Universidade Católica de Pelotas (UCPel)
Centro de Ciências Sociais e Tecnológicas
Engenharia de Computação

Sistema Automatizado de Controle de Jardins

Aluno: Kaller Moraes Gonçalves
Professor: Adenauer Correa Yamin

Pelotas
2020

Universidade Católica de Pelotas (UCPel)
Centro de Ciências Sociais e Tecnológicas
Engenharia de Computação

Sistema Automatizado de Controle de Jardins

Relatório da final da disciplina sistemas embarcados do curso de Engenharia de Computação da Universidade Católica de Pelotas (UCPel), com intuito de implementar o conceitos e métodos trabalhados durante o semestre.

Aluno: Kaller Moraes Gonçalves

Professor: Adenauer Correa Yamin

Pelotas
2020

Conteúdo

1 Introdução

As plantas são consideradas a principal fonte de sobrevivência e ajudam a purificar o ar cheio de poluentes. muitos se sentem responsáveis por plantar uma árvore e alguns consideram isso um hobby, plantar uma árvore não é somente enterrar uma semente no solo, tem muitos fatores a serem considerados.

Algumas plantas precisam de mais cuidados para um crescimento eficiente. Existem algumas plantas que são cultivadas apenas para fins de demonstração ou para gerar beleza ao lugar como nos jardins (agricultura caseira).

O ambiente necessário deve ser fornecido à planta, a mesma deve ser regada de vez em quando para fazer a fotossíntese acontecer. Também se sabe que um tipo de solo ou nutrientes não é suficiente para que todas as plantas cresçam da melhor forma, cada planta tem suas características para ganhar um crescimento satisfatório. para tentar contornar esses problemas, a ideia é montar um controlador para monitorar e agir, com o intuito de tornar a utilização da água mais eficiente.

2 Objetivos

Nesta seção serão apresentados os objetivos gerais do projeto, bem como os objetivos específicos.

2.1 Objetivos Gerais

O objetivo do trabalho consiste em aplicar as técnicas que estão sendo abordada na disciplina de forma que seja possível criar um sistema com uma capacidade de atuação em jardins, de forma que será capaz de utilizar a água da melhor maneira possível, consequentemente ajudando o meio ambiente e o melhor desenvolvimento das plantas.

2.2 Objetivos Específicos

Os objetivos específicos deste projeto será de como o sistema vai se comportar a medida que novos problemas surgem visando um entendimento maior do assunto e planejamento de projetos de sistemas automatizados(IoT).

3 Composição do Projeto

Nesta seção sera mostrado os componentes essenciais para o desenvolvimento do projeto, hardwares, protocolos e especificações mais detalhadas.

3.1 ESP32

O ESP32 é um dispositivo IoT (Internet das Coisas) que consiste de um microprocessador de baixa potência dual core Tensilica Xtensa 32-bit LX6 com suporte embutido à rede WiFi, Bluetooth v4.2 e memória flash integrada, essa arquitetura permite que ele possa ser programado de forma independente, sem a necessidade de outras placas microcontroladoras como o Arduino.

Na imagem abaixo pode-se ver um breve comparativo entre a ESP32, ESP8266 e o Arduino Uno R3:

	ESP32	ESP8266	ARDUINO UNO R3
Cores	2	1	1
Arquitetura	32 bits	32 bits	8 bits
Clock	160MHz	80MHz	16MHz
WiFi	Sim	Sim	Não
Bluetooth	Sim	Não	Não
RAM	512KB	160KB	2KB
FLASH	16Mb	16Mb	32KB
GPIO	36	17	14
Interfaces	SPI / I2C / UART / I2S / CAN	SPI / I2C / UART / I2S	SPI / I2C / UART
ADC	18	1	6
DAC	2	0	0

Figura 1: Comparativo ESP32.

O NodeMCU-32S é uma plataforma de prototipagem baseada no ESP32 e que é comumente utilizada no desenvolvimento de projetos IoT. A placa já conta com conversor USB serial integrado e porta micro USB para alimentação e programação.
[1]

[1]Informações para esp32 link:
https://www.espressif.com/en/products/socs/esp32
https://pt.wikipedia.org/wiki/ESP32
https://www.eletrogate.com/modulo-wifi-esp32-bluetooth-30-pinos

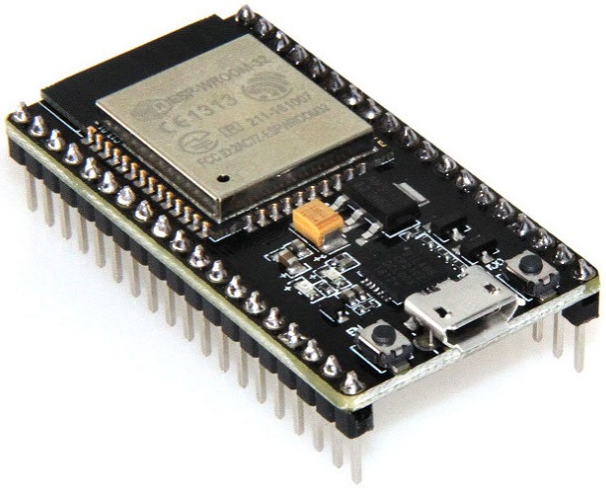

Figura 2: Placa NodeMCU-32S(ESP32).

3.2 Sensor de Temperatura LM35

O Sensor de Temperatura LM35 é um sensor de precisão, que apresenta uma saída de tensão linear relativa à temperatura em que ele se encontrar no momento em que for alimentado por uma tensão de 4-20Vdc e GND.

O LM35 não necessita de qualquer calibração externa ou "trimming" para fornecer com exatidão, valores temperatura com variações de 0,25ºC ou até mesmo 0,75ºC dentro da faixa de temperatura de –55ºC à 150ºC, o sensor tem saída com baixa impedância, tensão linear e calibração inerente precisa.

O sensor pode ser alimentado com alimentação simples ou simétrica, dependendo do que se desejar como sinal de saída, mas independentemente disso, a saída continuará sendo de 10mV/ºC. Ele drena apenas 60A para estas alimentações, sendo assim seu auto-aquecimento é de aproximadamente 0.1ºC ao ar livre.
[2]

[2]Dados sensores link das informações:
https://www.baudaeletronica.com.br/sensor-de-temperatura-lm35.html
https://www.usinainfo.com.br/blog/sensor-de-temperatura-lm35-primeiros-passos
https://www.vidadesilicio.com.br/lm35-sensor-de-temperatura
http://blog.novaeletronica.com.br/lm35-o-sensor-de-temperatura-mais-popular

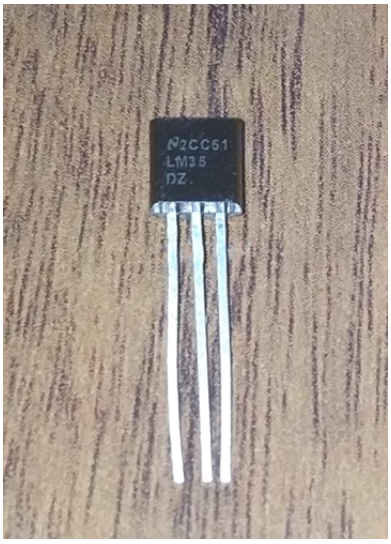

Figura 3: Sensor de temperatura LM35.

3.3 Sensor de umidade do solo

O sensor de umidade do solo pode ler a quantidade de umidade presente no solo ao seu redor, este especificamente é um sensor low tech mas ideal para monitorar um jardim urbano ou sua planta de estimação.

O funcionamento consiste em utilizar dois eletrodos para passar corrente pelo solo e ler o nível de umidade por comparação com a resistência, a medida que se tem mais água o solo diminui a resistência enquanto um solo seco conduz com mais dificuldade.

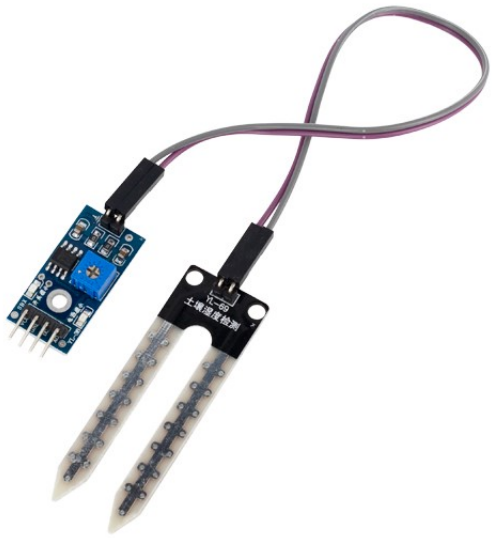

Figura 4: Sensor de Umidade do solo.

3.4 Sensor de Luminosidade

O Sensor de Luminosidade LDR (Light Dependent Resistor) é um componente cuja resistência varia de acordo com a intensidade da luz, quanto mais luz incidir sobre o componente, menor a resistência, o sensor de luminosidade pode ser utilizado em diversos projetos com vários microcontroladores diferentes e em diversas áreas como: alarmes, automação residencial, sensores de presença e etc.

Figura 5: Sensor de Luminosidade.

3.5 Micropython

MicroPython é uma implementação de software de uma linguagem de programação amplamente compatível com Python 3, escrita em C , que é otimizada para rodar em um microcontrolador.

MicroPython é um compilador Python completo e runtime que roda no hardware do microcontrolador, O usuário recebe um prompt interativo (o REPL) para executar os comandos suportados imediatamente. Estão incluídos uma seleção de bibliotecas Python centrais, sendo que o MicroPython inclui módulos que fornecem ao programador acesso a hardware de baixo nível.
[3]

3.6 MQTT

MQTT, sigla de MQ Telemetry Transport, é um protocolo de mensagens leve para sensores e pequenos dispositivos móveis otimizado para redes TCP/IP, o esquema de troca de mensagens é fundamentado no modelo Publicador-Subscritor, extremamente simples e leve.

[3]Link das Referencias: https://micropython.org
https://www.embarcados.com.br/micropython
https://www.usinainfo.com.br/blog/micropython-esp32-parte-1

Os princípios arquitetônicos são minimizar o uso de banda de rede e uso de recursos dos equipamentos enquanto garantindo confiabilidade e algum nível de garantia de entrega, estes princípios tornam esse protocolo ideal para as comunicações emergentes (M2M) "machine-to-machine" e para as aplicações "Internet of Things" (IoT) um mundo de equipamentos conectados, além das aplicações mobile onde banda e potência da bateria são relevantes. Atualmente se encontra na versão 5.0 e a 3.1.1. [4]

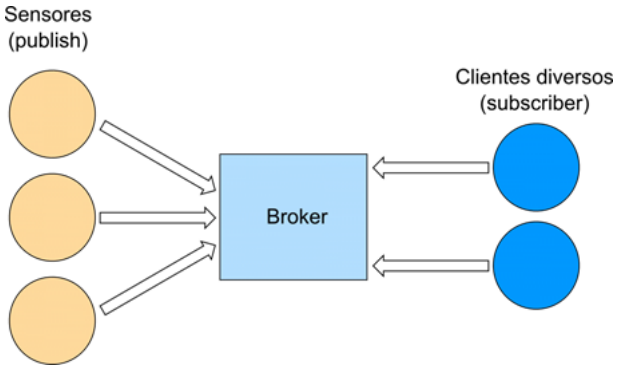

Figura 6: Exemplo Publicador-Subscritor.

3.7 OpenWeatherMap

OpenWeatherMap é um serviço on-line, de propriedade de OpenWeather Ltd, que fornece dados globais meteorológicos, incluindo os dados meteorológicos atuais, previsões , nowcasts e dados históricos (a partir de 1979), utilizando os serviços de radiodifusão meteorológicas e dados brutos de aeroporto estações meteorológicas , estações de radar e outros estações meteorológicas.

A empresa tem mais de 2 milhões de clientes, desde desenvolvedores independentes a empresas Fortune 500, ela fornece mais de vinte APIs de clima , com quase 7.000 repositórios no GitHub, as APIs oferecem suporte a vários idiomas, unidades de medida e formatos de dados.

A plataforma usa o OpenStreetMap para exibir mapas meteorológicos e fornece muitos de seus serviços gratuitamente. Mas, apesar do nome da plataforma, ela não está aberta a contribuições de usuários, e a maioria de seus serviços exige assinaturas.

[4]Link das Referencias: https://mqtt.org/
https://pt.wikipedia.org/wiki/MQTT
https://www.gta.ufrj.br/ensino/eel878/redes1-2019-1/vf/mqtt/
https://developer.ibm.com/br/technologies/iot/articles/iot-mqtt-why-good-for-iot/

5

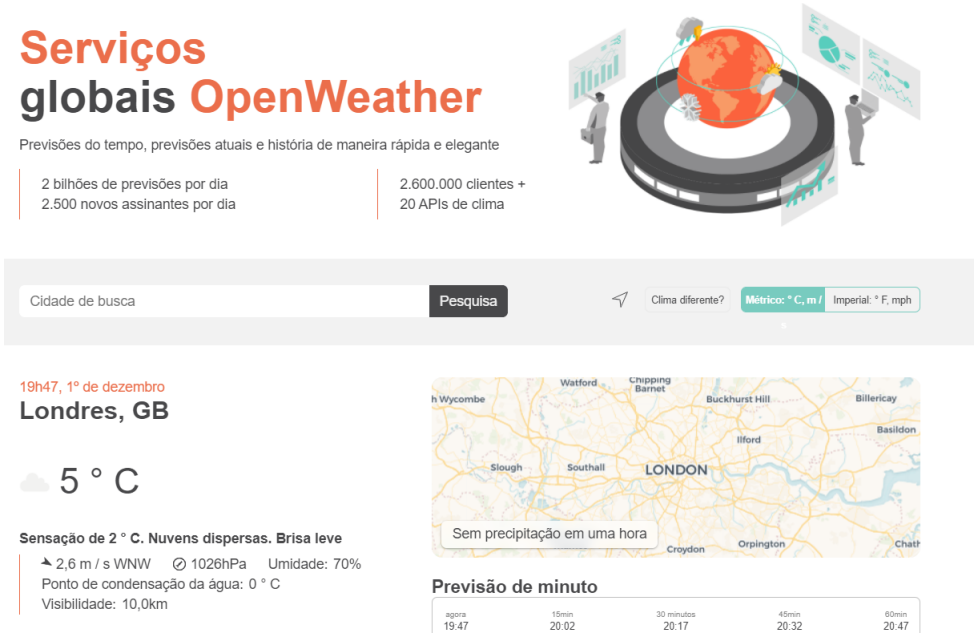

Figura 7: OpenWeather Plataforma digital.

[5]Link para referencias: https://openweathermap.org/

4 Levantamento das informações do processo

No cronograma de desenvolvimento foi estipulado como primeira etapa o levantamento de dados do processo para desenvolvimento do dispositivo e chegou-se a seguinte conclusão:

Após uma pesquisa sobre o assunto, foi possível perceber que o foco mais dialogado sobre o assunto é sobre a utilização e forma de otimizar o uso da água no ambiente, a partir disso foi possível ter uma ideia do que seria necessário para desenvolver o software para se utilizar com o microcontrolador.

5 Funcionamento detalhado

Qualquer alteração detectada pelos sensores que serão ligadas nas portas analógicas da ESP32, isso irá ocasionar uma reação no sistema, e através da conectividade WIFI e do protocolo MQTT ira se comunicar com o broker enviara as informações do estado atual, através do que o usuário deseja fazer a irrigação ou parar a mesma, ou através da inteligência do sistema automatizado que analisa a porcentagem de chover em determinado tempo, a temperatura e umidade ele toma a decisão de iniciar a irrigação.

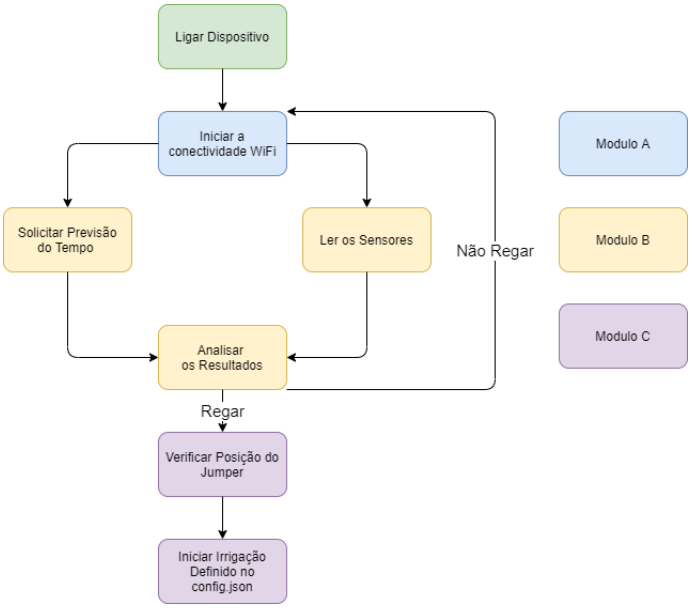

Figura 8: UML do Funcionamento.

A seguir a definição dos funcionamentos dos módulos separadamente:

5.1 Modulo A

Este modulo consiste em estabelecer uma conexão wifi para o funcionamento dos módulos MQTT e APIs de comunicação. A inicialização verifica se já existe uma Wifi pré configurada e salva, caso não exista o modulo cria uma wifi um celular ou computador conectar na rede, ao abrir o navegador no IP(192.168.4.1) poderá se escolher as redes disponíveis para a conexão, apos a inserção da senha, a conexão é inicializada.

Figura 9: Amostragem das redes disponíveis.

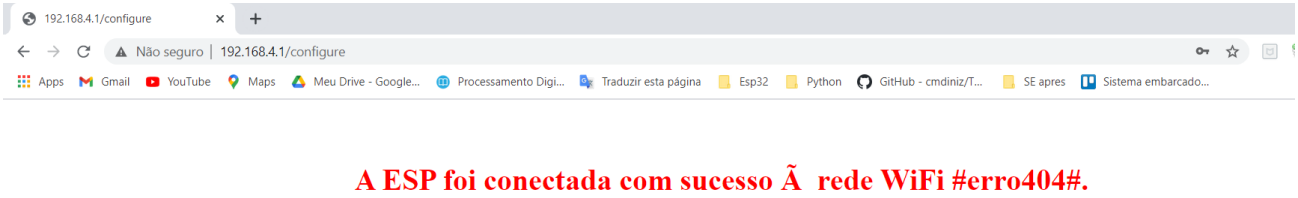

Figura 10: Conexão bem sucedida.

5.2 Modulo B

Após a conexão com a internet ser estabelecida, a microcontroladora ira se conectar com o broker MQTT definido no arquivo config.json, este arquivos contem todos os dados de configurações que o sistema precisa.

Após a conexão com o broker feita ele inicializa no tópico do ID(identificador único do dispositivo.), onde é possível configurar a cidade entre outros tópicos do dispositivo.

Na segunda etapa do modulo, o dispositivo inicia uma thread, que esta a fase de conexão com a API de previsão do tempo, assim solicitando os dados de determinada cidade configurada, sendo assim salva os dados recebidos da API no arquivo $clima_t.json$.

```
#Modulo para Conexão MQTT
import ujson
import urequests as requests
import network
from umqtt.simple import MQTTClient
import machine
import time
from time import sleep
import _thread


global led1
global led2
def rotina_led(pino,valor):
   led = machinePin(pino,Pin.OUT)
   while True:
       led.value(not led.value())
       sleep(valor)
#função para confirgutar o led 2 através dos dados obtidos via mqtt
def mqtt_cidade():
    ledd = "led1"+".txt"
    while True:
        f = open(ledd)
        print("Led1: " +"Valor:" +f.read())
        f = open(ledd)
        oi =f.read()
        sleep(int(oi))
#função para confirgutar o led 2 através dos dados obtidos via mqtt
def led_on_2():
    ledd = "led2"+".txt"
    while True:
        f = open(ledd)
        print("Led2: " +"Valor:" +f.read())
        f = open(ledd)
        oi =f.read()
        sleep(int(oi))


def sub_cb(topic, msg):
    with open('config.json', 'r') as config_file:
        dados = ujson.load(config_file)
```

```
    mqtt = dados['MQTT']
    topico = mqtt['topico']
    id = dados['ID']
    top = str(topico + "/" + id)
   #topico para o controle da cidade
    if(str(topic,'utf-8') == str(top + "/cidade")):
        dados['cidade'] = str(msg,'utf-8')
        arquivo = open("config.json", "w")
        ujson.dump(dados, arquivo)
        arquivo.close()
        print("cidade chegou")
    #topico para o controle do led 2
    elif(str(topic,'utf-8') == str(top + "/led2")):
        f = open('led2.txt','w')
        f.write(str(msg,'utf-8'))
        f.close()
        print("leddd 2 chegou")
    else:
        print("invalido")



def inicializar():
    #leitura json
    with open('config.json', 'r') as config_file:
        dados = ujson.load(config_file)

    mqtt = dados['MQTT']
    usuario_mqtt = mqtt['usuario']
    senha_mqtt = mqtt['senha']
    topico = mqtt['topico']
    id = dados['ID']
    top = str(topico + "/" + id)

    #dados do broker mqtt (nome do dispositivo, broker, usuario)
    client = MQTTClient("esp-kaller", "ioticos.org",user= usuario_mqtt,
    client.set_callback(sub_cb)
    client.connect()
    client.subscribe(topic= str(top + "/cidade"))
    client2 = client
    client2.subscribe(topic= str(top + "/led2"))
```

```
    def estou_conectado():
        while True:
            client.publish(topic=top + "/conectado", msg="conectado")
            time.sleep(15)
    #t =_thread.start_new_thread(mqtt_cidade,())
    #t2 =_thread.start_new_thread(led_on_2,())
    t =_thread.start_new_thread(estou_conectado,())

    while True:
        client.wait_msg()
        time.sleep(1)
        client2.check_msg()
        time.sleep(1)

#Módulo para pegar os dados da API de previsão do tempo
import urequests as requests
import ujson
import network
import time

#função pegar prev do tempo
def prev_temp():
        while True:
            with open('config.json', 'r') as config_file:
                dados = ujson.load(config_file)

            cidade = dados['cidade']
            api_key = dados['API_Key']
            cidade_v = cidade.replace(" ", "%20")
            link = "http://api.weatherapi.com/v1/forecast.json?key="
            + api_key + "&q=" + cidade_v +"&days=1"
            print (link)
            res = requests.get(url= link)
            dado = res.json()
            with open('clima_t.json', 'w') as clima:
                ujson.dump(dado,clima)
            clima.close()
            print("clima atualizado")
            time.sleep(600)
```

5.3 Modulo C

Neste modulo esta a etapa final,ou seja ler os sensores e o valor dos dados recolhidos da API, a partir dai com as definições contidas no arquivo config.json, é possível formar uma leve inteligência para a economia de água.

Analisando a probabilidade de chuva do dia e a forma que o jumper(seletor) se encontra o nível e o horário de atuação podem mudar, consequentemente o ligar da bomba ou não.

Ligando a bomba ela tem um tempo de atuação determinado no config.json, oq deixa o sistema bem configurável a medida que os desafios mudam.

6 Conclusão

A medida que o trabalho foi desenvolvido baseado nos conteúdos e formas descritos ao decorrer da cadeira, tendo seu apelo para uma implementação de sistemas embarcados em um microcontrolador, inicialmente o desafio foi caracterizar o "sistema operacional"melhor dizendo a forma de programação do projeto, foi optado por micropython, após os requisitos do sistema junto com qual área ele iria atuar, após a definição foi iniciada a etapa de implementação, sendo esta uma divisora de água pois realmente nós iríamos começar a desenvolver o projeto proposto, com o inicio do projeto tive diversos aprendizados de como utilizar as bibliotecas junto com o entendimento melhor de protocolos diferentes de comunicação, conectividade com a internet entre outros.

Como perspectiva de uma melhoria para o futuro seria a tentativa de implementar horários para diversos para atuação, junto com estabelecer um padrão de previsão de hora em hora em vez de uma previsão diária como é atualmente.

Finalizando pode-se dizer que o trabalho conseguiu fazer o aprofundamento do conhecimento sobre a área mostrando diversos aspectos para um desenvolvimento de projetos de sistemas embarcados.