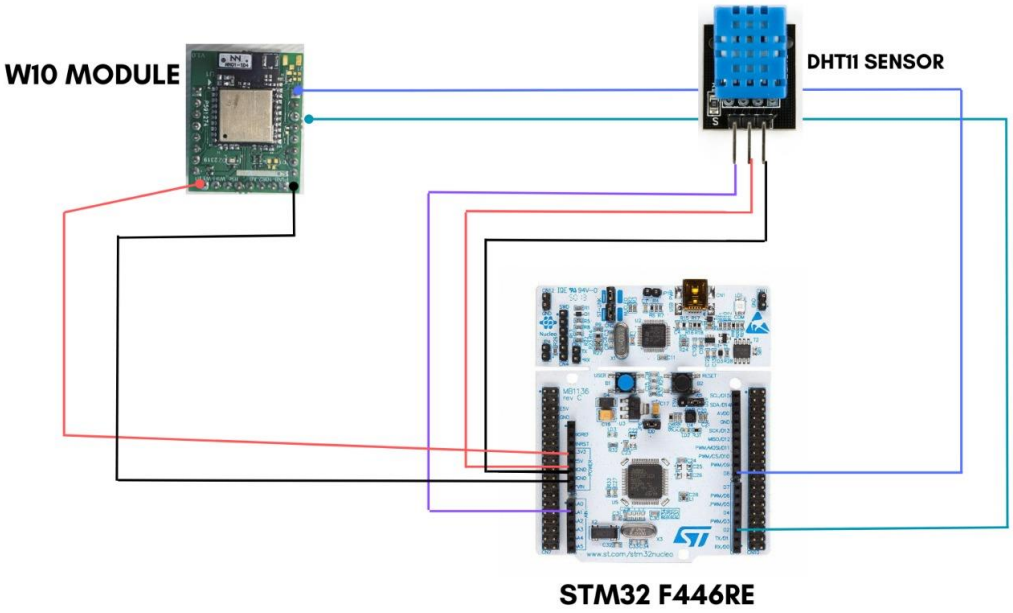
1.microcontroller
2.DHT11 sensor
3. Jumpers.
4. wifi module.

8.2.Connection between stm32 to sensor module:

1.Data -> PA1 pin (any gpio pins use).
2.VCC - > +5v.
3.GND -> GND.

8.3.Connection between stm32 to wifi module :

1.RX -> D8(stm32 uart1 is used to Tx connect to wifi rx)
2.TX -> D9 (stm32 uart1is used to Rx connect to wifi Tx)
3.VCC-> 3V (wifi 3V connect to stm32 3v)
4.GND -> GND(wifi GND connect to stm32 GND)



8.4.Sample code:

```
        while(1)
 {
                char buffer[64];
                        DHT11_Start();
                        Presence = DHT11_Check_Response();
                        Rh_byte1 = DHT11_Read ();
                        Rh_byte2 = DHT11_Read ();
                        Temp_byte1 = DHT11_Read ();
                        Temp_byte2 = DHT11_Read ();

                        TEMP = Temp_byte1;
                        RH = Rh_byte1;

                        Temperature = (int) TEMP;
                        Humidity = (int) RH;

                         snprintf(buffer, sizeof(buffer), "Temperature: %d°C, Humidity: %d%\r\n", Temperature,
Humidity);

                           HAL_UART_Transmit(&huart2, (uint8_t*)buffer,
strlen(buffer), HAL_MAX_DELAY);

                           HAL_UART_Transmit(&huart1, (uint8_t*)buffer, strlen(buffer), HAL_MAX_DELAY);


                           HAL_Delay(1000);

                         sprintf(&buffer[0], "CMD+MQTTPUB=base/state/humidity,%d\r\n", Humidity);
```

```
                    HAL_UART_Transmit(&huart1, (uint8_t *)buffer, strlen(buffer), 1000);
                    HAL_Delay(1000);

                            sprintf
(&buffer[0],"CMD+MQTTPUB=base/state/temperature,%d\r\n",Temperature);
                      HAL_UART_Transmit(&huart1, (uint8_t*)buffer, strlen(buffer),1000);
                      HAL_Delay(1000);

 }
```

9.Stage 3: Steps to connect stm32 to rugged board to control KY-034 flash led:

To interface the KY-034 module with an STM32F446RE board, you'll need to control the LEDs using the STM32 microcontroller's GPIO (General Purpose Input/Output) pins.and send the led status values into rugged board terminal.

9.0. Required parameters:

1. Microcontroller.
2. DHT11.
3. Jumpers.
4. Rugged board

9.1.Connection between stm32 to 7-colour flash:

1.Signal -> PA1 pin (any gpio pins use)
2 .VCC - > +5v.
3.GND -> GND

9.2 Connection between stm32 to rugged board :

1.By using stm32 uart4 the PA9 pin is connected to the rugged board microbus (uart3) RX pin.
2.In rugged board GND pin is connected to stm32 GND.

9.3.Sample code for STM32:

```
while(1)
{
            char buffer[64];
                DHT11_Start();
                Presence = DHT11_Check_Response();
                Rh_byte1 = DHT11_Read ();
                Rh_byte2 = DHT11_Read ();
                Temp_byte1 = DHT11_Read ();
                Temp_byte2 = DHT11_Read ();
                //SUM = DHT11_Read();

                TEMP = Temp_byte1;
```

```
                    RH = Rh_byte1;

                    Temperature = (int) TEMP;
                    Humidity = (int) RH;

                      snprintf(buffer, sizeof(buffer), "Temperature: %d°C, Humidity: %d%\r\n", Temperature,
Humidity);

                      HAL_UART_Transmit(&huart2, (uint8_t*)buffer, strlen(buffer), HAL_MAX_DELAY);

                      HAL_UART_Transmit(&huart1, (uint8_t*)buffer, strlen(buffer), HAL_MAX_DELAY);

                      HAL_Delay(1000);}
```

9.4.Sample code for rugged board:

```
int main() {
        const char *portname = "/dev/ttyS3";  // Replace with the actual UART device file
       int fd = open(portname, O_RDWR | O_NOCTTY | O_SYNC);
        if (fd < 0) {
       error("Error opening UART");}
      struct termios tty;
      if (tcgetattr(fd, &tty) < 0) {
      error("Error from tcgetattr");
      }
cfsetospeed(&tty, B9600);  // Set the baud rate
      cfsetispeed(&tty, B9600);
tty.c_cflag |= (CLOCAL | CREAD);    // Ignore modem control lines, enable receiver
      tty.c_cflag &= ~CSIZE;            // Clear data size bits
      tty.c_cflag |= CS8;               // 8-bit data
      tty.c_cflag &= ~PARENB;           // No parity bit
```

```
        tty.c_cflag &= ~CSTOPB;          // 1 stop bit
        tty.c_cflag &= ~CRTSCTS;         // No hardware flow control
 tty.c_lflag = 0;                // Non-canonical mode
tty.c_cc[VMIN] = 1;              // Minimum number of characters to read
        tty.c_cc[VTIME] = 1;             // Time to wait for data (in tenths of a second)
if (tcsetattr(fd, TCSANOW, &tty) != 0) {
        error("Error from tcsetattr");
        }
char buf[50];
        memset(buf, 0, sizeof(buf));
        int n = read(fd, buf, sizeof(buf));
        if (n < 0) {
        error("Error reading");
        }
printf("Received: %s\n", buf);
close(fd);
        return 0;
}
```

10. Stage 4 :Steps to connect stm32 to rugged board to control DHT11 BY Using righttech:

To interface the KY-034 module with an STM32F446RE board, you'll need to control the LEDs using the STM32 microcontroller's GPIO (General Purpose Input/Output) pins.and

send the led status values into rugged board and transmit the led values into righttech using wifi module.

10.0.Required parameters:

1.Microcontroller.

2.DHT11.

3. Jumpers.

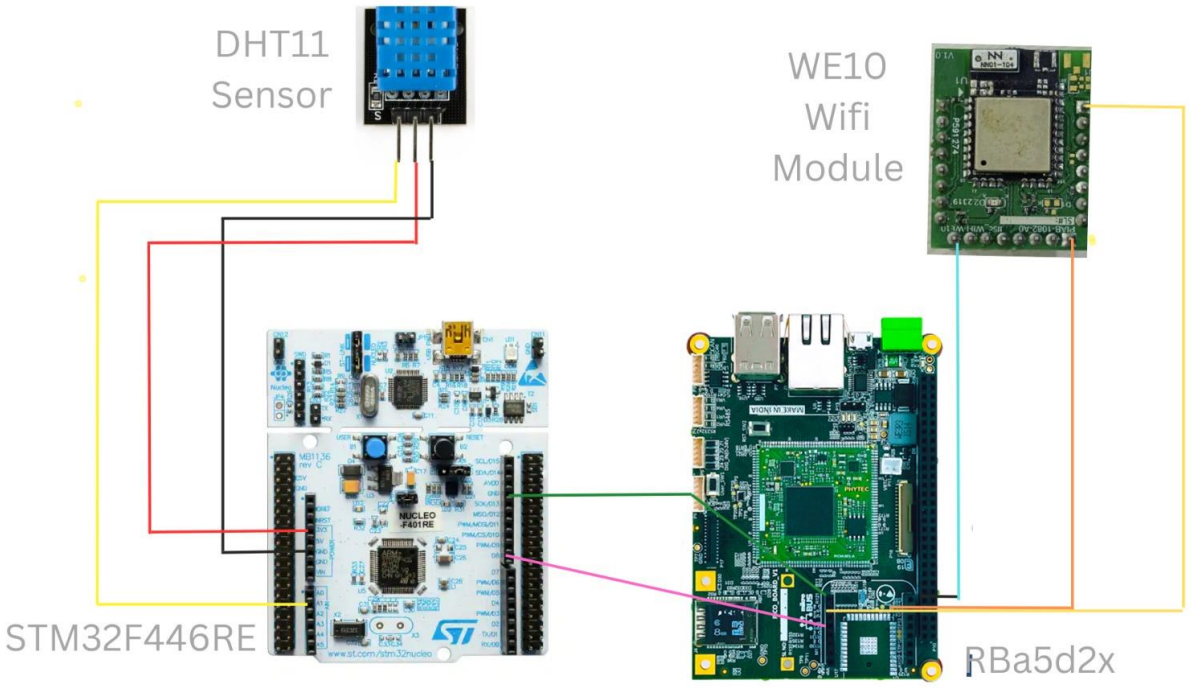
4.Rugged board.

5.WIFI module.

10.1.Identify Pin Assignments:

Connection between microcontroller and DHT11: signal pin connected to themicrocontroller PA1 pin ,and then -VE pin is connected to the GND pin on microcontroller.+VE pin connected to the 5V supply on microcontroller.

Connection between microcontroller and rugged board: In rugged board we have to take one UART for transmit the status of (DHT11) from STM32 to rugged board.we have to take microbus for uart.Connect From stm32 PA1 pin to rugged board receiver pin in micro bus.



10.2.Sample code for stm32 :

```
 while(1)
 {
                    char buffer[64];
                        DHT11_Start();
                        Presence = DHT11_Check_Response();
                        Rh_byte1 = DHT11_Read ();
                        Rh_byte2 = DHT11_Read ();
                        Temp_byte1 = DHT11_Read ();
                        Temp_byte2 = DHT11_Read ();
                        //SUM = DHT11_Read();

                        TEMP = Temp_byte1;
                        RH = Rh_byte1;

                        Temperature = (int) TEMP;
                        Humidity = (int) RH;


                          snprintf(buffer, sizeof(buffer), "Temperature: %d°C, Humidity: %d%\r\n", Temperature,
Humidity);


                          HAL_UART_Transmit(&huart2, (uint8_t*)buffer, strlen(buffer), HAL_MAX_DELAY);

                          HAL_UART_Transmit(&huart1, (uint8_t*)buffer, strlen(buffer), HAL_MAX_DELAY);


                          HAL_Delay(1000);
}
```

10.3.Sample code for rugged board:

```
while(1){
 rdlen = read(fd, buf, sizeof(buf) - 1);
if (rdlen > 0) {
    buf[rdlen] = '\0'; // Null-terminate the received data
        printf("%s\n", buf);
    // Format the data from 'buf' into 'buffer'
    int ret = snprintf(buffer, sizeof(buffer), "CMD+MQTTPUB=base/state/humidity,%s\r\n",
buf);
     int ret = snprintf(buffer, sizeof(buffer),
"CMD+MQTTPUB=base/state/temperature,%s\r\n", buf);
    if (ret < 0) {
        // Handle the error if snprintf fails
```

```
} else {
    // Open the file descriptor 'fd' if not already opened

    // Write the formatted message to the file descriptor
    ssize_t wlen = write(fd, buffer, ret);
    sleep(3);
    if (wlen == -1) {
        // Handle the write error if needed
    }
```

11.Advantages:

1. Easy of use.
2. Affortability
3. Digital Output
4. Low Power Consumption
5. Reliability
6. Availability

12. Disadvantages:

1. Limited Measurement Range.
2. Accurancy.

13. Applications :

- Indore Climate Monitoring.
- Temperature & Humidity Control in Electronics Cabinets.

- Food Storage.
- Educational Projects.
- Consumer Electronics.
- HAVC Systems.
- IOT Devices.
- Automated Plant Care system.

14. Resources

- Module Documentation.
- You Tube Tutorials.
- Online Retailers and Marketplace.
- Manufacturer’s Website.
- Online Electronics Forums.

15. References.

- https://www.community.ruggedboard.com/
- https://rightech.io/en.
- https://www.st.com/en/microcontrollers-microprocessors/stm32-32-bit-arm-cortex-mcus.html
- https://mqtt.org/.