UNIVERSIDADE FEDERAL DO CEARÁ

Projeto Sistemas Microprocessados

Greenhouse Management System

Arnaldo Aguiar - 422578
Nicolas Carvalho - 413327
João Gabriel Soares - 415607

2019.2

INTRODUÇÃO

Nossa ideia teve como objetivo criar um protótipo de um projeto que consiga auxiliar no desenvolvimento de plantas e flores em estufas, promovendo um melhor gerenciamento de fatores como umidade do solo e temperatura do ar no ambiente.

Sendo de grande de serventia para casos em que o cultivo exige determinadas condições ambientais específicas para uma maior produtividade ou análises desejadas.

MATERIAIS

1. Microcontrolador Greenpill (STM32F030F4P6);
2. Módulo Wi-Fi ESP8266-01*;
3. Sensor de Temperatura LM35DZ;
4. Sensor de Umidade do Solo;
5. Programador ST-Link V2 STM8 e STM32 MCU;
6. Protoboard;
7. Jumpers.

*Lembrando que para o correto funcionamento da ESP8266 é necessário a realização de uma atualização de firmware (utilizamos um USB Serial). No caso da ESP8266-01, além da atualização, utilizamos um circuito específico para enviarmos os dados lidos para o servidor do ThingSpeak.

FUNCIONAMENTO

O projeto tem como objetivo simular situações que podem ser encontradas no dia-a-dia e auxiliar os botânicos ou pessoas que são encarregadas de controlar as situações de estufas, que muitas vezes exigem condições específicas de temperatura ambiente e umidade apresentada pelo solo.

Como exemplo, a semente de soja, para a germinação e a emergência da plântula, requer absorção de água de, pelo menos, 50% do seu peso seco. A temperatura média adequada para semeadura da soja vai de 20°C a 30°C, sendo 25°C a ideal para uma emergência rápida e uniforme. Semeadura em solo com temperatura média inferior a 18°C pode resultar em drástica redução nos índices de germinação e de emergência, além de tornar mais lento esse processo. Temperaturas acima de 40°C, também, podem ser prejudiciais.

O microcontrolador gerencia a leitura de dois sensores analógicos, realiza a conversão desses valores, um de cada vez, por meio de canais e os transmite por WiFi para um servidor (o ThingSpeak), que é lido por um aplicativo mobile e mostrado ao usuário de forma interativa.

Lembrando que a atualização dos dados ocorre de acordo com o que é determinado pelo código, ou seja, de acordo com os delays que são implementados.

Figura 1: Estufa.

CUBEMX

Dentro do aplicativo CubeMX configuramos a porta PA2 e PA3 para serem a USART1_TX e a USART1_RX, respectivamente, portas por onde serão enviados os dados para o módulo WIFi, e recebidas as respostas da ESP8266-01, que se comunica com o ThingSpeak.

Para realizar a leitura dos sensores, os dois utilizando o mesmo Conversor AD presente no CortexM0, utilizamos as portas PA0 e PA1 do Microprocessador como ADC_IN0, que recebe a temperatura, e ADC_IN1, que recebe a umidade, respectivamente.

E, por fim, configuramos as portas PA13 e PA14 para o recebimento do clock, SYS_SWCLK, e para o teste no debug, SYS_SWDIO, respectivamente.

A imagem abaixo mostra o modelo esquemático da configuração realizada:

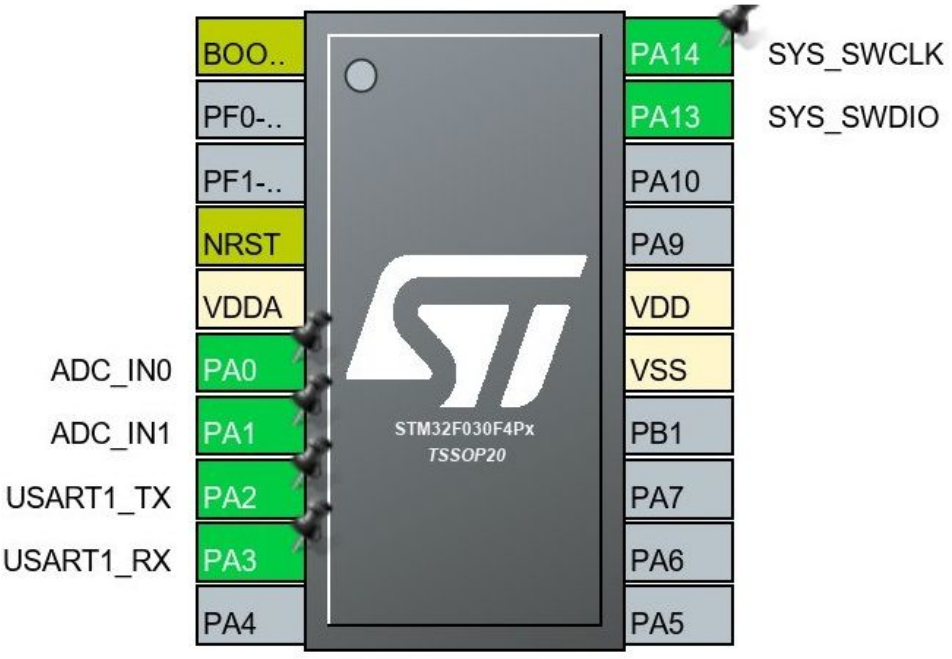


Figura 2: Configuração no CubeMX.

THINGSPEAK

Para realizar os ajustes no ThingSpeak necessitamos, primeiramente, criar uma conta com o email institucional da UFC dentro dos servidores da Universidade, após esse passo, o login pode ser realizado de qualquer local.

Após a criação do canal, é necessário habilitar os Fields que serão utilizados, em nosso caso, o Field1 e o Field2, e copiar a API (na aba API Key você irá observar o local API Requests e copiar o link que fica na box Write a Channel Feed), que irá viabilizar a comunicação do código lido pelo Microcontrolador e a ESP8266 para o envio correto dos dados ao servidor.

Abaixo está uma imagem que indica a forma de como os dados devem ser mostrados no Canal, lembrando que os dados abaixo foram obtidos durante as configurações e ajustes dos sensores, por esse motivo estão destoantes.

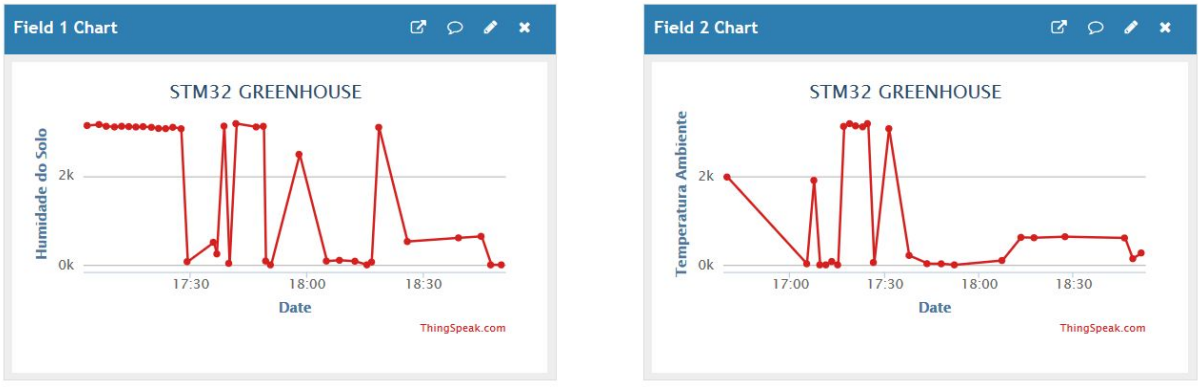


Figura 3: Dados no ThingSpeak.

MONTAGEM E CÓDIGO

Tendo como base a configuração das portas realizadas pelo CubeMX, ficamos com a seguinte configuração física do projeto:

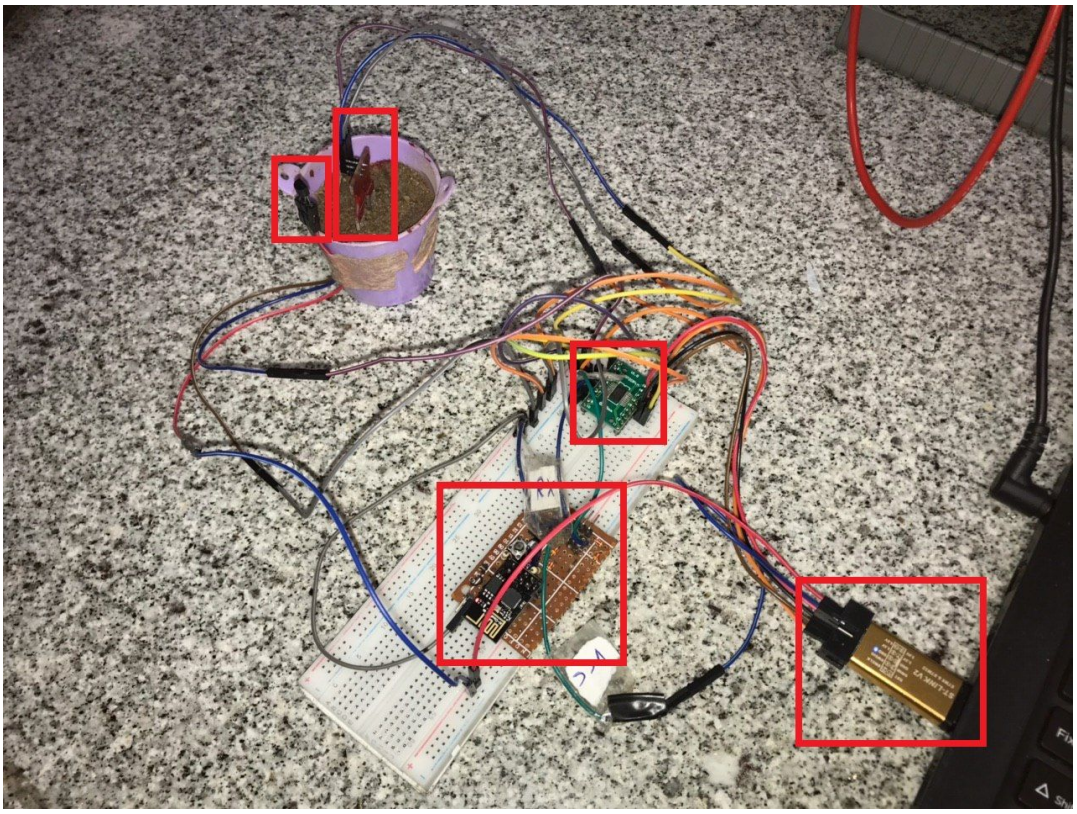


Figura 4: Montagem do circuito.

Durante a execução do código realizamos, de forma resumida, a leitura do dois sensores analógicos e enviamos as informações para os servidores do ThingSpeak por meio da ESP8266-01, que é configurada com a API do servidor criado no site para receber os dados.

Como os sensores são analógicos e passam pelo Conversor AD, necessitamos realizar um tratamento nos valores lidos, com isso, realizamos os seguintes procedimentos para ajustar os valores para serem mostrados em um valor interpretável em porcentagem, lembrando que o STM32F030F4P6 recebe dados de 12 bits, por isso ocorre a divisão por 4095, pois 2^{12} - 1 = 4095. Dessa forma,

parametrizamos o valor para transformá-lo em porcentagem da temperatura e da umidade, respectivamente:

```
valueTemperature = ((valueTemperature*205)/4095);

valueHumitidy=(valueHumitidy*100)/4095;
```

Entretanto, para realizar esses passos necessitamos ultrapassar uma limitação física da Greenpill, que só apresenta um Conversor AD interno. Dessa forma, em nosso projeto foi necessária a criação de dois canais de transmissão de dados, que vão sendo alternados sequencialmente para a leitura e o consequente envio dos respectivos sensores para o servidor. Abaixo está o código em que estabelecemos os dois canais necessários para a leitura:

```
int ler_AD1 (int numero){
    ADC_ChannelConfTypeDef chConfig = {0};
    switch(numero){
    case 0:
              chConfig.Channel = ADC_CHANNEL_0;
              break;
    case 1:
              chConfig.Channel = ADC_CHANNEL_1;
              break;

    }

    HAL_ADC_ConfigChannel(&hadc, &chConfig);
    HAL_Delay(500);
    HAL_ADC_Start(&hadc);
    HAL_Delay(500);
    int v = HAL_ADC_GetValue(&hadc);
    HAL_Delay(500);
    HAL_ADC_Stop(&hadc);
        chConfig.Rank = ADC_RANK_NONE;
        HAL_ADC_ConfigChannel(&hadc, &chConfig);
    return v;
}
```

No while, após as configurações de início presentes, o código realiza a troca de field, do 1 para o 2, realiza a leitura da variável em questão e realiza o seu envio.

A troca o field:

```
strcpy(msg,"GET /update?api_key=B4XQN73HCTMIZTMM&field1=0000\r\n");
```

A leitura do valor do sensor de umidade no canal 1 (que é similar à configuração da leitura da temperatura, sendo realizada a troca do canal de leitura), e por meio dos comandos HAL_UART_Transmit e HAL_UART_Receive os dados são enviados para o servidor (ThingSpeak) e o retorno do envio para o MCU.

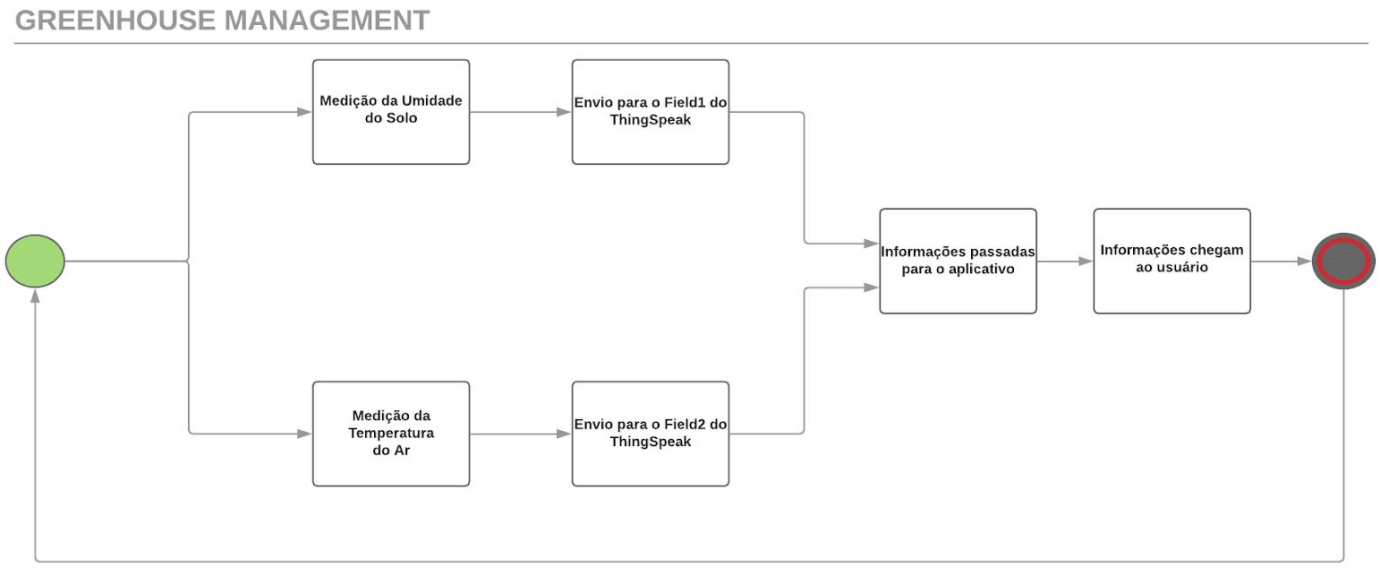
```
        valueHumitidy = ler_AD1((int) 1);
        valueHumitidy=(valueHumitidy*100)/4095;
        int_to_string(valueHumitidy, valueStrH, 4);
        strcopy(msg, valueStrH, 44, 47);

       //Starts the communication
        HAL_UART_Transmit(&huart1,(uint8_t*)"AT+CIPSTART=\"TCP\",\"api.thingspeak.com
\",80\r\n", 70, 2000);
        HAL_UART_Receive(&huart1, buf1, 100, 5000);
        HAL_Delay(1000);

       //Inform the number of bytes will be sended
        HAL_UART_Transmit(&huart1, (uint8_t *)"AT+CIPSEND=50\r\n", 15, 2000);
        HAL_UART_Receive(&huart1, buf1, 100, 5000);
        HAL_Delay(500);

       //Transmit the message to thinkspeak through the ESP
        HAL_UART_Transmit(&huart1, msg, 50, 2000);
        HAL_UART_Receive(&huart1, buf1, 100, 5000);
```

DIAGRAMA DE BLOCOS



GITHUB e THINGSPEAK

Repositório GitHub em que se encontra o código implementado:

- github.com/arnaldoaguiarp/Greenhouse

Canal do ThingSpeak para onde os dados obtidos foram enviados:

- thingspeak.com/channels/919289