Controlador PID de Temperatura de Piscina Inteligente

Ponto de Controle 4 - Atualização do Projeto

Sistemas Operacionais Embarcados -

Matheus Gois Vieira
Engenharia Eletrônica
Faculdade do Gama - UnB
Email: matheusmaxi9.0@gmail.com

Victor Kaio Rodrigues Pires
Engenharia de Eletrônica
Faculdade do Gama - UnB
Email: victorkaiorp@gmail.com

***Abstract*—O projeto é o desenvolvimento de controle PID da temperatura de uma piscina a partir de processamento de áudio. Basicamente será feito a coleta, caracterização e classificação de um áudio coletado e em seguida será feito o controle da temperatura de uma piscina a partir dos dados coletados. Todos os dados podem ser vistos com a utilização de telemetria e com a estabilidade do controle PID alcançada será avisado ao usuário com sinais sonoros e visuais, bem como uma mensagem no Telegram, para que o mesmo possa se situar se a piscina está na temperatura desejada. O projeto está sendo feito pois poderá ser testado posteriormente em um centro de natação, tem uma causa social, pois existem pessoas que praticam esportes aquáticos e possuem algum tipo de deficiência e estão em algum tipo de tratamento utilizando a piscina e estão impossibilitados de manusear um equipamento com as mãos, e o controle por voz facilitaria o processo. O desenvolvimento do projeto está sendo finalizado e os objetivos estão sendo alcançados.**

I. Objetivos

Desenvolver um sistema embarcado que através da captura de áudio, e utilizando uma API do Google Assistant, é identificado e interpretado o conteúdo do áudio de entrada com o objetivo de controlar por voz a temperatura de uma piscina. Com isso, os seguintes objetivos específicos são traçadas:

- Capturar áudio com um microfone e utilizar o hardware interno da Raspberry Pi .
- Desenvolver camada de software capaz de conectar a API do Google Assistant com a microprocessador.
- Definir e configurar frases chaves para o controle da temperatura da piscina.
- Desenvolver sistema de controle PID da piscina.
- Desenvolver camada de software capaz de conectar o controle PID com o Google Assitant.
- Criar hardware capaz de capturar temperatura da piscina
- Comunicar camada de hardware com o sistema de aquecimento da piscina.
- Disparar alarme para possíveis adversidades.
- Apresentar dados de temperatura utilizando o Node-Red.
- Enviar os dados de temperatura utilizando o protocolo MQTT.

II. Justificativa

Hoje em dia com estudos e análises feitos em piscinas foram encontrados grandes benefícios no seu controle de temperatura, seja ela por questões recreativas, a água quente acaba aumentando o tempo que você a aproveita, seja por questões terapêuticas, pois o conforto e comodidade de pessoas quando utilizam uma piscina é necessário, ou sejam por motivos esportivos. E evidentemente, o nível de pH de uma piscina deve ser medido constantemente, afinal é muito importante medir e, caso seja necessário, corrigir o pH da sua piscina. O controle de pH permite que os usuários usufruam melhor da água, com mais conforto, sem o risco de ficar com os olhos ardendo, com cabelos danificados e problemas na pele.

Somando a isso, esse sistema pode servir como ferramenta de inclusão social, devido o fato que existem alunos de esportes aquáticos, ou até mesmo, alunos que possuem algum tipo de deficiência e estão em algum tipo de tratamento utilizando a piscina e que são impossibilitados de manusear um equipamento com as mãos, e o controle por voz facilitaria o processo.

Outro grande motivo para a realização do projeto é a possibilidade para aplicação do mesmo em um sistema real para piscina do SESI - Gama, assim, tendo um produto com viabilidade de ajudar alguém.

Mas o grande motivo da realização desse projeto é a automação com IoT do sistema de controle de temperatura de piscinas, pois hoje em dia basicamente tudo pode vem sendo automatizado e integrado com IoT. Existem sistemas que fazem o controle da temperatura de piscinas e existem sistemas que fazem a medida do pH, mas nenhum deles é integrado com IoT, até mesmo nas piscinas olímpicas o controle de temperatura e de medição de pH vem sendo feitos de forma manual e não possuem um sistema de telemetria. Assim, um sistema de Telemetria e controle seriam ideias.

III. Requisitos

Por convenção, a referência a requisitos é feita através do nome da subseção na qual eles estão descritos, seguidos do identificador do requisito, de acordo com o seguinte modelo:

[nome da subseção/identificador do requisito]

Os requisitos serão identificados com um identificador único, com numeração iniciada em [RF01] para um requisito funcional e [NF01] para um requisito não-funcional.

Para estabelecer a prioridade dos requisitos foram adotadas as denominações "essencial", "importante" e "desejável", tal que: Essencial é o requisito sem o qual o sistema não entra em funcionamento. Requisitos essenciais são aqueles imprescindíveis, que devem ser implementados impreterivelmente. Importante é o requisito sem o qual o sistema entra em funcionamento, mas de forma não satisfatória. Requisitos importantes devem ser implementados, mas, se não forem, o sistema poderá ser implementado e usado normalmente. Desejável é o requisito que não compromete as funcionalidades básicas do sistema, isto é, o sistema pode funcionar de forma satisfatória sem ele. Requisitos desejáveis podem ser deixados para versões posteriores do sistema, caso não haja tempo hábil para implementação dos mesmos nesta etapa.

A. Requisitos Funcionais

[RF01] O sistema deve medir temperatura: O sistema deve ser capaz de fazer medidas de temperatura da piscina com o sensor adequado. Prioridade: Essencial

[RF02] O sistema deve medir o pH da água: O sistema deve ser capaz de fazer medidas do pH da piscina com o sensor adequado. Prioridade: Essencial

[RF03] O sistema deve enviar as informações via protocolo MQTT: O sistema deve ser capaz de se comunicar com os demais hardwares com a utilização do protocolo MQTT. Prioridade: Essencial

[RF04] O sistema deve mostrar os dados coletados ao usuário: O sistema deve ser capaz de mostrar os dados coletados em um site ou APP quando o usuário solicitar. Prioridade: Essencial

[RF05] O sistema deve identificar palavras-chaves via áudio: O sistema dever ser capaz de identificar com a ajuda da inteligência artificial e com os conceito de processamento de sinais as palavras chaves. Prioridade: Essencial

[RF06] O sistema deve processar ações a partir do que é identificado no áudio: O sistema deve ser capaz de realizar uma ação específica e já programada a partir da identificação da palavra-chave identificada. Prioridade: Essencial

[RF07] O sistema deve fazer o controle da temperatura com PID: O sistema dever ser capaz de realizar o controle da temperatura da piscina com um sistema de controle, sistema esse que será o PID. Prioridade: Essencial

[RF08] O sistema deve ter um Hardware capaz de coletar as informações necessárias: O sistema deve ter um Hardware, simulado e testado, capaz de realizar a coleta de dados necessários para processamento. Prioridade: Essencial

[RF09] O sistema deve informar ao usuário quando a temperatura for alcançada: O sistema dever ser capaz de informar ao usuário quando o processamento e controle acabar, ou seja, quando a temperatura chegar na ideal escolhida pelo o usuário. Prioridade: Essencial

[RF10] O sistema deve ter o controle para diferentes temperaturas: O sistema deve ser capaz de realizar o controle de temperatura para faixas diferentes, de acordo com o critério da FINA. Prioridade: Essencial

[RF11] O sistema deve ser capaz de se comunicar com o sistema de aquecimento da piscina: O sistema de que faz a identificação e processamento das palavras-chaves deve ser capaz de realizar uma comunicação a distância com o sistema de controle de temperatura da piscina. Prioridade: Essencia

[RF12] O sistema deve ter um Hardware capaz de realizar o controle PID: O sistema deve ter um Hardware capaz de realizar o controle da temperatura com um PID. Prioridade: Essencial

[RF13] O sistema deve fazer a medida de consumo: O sistema deve realizar a análise do consumo de potência do sistema. Prioridade: Desejável

[RF14] O sistema deve funcionar com a Raspberry pi: O sistema de processamento e coleta de áudio deve ser feito totalmente na Rapsberry pi. Prioridade: Importante

[RF15] O sistema deve ter telemetria: O sistema deve ter telemetria para mostrar ao usuário os dados. Prioridade: Essencial

B. Requisitos Não-Funcionais

[NF01] Portabilidade: O sistema de coleta de dados do áudio deve ser portátil para um conforto melhor para o usuário. Prioridade: Desejável

[NF02] O sistema deve ter os sensores discretos: O sistema deve ser discreto para evitar danos causados pelos banhistas. Prioridade: Importante

[NF03] Tamanho compacto: O sistema de coleta de áudio deve ser compacto para validar melhor a sua portabilidade. Prioridade: Desejável

[NF04] O sistema deve ter uma alta precisão: O sistema deve ter uma alta precisão na coleta de dados e no controle da temperatura. Prioridade: Desejável

[NF05] O sistema deve ter um banco de baterias: O sistema dever ter um banco de baterias para validar a portabilidade do mesmo. Prioridade: Desejável

[NF06] O sistema deve ter um banco de dados: O sistema deve ter um banco de dados para guardar as medidas feitas e disponibilizar posteriormente para estudo. Prioridade: Importante

[NF07] O sistema deve fazer gráficos com os dados medidos: O sistema deve fazer gráficos personalizados dos dados de temperatura e Ph coletados. Prioridade: Importante

IV. Benefícios

Na aplicação deste projeto, é possível enxergar benefícios que ajudam a compreender a importância e sua contribuição para o mercado de controle termal de piscinas aquecidas, são eles:

- Oportunidade para aplicar em um sistema real (piscina do SESI)
- Inclusão social para pessoas com comodidade para paraolimpíadas
- Comodidade e praticidade para controlar a temperatura.
- Não requer a presença de controles ou dispositivos físicos para controlar o sistema, apenas por voz
- Controle de temperatura eficiente para atingir condições ideias para práticas de atividades aquáticas
- Monitoramento preventivo
- Acionamento remoto do sistema
- Alarmes adversidades

V. Revisão Bibliografica

Google Assistant Docs

O documentação da Goggle Assistant Service provê como embarcar a API em um dispositivo que contém um sistema operacional como a Raspbery, bem como, exemplo de integração com projeto auxiliares. Google Assistant oferece controle total sobre a integração com o Assistant, fornecendo um endpoint de streaming. Transmite uma consulta de áudio do usuário para este endpoint para receber uma resposta de áudio do Google Assistant.

Engenharia de Sistemas de Controle - Norman Nise

Um sistema de controle consiste em subsistemas e processos (ou plantas) reunidos com o propósito de controlar as saídas dos processos. A construção de um sistema de controle é dado por razões como amplificação de potência, controle remoto, facilidades de uso da forma de entrada e compensação de pertubações. No projeto será construído um Controle PID, que fornece uma variação contínua da saída dentro de um mecanismo de realimentação de loop de controle para controlar com precisão o processo, removendo a oscilação e aumentando a eficiência.

Fundamentos da Instrumentação - Luis Antonio Aguirre

Um sistema de medição é um conjunto de dispositivos (sensores, circuitos, cabos, visores, equações, programas de computador), cujo objetivo é fornecer informação sobre o valor da grandeza física que se deseja medir, o mesurando. A obra fornece etapa de análise e validação de dados coletar a partir de um sistema de medição de malha fechada com realimentação.

Swimming Pool Certificate Guide - Fina

A FINA, Federação Internacional de Natação, foi criada em 1908 e atualmente representa 202 federações nacionais distintas. A FINA representa o Comitê Olímpico Internacional e é o órgão regulador mundial dos esportes aquáticos. Ele determina as regras e regulamentos para natação, mergulho, pólo aquático, nado sincronizado e nado em águas abertas. Para natação nas competições padrão e nas Olimpíadas, a FINA exige uma temperatura da água entre 25 e 28 graus C, ou entre 77 e 82 graus F.

Embedded Technology: Linux for Embedded and Real-Time Application - Doug Abbott

Nesta obra orientada a aplicativos embarcados, o autor mostra como colocar o Linux para funcionar em aplicativos integrados e em tempo real. Descreve soluções de como incluir gerenciamento de memória, drivers de dispositivo, tratamento de interrupção, instrumentação de kernel, carregadores de barco, rede incorporada, comunicações entre tarefas, camadas de abstração de hardware e depuração de programa.

VI. Hardware

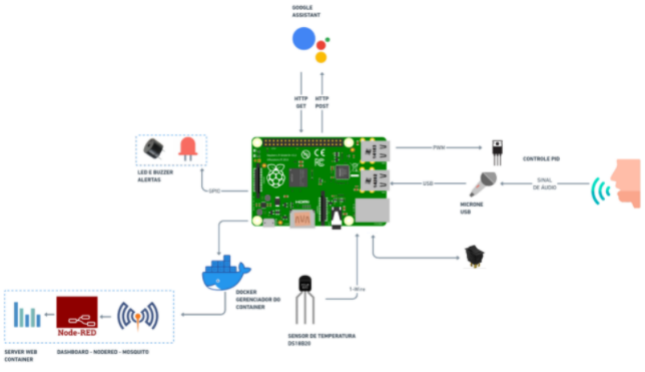

Fig. 1: *Hardware do sistema*

A. Subsistema de Sinalização

Foi realizado o esquemático e a montagem do subsistema de sinalização com o Ler RGB e o buzzer, para avisos visuais e sonoros, respectivamente.

B. Led RGB

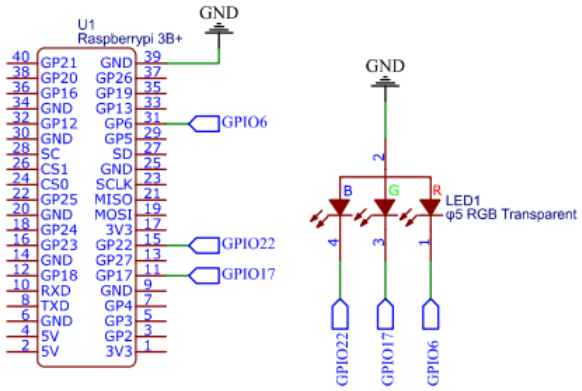

Fig. 2: *Esquemático da ligação do led rgb com o RPi*

É interessante ressaltar que o RPi possui um resistor de pull-up interno.

C. Buzzer

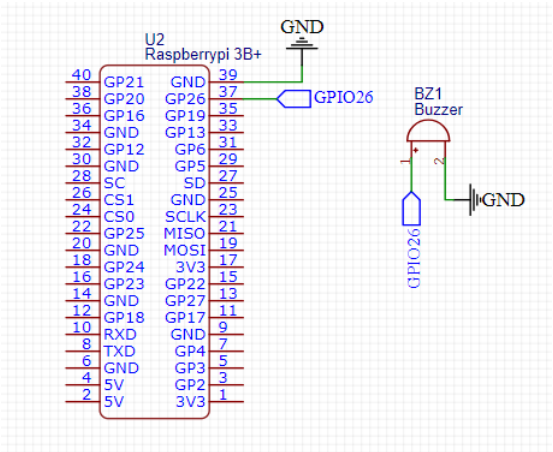

Fig. 3: *Esquemático da ligação do buzzer com o RPi*

D. Controle PID

Foi feito o subsistema para coleta de dados e para o controle PID.

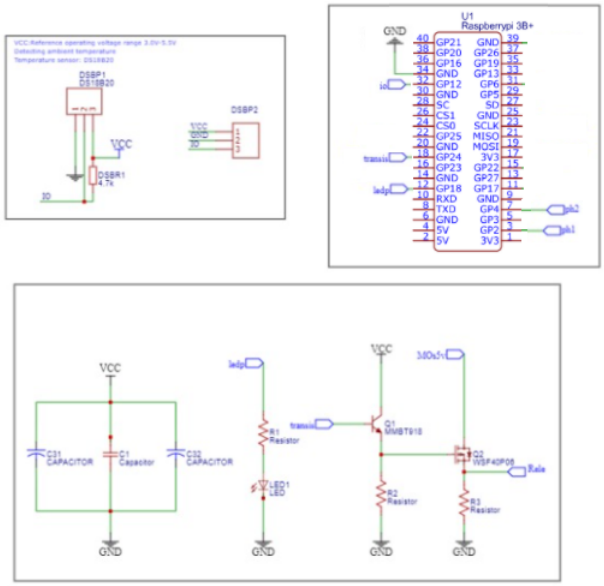

Fig. 4: *Esquemático do sistema de coleta e controle PID*

E. Layout para circuito impresso

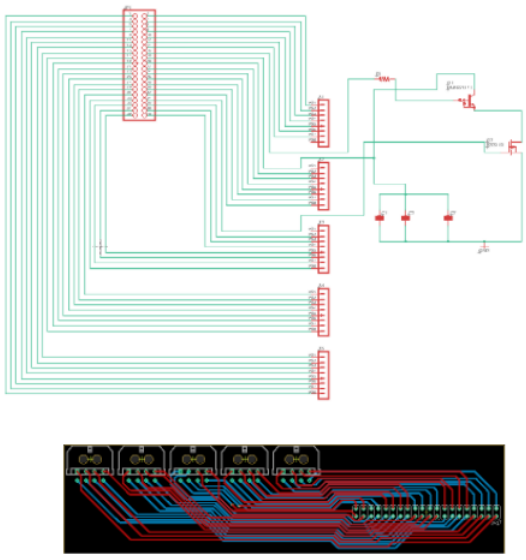

Fig. 5: *Layout*

Foi feito teste e coleta de dados para validar o sistema e a sua funcionalidade.

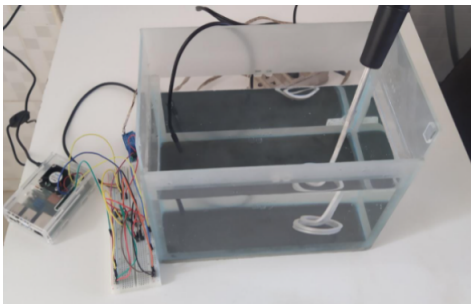

Fig. 6: *Sistema completo montado*

VII. Software Embarcados

A. Fluxo de funcionamento

Para que o sistema seja um produto final entregável e para aplicação real, é necessário que o sistema realize uma série de process e threads em uma ordem definida desde da sua inicialização. Assim, foi definido um fluxograma para que o sistema trabalhe conforme um produto final.

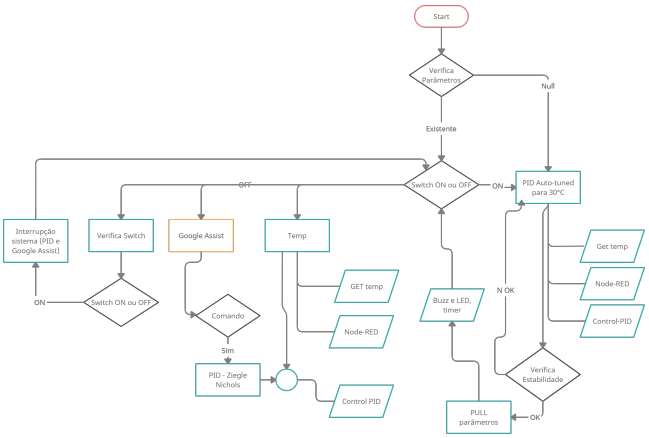

Fig. 7: *Sistema completo montado*

B. Google Assistant

Para a integração da API do Google Assistant com a Raspberry Pi foram feito 4 etapas.

C. Registro do device

Para registrar um device na API do Google é necessário acessar o Google Action Console, criar um projeto e registrar um device com funções de On/Off.

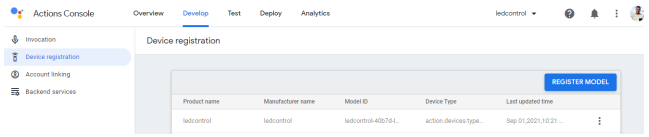

Fig. 8: *Configuração de device no action console*

D. Configurações no GoogleAPIs

É necessário configurar informações de desenvolvimento para a comunicação com a Raspberry.

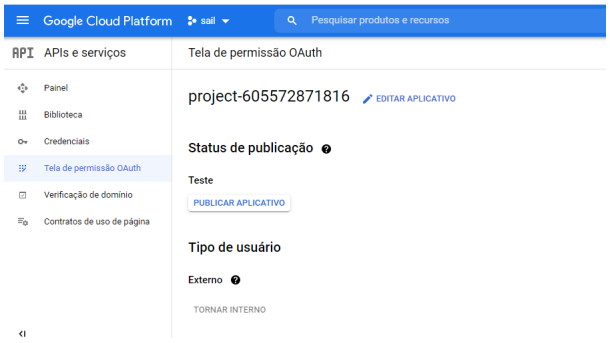

Fig. 9: *Autorização do Google Cloud*

É feito uma modificação na API para que a chamada e as leitura dos arquivos seja realizada

E. Permissão de Usuário

Após as credenciais serem inseridas na Raspberry é necessário permitir que usuários tenham acesso a está requisição configurada.

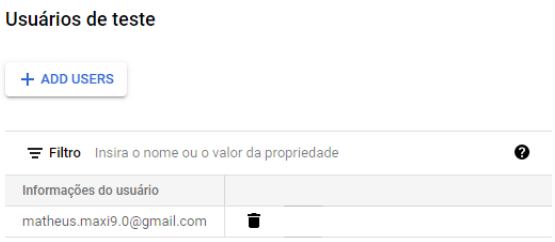

Fig. 10: *Permissão de usuários*

F. Configuração de Microfone e Alto-Falante

Na documentação da Google, é recomendado utilizar um microfone que possua comunicação USB. A primeira coisa a ser feita é verificar as configurações de entrada e saída de áudio da Raspberry.

```
arecord -l // Verificar microfone
aplay -l  // Verificar alto-falante
```

Após verificar as configurações da Raspberry é necessário criar um arquivo .asoundrc no diretório home do RPi e inserir os valores de card e device do microfone e alto-falante.

```
pcm.!default {
      type asym
      capture.pcm "mic"
      playback.pcm "speaker"
}
pcm.mic {
      type plug
```

```
        slave {
                pcm "hw:<card number>,<device
                    number>"
        }
}
pcm.speaker {
        type plug
        slave {
                pcm "hw:<card number>,<device
                    number>"
        }
}
```

G. Subsistema de Sinalização

Duas bibliotecas foram criadas com o intuito de controlar as ações do led rgb e buzzer.

1) Led RGB: A biblioteca criada para controle do led foi chamada de rgb.h, é responsável por 3 principais funções, listadas abaixo.

```
int RGBGreen(int value);// Light on green
int RGBBlue(int value); // Light on blue
int RGBRed(int value);// Light on green
void RGBOff();// Power off all leds
```

- **RGBGreen**: Acende o led verde.
- **RGBRed**: Acende o led vermelho.
- **RGBBlue**: Acende o led azul.
- **RGBOff**: desliga todos os leds.

2) Buzzer: A biblioteca criada para controle do buzzer foi chamada de buzzer.h, é responsável por 1 função apenas, que defini o estado do buzzer. Se o parâmetro value=1 o buzzer emite o sinal sonoro e se o value=0 é desligado o sinal sonoro.

```
int BuzzerState(int value);// toggle buzzer
```

H. Telemetria

O subsistema de telemetria consiste em fazer a integração com a medida, que é realizada pelo sistema de coleta de dados, e o envio de dados via MQQT para o Node-Red, onde devemos cadastrar os dispositivos utilizados e realizar o desenvolvimento de uma interface para mostrar as medidas desejadas, como demostrado nas imagens a seguir:

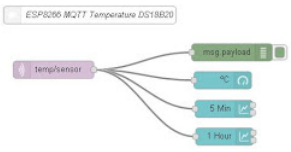

Fig. 11: *Imagem da construção em blocos do Node - Red*

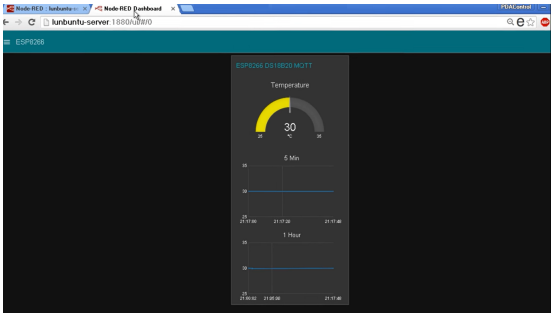

Fig. 12: *Interface do sistema*

As medidas são realizadas com um DS18B20 e enviadas por Wi-fi. Em seguida, é feita a interface e o envio de mensagem pelo Telegram.

E importante ressaltar que para realizar a conexão com o MQQT é necessário configurar, o MQQT broker e o dispositivo que é utilizado para comunicar por ele:

```
const char* ssid = "************";
const char* password = "************";
const char* mqtt_server = "broker.hivemq.com
    "; /// MQTT Broker
int mqtt_port = 1883;
```

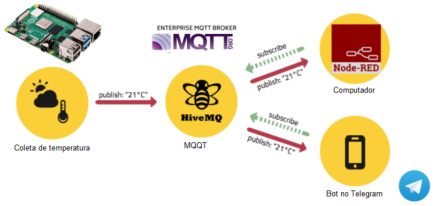

Fig. 13: *Interface do sistema*

I. Coleta de Temperatura

O sensor DS18B20 utiliza do protocolo de comunicação 1-Wire, estes procolo utiliza um barramento para transmissão de dados, provê dados de baixa velocidade, sinalização e sinal único de energia, com taxas mais baixas de dados e maior alcance.

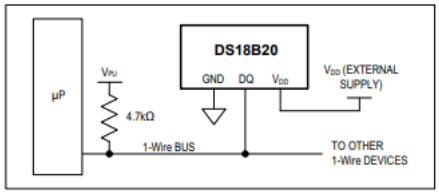

Fig. 14: *Alimentação do DS18B20 com uma fonte externa.*

Na Figura 14 é possível notar a presença de um resistor de pullup que está entre o V_{DD} e DQ, esse resitor é responsável por definir um nível de comparação e inicialização do 1-Wire.

O bloco de código abaixo contém o um estrutura e duas funcões. A estrutura armazena do endereço do arquivo que contém a amostra coletada pelo sensor e o nome dele.

As funções tem os seguintes objetivos:

- **GetSensor**: Definir os dados inciais para a estrutura.
- **ReadTemperature**: Coletar a temperatura do arquivo criado pelo protocolo 1-Wire realizar transformações e entregar um float que representa a temperatura em graus Celsius.

```
typedef struct Sensor
{
    char *SensorName;
    FILE *SensorFile;
} Sensor;

Sensor *GetSensor(char *sensorId, char *
    sensorName);

float ReadTemperature(const Sensor *sensor);
```

J. *Integração do sistemas*

É feito a integração dos sistemas com a chamada do Google Assist e a coleta de temperatura. Basicamente é feito a chamada do Google Assit, no qual é feito uma requisição, e essa requisição faz a leitura de uma arquivo que contem um dado de temperatura, assim como é a leitura do estado do aquecedor.

A integração com o hardware (o Buzzer, o led e o controle PID) é feita diretamente com os sensores e os circuitos conectados na Raspberry.

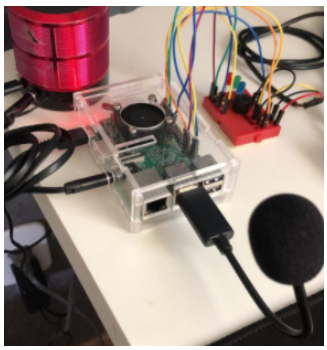

Fig. 15: *Sistema integrado*

K. *Controle PID*

O subsistem de controle PID é modulado conforme a figura a seguir:

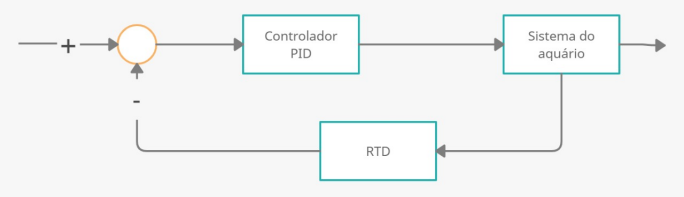

Fig. 16: *Sistema PID*

O PID inicialmente é feito com o método do auto-ajuste (auto-tuned), assim elimina a necessidade de entender o processo e a dinâmica do PID, bem como elimina a necessidade de realizar experimentos para determinar os ganhos do controlador PID.

Esse método é o ideal visto que ele é usado para ajustar o controlador PID automaticamente com base na demanda do usuário. Como existe diferentes tamanhos de piscinas com diferentes aquecedores, esse método ajuda o produto a atender a todas as demandas.

Posteriomente, após a coleta dos parâmetros com o método do auto-ajuste, o PID funcionará a partir do método de Ziegle Nichols.

No entanto é necessário ter um certo cuidado com o sistema, comparando os índices coletados com o experimento e com o método do auto-ajuste temos:

Índices	Ziegler Nichols tuned PID	Auto-tuned PID
Mp	26%	28%
Td	110s	120s
Tr	150s	100s
Tp	460s	300s
Ts	1080s	208s

Fig. 17: *Índices PID*

Como podemos ver, não é exatamente igual, mas para contornar tal problema basta que o aquecedor realize o aquecimento da água lentamente. Caso contrário teremos um sistema com um erro de estabilidade relativamente grande, entre ± 2°C.

A partir da coleta de dados com o método do ajuste, com um aquecedor lento, temos como resultado:

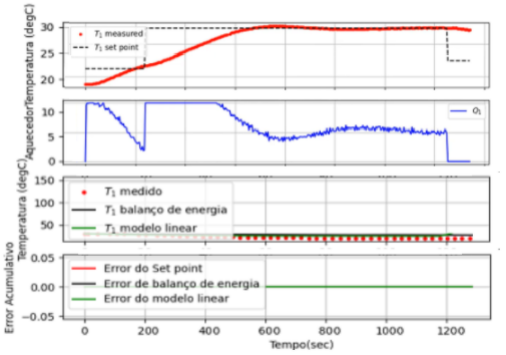

Fig. 18: *Resultado de um dos testes*

Vemos que temos estabilidade, mas o sistema demora 11 minutos para estabilizar (no caso saiu de 20° para 30° em 11 minutos aproximadamente)

O código do controle PID (de Ziegle Nichols e do auto-tuned) se encontra no repositório do GitHub, onde

definimos os cálculos do ganho e é calculado os erros e a resposta do sistema. Temos como exemplo o códifo de Ziegle Nichols:

```
def pid(sp,pv,pv_last,ierr,dt):
    Kc   = 10.0 # K/%Heater
    tauI = 50.0 # sec
    tauD = 1.0  # sec
    # Parameters in terms of PID
        coefficients
    KP = Kc
    KI = Kc/tauI
    KD = Kc*tauD
    # ubias for controller (initial heater)
    op0 = 0
    # upper and lower bounds on heater level
    ophi = 100
    oplo = 0
    # calculate the error
    error = sp-pv
    # calculate the integral error
    ierr = ierr + KI * error * dt
    # calculate the measurement derivative
    dpv = (pv - pv_last) / dt
    # calculate the PID output
    P = KP * error
    I = ierr
    D = -KD * dpv
    op = op0 + P + I + D
    # implement anti-reset windup
    if op < oplo or op > ophi:
        I = I - KI * error * dt
        # clip output
        op = max(oplo,min(ophi,op))
    # return the controller output and PID
        terms
    return [op,P,I,D]
```

VIII. BOM

Material	Funcionalidade	Qualidade
Led Rgb	Responsável por criar 3 tipos diferentes que cores para o projeto.	1
Buzzer	Responsável pelo o alerta sonoro.	1
R 220 Ω	Para criar queda de tensão e evitar uma alta tensão nos leds.	3
Raspberry pi 3 B+	Para o controle PID e coleta de dados a distância.	1
DS18B20	Para coleta de dados de temperatura	1
IRFZ44n	Para o controle chaveamento para o controle PID	1
BC548	Para o controle chaveamento para o controle PID	1
Capacitor 100n	Para estabilizar a resposta No chaveamento do controle PID	1
Capacitor 10n	Para estabilizar a resposta No chaveamento do controle PID	1
Capacitor 47 μ	Para estabilizar a resposta No chaveamento do controle PID	1

IX. Conclusão

Podemos notar que o projeto está desenvolvendo e está sendo possível alcança os objetivos planejados. O controle PID já foi desenvolvido, mas uma otimização com configuração do duty cycle do chaveamento para o aquecedor seja ideal para otimizar o tempo de resposta, que está muito lento.

A integração foi realizada, mas existe ainda pequenos erros que são necessários realizar a correção para ter um produto final entregável. A integração para coleta e telemetria está sendo feita sem problemas, mais alguns pequenos problemas para a integração com o controle PID está ocorrendo.

Por fim, apesar desses pequenos problemas, o projeto está finalizando e estará totalmente feito para o próximo (e último) ponto de controle.

References

Google Assistant Docs. Disponível em: <https://developers.google.com/assistant/sdk/guides/service/python> .Acesso em 03 de Agosto de 2021.

Nise, Norman S., Engenharia de Sistemas de Controle, Editora LTC, 3 ed., 2000

Aguirre, Luis Antonio.Fundamentos da Instrumentação, 2ª edição, Pearson Education, 2013, São Paulo - SP

Swimming Pool Certificate Guide. Disponível em: <shorturl.at/suDE5> .Acesso em 03 de Agosto de 2021.

Abbott, Doug.Swimming Pool Certificate Guide, 2ª edição, Editora LTC

Repositorio do Projeto no Github. disponível em: <https://github.com/matheusgvieira/sailfish>.