UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS

Lawrence Righi Boechat

Proposta economicamente viável para monitoramento remoto de fluxo de hidroponia utilizando sistemas embarcados

São Carlos

2018

Lawrence Righi Boechat

Proposta economicamente viável para monitoramento remoto de fluxo de hidroponia utilizando sistemas embarcados

Trabalho de conclusão de curso apresentado à Escola de Engenharia de São Carlos da Universidade de São Paulo, para obtenção do título de Engenheiro Eletricista

Área de concentração: Engenharia Elétrica com ênfase em Sistemas de Energia e Automção

Orientador: Prof. Dr. Rogério Andrade Flauzino
Co-orientador: Prof. Dr. Evandro Luis Linhari Rodrigues

São Carlos

2018

Ficha catalográfica elaborada pela Biblioteca Prof. Dr. Sérgio Rodrigues Fontes da EESC/USP com os dados inseridos pelo(a) autor(a).

Righi Boechat, Lawrence
R669p Proposta economicamente viável para monitoramento remoto de fluxo de hidroponia utilizando sistemas embarcados / Lawrence Righi Boechat; orientador Rogério Andrade Flauzino; coorientador Evandro Luis Linhari Rodrigues. São Carlos, 2018.

Monografia (Graduação em Engenharia Elétrica com ênfase em Sistemas de Energia e Automação) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2018.

1. Hidroponia. 2. Monitoramento. 3. Agrotecnologia. 4. Irrigação. I. Título.

Eduardo Graziosi Silva - CRB - 8/8907

FOLHA DE APROVAÇÃO

Nome: Lawrence Righi Boechat

Título: "Proposta economicamente viável para monitoramento remoto de fluxo de hidroponia utilizando sistemas embarcados"

Trabalho de Conclusão de Curso defendido e aprovado em 30 / 11 / 2018,

com NOTA 9,9 (NOVE, NOVE), pela Comissão Julgadora:

Prof. Associado Rogério Andrade Flauzino - Orientador - SEL/EESC/USP

Mestre Murilo Eduardo Casteroba Bento - Doutorando - SEL/EESC/USP

Mestre Amélia Moreira Santos - Doutoranda/Docente Temporária - SEL/EESC/USP

Coordenador da CoC-Engenharia Elétrica - EESC/USP:
Prof. Associado Rogério Andrade Flauzino

Este trabalho é dedicado a todos entusiastas da agrotecnologia,
como uma pequena contribuição no esforço em disseminar conhecimento
para o desenvolvimento sustentável.

AGRADECIMENTOS

Agradeço inicialmente ao professor Evandro Luis Linhari Rodrigues, ministrante da fantástica disciplina de aplicação de microprocessadores, aprendizado pela qual este projeto teve início. Como também, deixo aqui um agradecimento especial ao professor e orientador Rogério Andrade Flauzino pela assistência em todo decorrer do curso, como professor e conselheiro em horas difíceis. É uma honra ter sido aluno de ambos.

Meu gentil agradecimento ao departamento da engenharia elétrica e todo corpo integrante da composição do campus de São Carlos, os serviços prestados a mim foram exímios e compuseram não só minha formação profissional como de igual modo na formação humana e de caráter.

Lembranças calorosas a todos amores e amigos, aos que fizeram e aos que fazem parte de minha caminhada. Em especial, aos grandes irmãos de vínculo criados pela república "7 eh poko", alicerces indispensáveis para chegar até aqui.

Por fim, este humilde agradecimento soando pequeno - diante de tamanho apoio, amor e estrutura que foram-me oferecidos - por minha família. Ao meu irmão, meus avós e meus pais, Alvaro e Selene, que este trabalho possa apresentar uma mínima figura da minha eterna gratidão por vocês.

RESUMO

BOECHAT, L. **Proposta economicamente viável para monitoramento remoto de fluxo de hidroponia utilizando sistemas embarcados**. 2018. 60p. Trabalho de conclusão de curso - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2018.

Este trabalho apresenta soluções para problemas existentes no âmbito de cultivos hidropônicos de fluxo laminar, tecnicamente compreendido como NFT(Nutrient Film Technique), visando a rentabilidade econômica de implantação, tanto na inclusão do planejamento de novas estruturas, quanto no anexo de sistemas já operantes. Dentre as adversidades existentes para este tipo de cultivo, destaca-se com alta relevância a possível obstrução dos dutos hídricos, pelos quais fluem a solução nutritiva de alimentação das plantas, ocasionando perdas no cultivo por casos de falta de nutrientes e excesso de vazão na seção irregular do sistema. Para superar tais adversidades, destaca-se a utilização de sistemas embarcados de alta capacidade computacional, baixo custo monetário e consumo energético, aliados ao engenho do uso de dispositivos como sensores de nível de água para a confirmação do bom estado de fluidez nos dutos hídricos do sistema. Nesse sentido, esta pesquisa propõe um protótipo que leva em conta modelos construídos e disponíveis em *Opensource* pela internet em conjunto com a adição de micro tanques de identificação nos finais das tubulações. Com isto, é possível obter a capacidade de monitoramento remoto, a manutenção preventiva do sistema, bem como o poder de decisão da frequência de irrigação no cultivo, gerando aumento de eficiência da safra e redução expressiva no consumo de energia elétrica.

Palavras-chave: Hidroponia. Monitoramento. Agrotecnologia. Irrigação. Sistema Radicular.

ABSTRACT

BOECHAT, L. **Proposta economicamente viável para monitoramento remoto de fluxo de hidroponia utilizando sistemas embarcados**. 2018. 60p. Trabalho de conclusão de curso - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2018.

This work consists in presenting solutions to existing problems in the field of nutrient field technique of hydroponic crops, technically known as NFT, aiming at implementations both in the inclusion of new planned structures and also in the annex of already operating systems. Among the adversities that exist for this type of crop, with high priority of relevance, is the possible obstruction of the dissolved nutrient stream of the pipes, causing losses due to lack of nutrients and excess flow in the irregular section of the system. To solve this issue, this work proposes the advent of embedded computer systems with high computational capacity, low cost and energy consumption, coupled with the ingenuity of using devices such as water level sensors to confirm the optimal flowability in the water system. The proposed prototype takes into account models built and available by Opensource in the internet with the conjunction of micro-identification tanks at the end of the pipes. With this, it is possible to obtain the remote monitoring capacity, the ability of preventive maintenance of the system, as well as the decision power in the frequency of irrigation of the crop, generating increase of efficiency and a significant reduction in the consumption of electric energy.

Keywords: Hidroponic. Monitoring. Agrotechnology. Irrigation. Root System.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

NFT	*Nutrient Film Technique* - Técnica de Fluxo Laminar de Nutrientes
PH	Potencial Hidrogeniônico
IOT	*Internet of Things* - Internet das Coisas
VAC	Volts em corrente alternada
VDC	Volts em corrente contínua
MUX	Multiplexador
HDMI	*High-Definition Multimedia Interface* - Interface Multimídia de Alta Resolução
SSH	*Secure Shell*
IP	*Internet Protocol* - Protocolo de Internet
SO	Sistema Operacional
HTML	*HyperText Markup Language* - Linguagem de Marcação de Hipertexto
USB	*Universal Serial Bus* - Conexão Serial Universal
GPIO	*General Purpose Input/Output* - Entradas/Saídas de Uso Geral
MVC	*Model-View-Controller* - Modelo-Visão-Controlador
POO	Programação Orientada a Objetos
UI	*User Interface* - Interface do Usuário
UML	*Unified Modeling Language* - Linguagem de Modelagem Unificada

SUMÁRIO

1 INTRODUÇÃO

Hidroponia é uma técnica de cultivo de plantas sem o uso do solo ou outro substrato, pela qual os elementos essenciais para o desenvolvimento da cultura são fornecidos através de dutos de solução nutritiva elaborada. O cultivo hidropônico deve ser desenvolvido em ambiente protegido de fatores externos de forma com que ainda esteja disponível uma fonte luminosa, para tanto, um método comumente utilizado é a técnica do fluxo laminar de nutrientes (NFT) em tubos de frestas seccionadas por onde fluem esta solução nutritiva. O conjunto é elementarmente composto por motobomba, reservatório da solução, mecanismo de controle de temporização e o método de canalização de fluxo implementado. As mudas são colocadas com à raízes expostas nestas frestas, geralmente em períodos de 15 minutos alternados entre circulação em intervalo para períodos diurnos e de 15 minutos seguidos de duas horas para os períodos noturnos(CARRIJO O. A.; MAKISHIMA, 2000).

A figura 1 ilustra o funcionamento para este tipo de método de hidroponia.

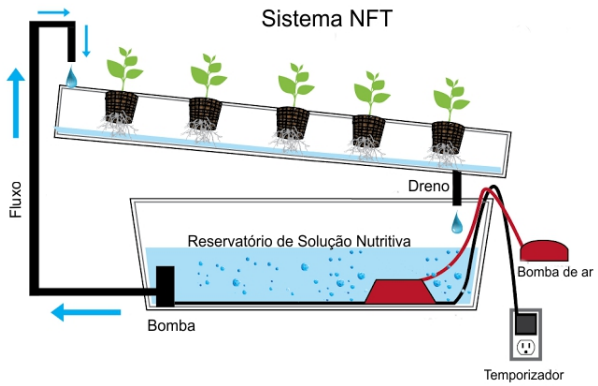

Figura 1: Esquema de funcionamento de hidroponia NFT. Fonte:(VENTURINI, 2014).

Não há consenso ideal para o tempo de circulação (CARRIJO O. A.; MAKISHIMA, 2000), pois existem fatores determinantes, tais como a incidência solar, a evapotranspiração variável e específica para diferentes espécies, como também a sazonalidade do ambiente e o próprio estado de manutenção em que encontram-se os canais, variável esta que este trabalho também pretende aprimorar. Esta frequência de irrigação do cultivo, em conjunto com o alto custo de implantação de um sistema hidropônico, se comparado em um caso como o de alface ao cultivo simplificado de solo arado (SILVA EDUARDO TEIXEIRA. SCHWONKA, 2001), estimula o desenvolvimento de pesquisas e tecnologias a fim de reduzir os investimentos, principalmente em custos variáveis como energia elétrica e solução nutritiva.

Destaca-se que o gasto com eletricidade corresponde isoladamente a aproximados 20% do total dos custos variáveis da produção hidropônica (AITA A.; LONDERO, 2000),

portanto, existe a preocupação com a redução do consumo de energia. Assim, se o intervalo entre os tempos de funcionamento da motobomba pode ser ampliado, este intervalo em atuação caracteriza um mal aproveitamento da energia elétrica.

Citando em sentido filosófico a segunda lei da termodinâmica, enunciada de maneira diversa por grandiosos físicos, nenhum processo é possível sem que hajam perdas. Embora o cultivo seja realizado em ambiente protegido, o sistema hidropônico NFT está sujeito a diversas avarias do seu bom funcionamento. Entre elas, a ocorrência de obstruções, em caso geral, a incorreta fluidez da solução nutritiva, levam a perdas que dificilmente são detectadas de maneira preventiva, dado o elevado número de dutos apresentados em cultivos de maior escala. Portanto, mesmo com o monitoramento frequente e presencial do agricultor, tais falhas acabam por se revelar somente após a apresentação de sintomas nas plantas, dada a dificuldade de recuperação para este tipo de cultivo, estas plantas acabam sendo descartadas (CARRIJO O. A.; MAKISHIMA, 2000).

1.1 Motivação

A idealização desta proposta tem como motivação inicial o interesse de aprendizado por este método de agricultura. O método de hidroponia NFT pode ser empregado em uma diversidade de locais onde não se considerava a possibilidade de plantio, desde que se manipule o ambiente de forma adequada. Portanto, surge o interesse neste conceito dado a grande gama de aplicações e às suas possibilidades de aprimoramento.

Outra fonte de inspiração para esta proposta é a inserção na esfera agrotecnológica. A agricultura é um dos pilares do que se considera o início da civilização humana, símbolo de prosperidade e da própria tecnologia. Dado a capacidade computacional que se tem disponível, aliados ao surgimento de sistemas de baixo custo monetário, mais possibilidades de estudo e aplicações tornam-se possíveis para este ramo.

Por fim, a principal motivação para este trabalho foi a vontade de aprender e manipular sistemas embarcados. Computadores potentes e miniaturizados como o RaspberryPi estão sendo oferecidos a preços tão baixos como 5 dólares, e assim sendo, é de grande interesse o aprendizado quanto à utilização desta ferramenta nas suas mais diversas formas possíveis.

1.2 Objetivos

Utilizando *softwares opensource* gratuitos, assim como o *hardware* e o engenho de ferramentas de baixo custo, o objetivo deste trabalho é apresentar uma proposta economicamente rentável para o monitoramento remoto de um cultivo hidropônico de fluxo laminar.

1.3 Relevância

Mesmo com a elevada eficiência apresentada pela agricultura hidropônica(CARRIJO O. A.; MAKISHIMA, 2000), existe um grande interesse em minimizar as perdas causadas por um fluxo defectivo de nutrientes, como também reduzir o consumo de eletricidade para o seu funcionamento.

A justificativa para a escolha desta solução por um sistema embarcado se dá pela capacidade de *hardware* que o microcontrolador possui, em conjunto com a disponibilidade de acesso via internet. Se corretamente executado, o agricultor tem a possibilidade de verificar o estado de funcionamento de seu cultivo, bem como determinar ou automatizar a escolha do intervalo de acionamento da motobomba via computador ou até mesmo por celular.

1.4 Organização do Trabalho

Este trabalho está distribuído em 5 capítulos, em junção com esta introdução, dispostos conforme a descrição que segue:

Capítulo 2: Será apresentado uma revisão teórica necessária para a compreensão do que se propõe executar com o material disponível.

Capítulo 3: Dispõe os itens necessários para a execução da proposta visando o dimensionamento orçamentário de um projeto completo. Como também, a orientação adequada para seu correto funcionamento.

Capítulo 4: Apresenta os resultados obtidos conforme o que fora desenvolvido.

Capítulo 5: Discute-se a aplicabilidade e promove uma análise geral da solução proposta.

2 FUNDAMENTAÇÃO TEÓRICA

A hidroponia não é um método de agricultura recente, Resh(RESH, 1997), em seu trabalho pioneiro na compilação e divulgação deste tipo de cultivo, cita sua utilização nos famosos jardins suspensos da Babilônia. Em uma análise histórica, compreende-se que a cultura de vegetais com raízes flutuantes em água está presente também nas civilizações Astecas e Chinesas, consideradas como o primórdio da Hidroponia.

Já para o recente século, o cultivo hidropônico se tornou uma ótima opção para o abastecimento da sempre crescente demanda populacional. Barros(BARROS, 1970), desenvolve o tema da lavoura hidropônica em estufas, fator principal para a sua consolidação, pois permite o plantio durante todo o período anual. Os avanços científicos desta área andam intimamente ligados com a inclusão de dispositivos computacionais para o controle e quantificação.

A utilização de sistemas embarcados para controle e monitoramento deste tipo de cultivo é um advento recente neste ramo da agroindústria, que só se tornou possível por conta dos grandes avanços tecnológicos e a drástica redução de preço do hardware.(ANDRADE, Acesso em 7 de Outubro de 2018)

2.1 Hidroponia por Filme de Nutrientes

A Técnica de Filme de Nutrientes, ou NFT, é um sistema popular e versátil de hidroponia(NETO EGÍDIO B.; BARRETO, 2011). O sistema utiliza uma bomba para fornecer água fertilizada para a bandeja de cultivo e um tubo de drenagem para reciclar a solução de nutrientes não utilizada. A bandeja de crescimento é colocada em um ângulo para permitir que a água flua para baixo em direção ao tubo de drenagem, e o fluxo com solução nutritiva é constantemente bombeado para a extremidade alta do tubo.

A solução de nutrientes flui em um filme fino sobre as raízes, garantindo que elas são regadas e alimentadas, mas deve se impedir que sejam completamente embebidas. O filme de água fino garante que a parte superior das raízes permaneça seca e que tenha acesso ao oxigênio no ar.

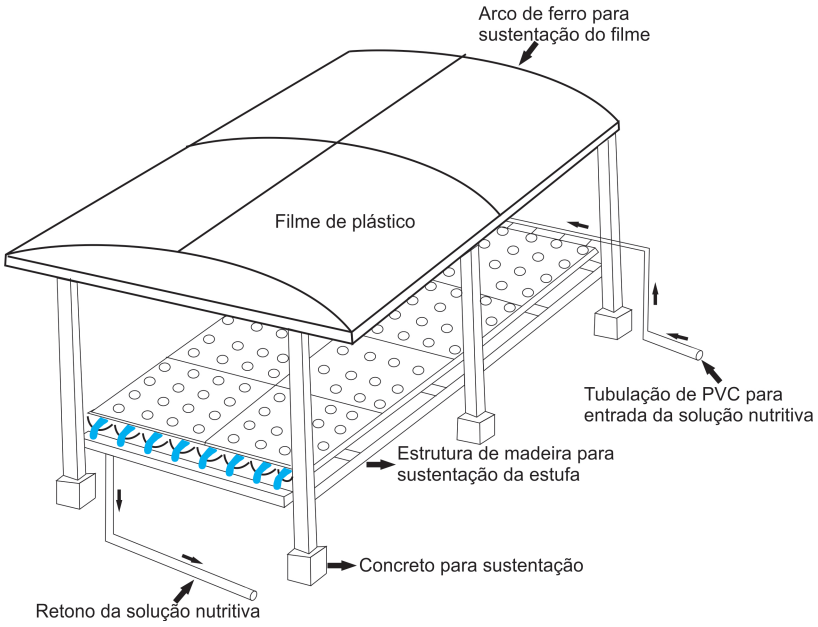

Figura 2: Modelo de cultivo NFT em estufa. Fonte: (VENTURINI, 2014).

A Técnica do Filme Nutriente funciona melhor para plantas de crescimento rápido que não requerem extensa firmeza de suporte. As raízes não são suspensas em um meio neste sistema, dessa forma, elas não conseguem lidar com o peso de uma planta de alta intensidade, portanto, uma planta hidropônica dificilmente alcança proporções de vegetais encontradas no cultivo em solo.

Podemos citar das vantagens de NFT(CARRIJO O. A.; MAKISHIMA, 2000):

- Menor consumo de água e nutrientes;
- Evita os problemas de fornecimento, eliminação e custo associados aos sistemas baseados em solo;
- Relativamente fácil de desinfectar raízes e realizar manutenção em comparação com outros tipos de sistema;
- A planta inteiramente exposta facilita a inspeção para detectar sinais de doença, adequação alimentar, etc;
- A Alimentação regular evita a acumulação de sal localizada na zona da raiz e contribui para manter o pH e a condutividade normais;
- Ambientalmente amigável - potencial mínimo para a contaminação local das águas subterrâneas;

E das desvantagens(CARRIJO O. A.; MAKISHIMA, 2000):

- A falha da bomba pode resultar em morte de plantas dentro de algumas horas, especialmente em clima quente;
- Requer cobertura completa de todos os sistemas de tubulação por onde a solução de nutrientes flui. O crescimento de algas danosas bem como a obstrução dos canos podem ocorrer mesmo com pouca exposição;
- Não é adequado para plantas com sistemas de raiz grande (por exemplo, cenouras);
- Comparado com a lavra irrigada em solo, é menos adequado para águas salinas (Como a de abastecimento de locais litorâneos), porque a salinidade da água recirculante aumentará gradualmente;

2.1.1 Controle e Temporização

De modo geral, a escolha das frequências de irrigações no sistema hidropônico depende das características ambientais, especialmente da intensidade luminosa e da temperatura do ar, do meio de cultivo e da fisiologia da planta (ANDRIOLO, 1999).

Existem diversos fatores que compõem a verificação do bom estado da lavra. A tabela 1 destaca os principais elementos a serem analisados por ordem de prioridade. Inclui-se que todas as variáveis de análise ou utilizam dispositivos eletrônicos para sua quantificação ou podem ser analisadas por equipamentos de comunicação digital, o que traz para esta forma de cultura um grande interesse no desenvolvimento de soluções unificadas por sistemas integrados.

O alto custo do sistema hidropônico tem estimulado a condução de novas pesquisas, especialmente a fim de reduzir os custos variáveis do sistema. Gastos com energia elétrica e solução nutritiva podem chegar a 42% do total dos custos variáveis (AITA A.; LONDERO, 2000). Estima-se que os custos da energia elétrica e da solução nutritiva esteja em 19,7 e 9,2% do custo variável total, respectivamente(AITA A.; LONDERO, 2000). O aumento do preço desse insumo tem sido um fator negativo para a expansão da hidroponia, visto que o sistema de irrigação é dependente da eletricidade.

Para determinar a frequência de irrigação no cultivo, adota-se a nomenclatura de intervalos seguidos de 15 minutos de fluxo laminar contínuo. Do material acadêmico disponível, são padronizados e descritos como T15, T30, T45 e T60, para as principais escolhas, identificando o tempo pela qual o fluxo deve permanecer estático em termos de minutos para os períodos diurnos. E TN2 e TN3, identificando o tempo de intervalo em termos de horas para os períodos noturnos. (PILAU, 2002)

A implantação destes intervalos é objetivo de muitos estudos relacionados ao aprimoramento, como também do próprio agricultor em encontrar o equilíbrio adequado

Tabela 1: Elementos de análise de um cultivo hidropônico NFT por ordem de prioridade. Fonte: (CARRIJO O. A.; MAKISHIMA, 2000).

Prioridade	**Variável**	**Análise**
Alta	Ph - Condição ácida ou Básica	Controle da Qualidade da Água
Alta	Temperatura	O PH é medido em função da temperatura, deve ser monitorada em faixas de temperatura ideal
Alta	Condutividade Elétrica	Controle do nível de nutrientes (sais minerais) fornecidos à planta
Alta	Fluxo	A solução nutritiva a ser fornecida em um fluxo contínuo periódico
Média	Oxigenação da Água	O consumo de oxigênio a ser suprido conforme o fluxo de fornecimento de nutrientes
Média	Sistemas de Indicadores	As variáveis devem ser explícitas em intervalos ou através de indicador para verificação ao operador
Baixa	Controle de Volume do Reservatório	Deve ser constante para não acarretar na falta de água ao sistema, no sistema convencional inserido uma quantidade extra para suprimir as perdas.

entre a redução de custos e a saúde da safra. Para o fim que se pretende apresentar nesta proposta, a possibilidade de seleção deste intervalo será retratada em favor da análise do potencial de economia de energia do sistema.

2.2 Sistemas Embarcados

Um sistema embarcado é um sistema de propósito especial no qual o computador está completamente encapsulado pelo dispositivo que ele controla. Ao contrário de um computador de propósito geral, como um computador pessoal, um sistema embarcado executa tarefas pré-definidas, geralmente com requisitos muito específicos. Uma vez que o sistema é dimensionado, integra-se todos os elementos necessários para o seu funcionamento, reduzindo o tamanho e o custo do produto. Este tipo de hardware é geralmente produzido em massa, de modo que leva a uma significativa economia de custo.

Alguns exemplos de sistemas embarcados incluem caixas eletrônicos, celulares, impressoras, termostatos, calculadoras e consoles de videogames. Computadores portáteis também são considerados sistemas embarcados devido à natureza do seu design de hardware, mesmo que eles sejam mais expansíveis em termos de software. A definição deste tipo de sistema tende a ser cada vez mais obtusa à medida que os dispositivos se expandem.

2.2.1 Arquitetura ARM

A arquitetura ARM, originalmente Acorn Risc Machine e posteriormente Advanced RISC Machine, é projetada e licenciada pela ARM Holdings,(ARM, Acesso em 13 de Outubro de 2018) com sede em Cambridge (UK). Esta arquitetura de microprocessadores é encontrada em diversos dispositivos de diferentes áreas, tais como: automotiva, médica, infraestrutura, IoT (Internet das Coisas), dispositivos móveis, casas inteligentes e acessórios.

Os processadores ARM são famosos pelo seu elevado desempenho a troco de baixo consumo de potência,(SLOSS A.; SYMES, 2004) o que os torna ideal para uso em sistemas embarcados e dispositivos móveis (i.e celulares, tablets, smart TVs, dentre outros). Com o crescimento da procura por tais dispositivos, a arquitetura ARM se tornou uma gigante fornecedora de arquiteturas no mundo. Só em 2017, foram produzidos 16,7 bilhões de chips ARM.(ARM, Acesso em 13 de Outubro de 2018) Em 2010, o número de chips ARM fabricados foi aproximadamente 20 vezes superior ao de chips da concorrente Intel.(ARM, Acesso em 13 de Outubro de 2018)

A arquitetura ARM é comumente classificada como do tipo RISC, mas apresenta algumas características especiais que fogem da filosofia de um RISC puro (é raro o emprego de uma arquitetura RISC ou CISC pura, sempre existem pequenas conjunções a fim de melhorar características desejadas).

2.2.2 Linux Embarcado

O sistema operacional Linux foi criado em 1991 por Linus Benedict Torvalds, na Universidade de Helsinki. Seu modelo de desenvolvimento aberto e sua publicação via GNU GPL (General Public License) atraíram milhares de contribuidores ao redor do mundo.(RAGHAVAN, 2006) Graças a essa licença, o código fonte do kernel linux é integralmente disponível para uso comercial ou pessoal. Diz-se que a origem da revolução Linux se deu com um e-mail de Linus, para uma lista de e-mail de desenvolvedores de minix(RAGHAVAN, 2006).

A chegada do Linux aos sistemas embarcados se deu em 1996, com uma pesquisa liderada por Michael Barabanov e Victor Yodaiken. A pesquisa se baseava em usar um

pequeno kernel de tempo real em parceria com Linux, para garantir resposta em tempo real.(RAGHAVAN, 2006) Desde então, vários pesquisadores continuaram o desenvolvimento de sistemas embarcados utilizando Linux, e várias empresas adotaram o Linux embarcado em suas linhas de produto. Como exemplo dessa grande difusão, podemos citar o Raspbian e Arch Linux ARM, ambos distribuições de Linux usadas na placa Raspberry Pi.

3 DESENVOLVIMENTO DO PROJETO

3.1 Materiais

3.1.1 RaspberryPi 3 Modelo B

O Raspberry Pi é uma série de pequenos computadores de placa única desenvolvidos no Reino Unido, pela Fundação Raspberry Pi, para promover o ensino de informática básica nas escolas e nos países em desenvolvimento. O modelo original tornou-se muito mais popular do que o previsto, sendo utilizado para várias aplicações fora de seu mercado-alvo, para usos como a robótica e até na composição de supercomputadores paralelos (ALMEIDA, 2015). Os periféricos (incluindo teclados, mouse, caixas e fonte de alimentação) não estão incluídos na compra de um Raspberry Pi singular. Entretanto, alguns acessórios foram incluídos em vários pacotes oficiais e não oficiais.(RPIFOUNDATION, Acesso em 8 de Outubro de 2018)

O Raspberry Pi 3 é a terceira geração desse dispositivo, e é encontrada nacionalmente a um custo de R$ 165,00. Ou ainda oferecido diversos tipos de kits com dissipadores e elementos adicionais por uma faixa de R$ 220,00 (MERCADOLIVRE, Acesso em 10 de Outubro de 2018). A seguir, uma lista de suas especificações.

- Quad Core 1.2GHz Broadcom BCM2837 64bit CPU (Arquitetura ARM)
- 1GB RAM
- BCM43438 wireless LAN e Bluetooth Low Energy (BLE) on board
- 40-pinos de portas de entrada e saída de uso geral
- 4 portas USB 2
- Full size HDMI
- Porta CSI câmera
- Porta DSI display
- Porta Micro SD
- Fonte Micro USB (até 2.5A)

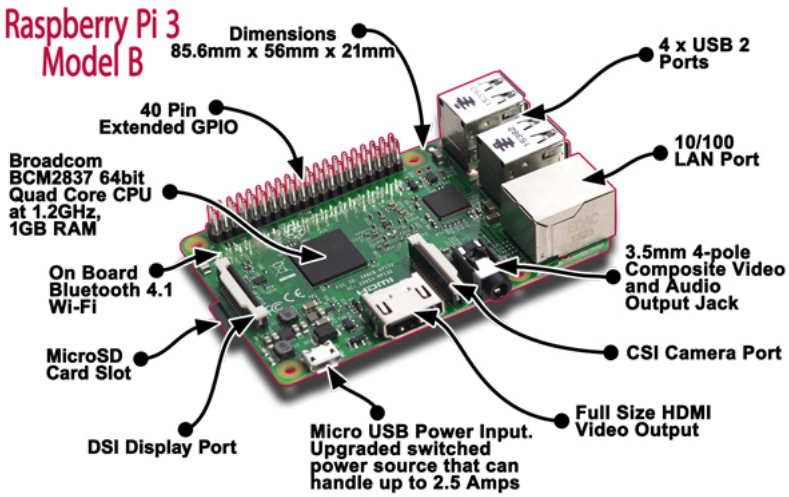

Figura 3: Estrutura e Componentes da RaspberryPi3 Modelo B. Fonte:(RPIFOUNDATION, Acesso em 8 de Outubro de 2018).

O sistema operacional do RaspberryPi roda através de uma pré programação de um cartão SD. Esse cartão servirá para carregar o sistema operacional e dados de armazenamento. Geralmente, este cartão não está incluso no custo inicial da placa, tornando interessante à usuários iniciantes adquirir o kit com um novo cartão incluso, ou com um cartão pré programado.

3.1.2 Arch Linux ARM

Arch Linux ARM é uma portabilidade do Arch Linux para processadores ARM. Sua filosofia de design é "simplicidade e controle total para o usuário final". Assim como o sistema operacional pela qual teve origem, o Arch Linux para desktop, ele fora desenvolvido para ser muito parecido com o Unix. Este objetivo de minimalismo e controle de usuário completo, no entanto, pode tornar o Arch Linux difícil para iniciantes do Linux, pois exige mais conhecimento e responsabilidade pelo sistema operacional.(ARM, Acesso em 8 de Outubro de 2018)

O Arch linux é uma das distribuições de Linux que mais oferece liberdade de customização ao usuário. O sistema vem apenas com os componentes mais básicos, ficando a cargo do usuário a escolha e instalação dos pacotes necessários. Isto faz com que o Arch Linux ARM se apresente como um sistema extremamente leve, ágil e de baixa carga processual, fator importantíssimo no desenvolvimento de sistemas embarcados.

Figura 4: Logo Arch Linux ARM. Fonte: (ARM, Acesso em 8 de Outubro de 2018).

3.1.3 Dispositivos de Identificação

Para poder identificar o fluxo nos dutos de hidroponia, foram utilizados sensores de nível de água de baixo custo. Este tipo de dispositivo pode ser encontrado em diversos designs, com seu funcionamento sendo semelhante para a grande maioria dos casos.

O sensor utilizado é feito de material plástico, composto de uma haste feita de material flutuante na qual desliza uma dobradiça. Esta haste possui um ímã que aciona um sensor magnético na base do dispositivo, que por sua vez fecha o contato dos 2 fios que saem do sensor.

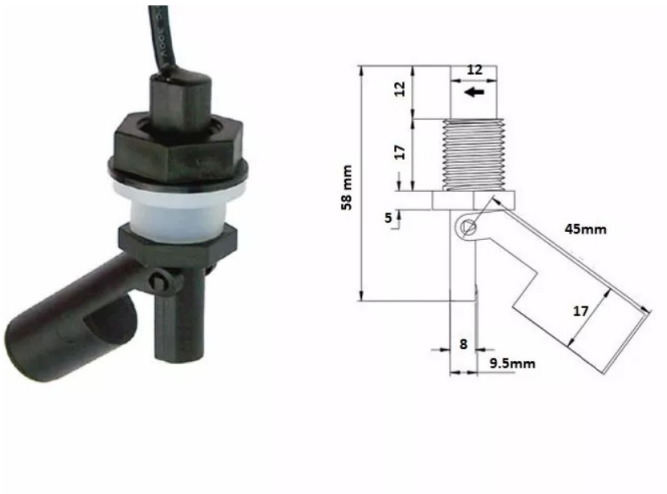

Figura 5: Sensor de Nível Utilizado. Fonte: (MERCADOLIVRE, Acesso em 10 de Outubro de 2018).

O sensor de nível funciona como uma chave liga-desliga que pode acionar chaves, bombas, lâmpadas ou enviar um sinal para o microcontrolador como o Arduino, Pic ou Raspberry Pi.

3.1.4 Dispositivos de Controle

3.1.4.1 Módulo Relé 5V

O módulo relé 5V é utilizado para controlar cargas como lâmpadas, motores, fechaduras e eletrodomésticos, desde que a corrente de operação não ultrapasse 10 A (ampéres).

Acionar o relé é muito simples, para isso, basta um dispositivo enviar um sinal de borda para mudar o estado das portas digitais ligadas aos pinos IN1 e IN2. Um detalhe importante desse módulo, é que os relés são ativados em nível baixo, ou seja, quando o estado da porta estiver em LOW, o relé será acionado.

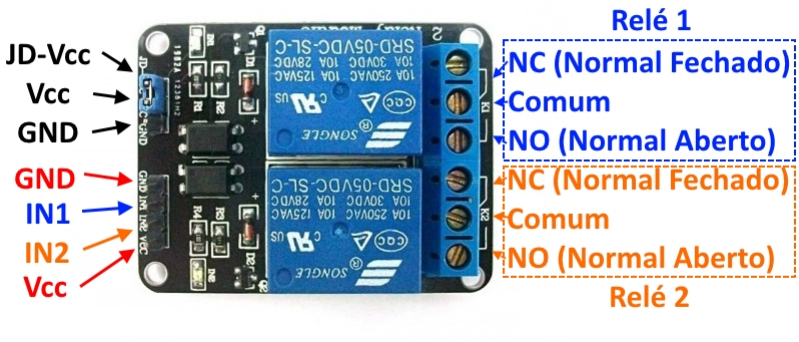

Figura 6: Módulo Relé Utilizado. Fonte: (MERCADOLIVRE, Acesso em 10 de Outubro de 2018).

Cada relé desse módulo suporta cargas de até 10 A, em 125 VAC, 250 VAC ou 30 VDC. Leds indicadores mostram o estado do relé (ligado/desligado) em cada canal. O módulo já contém todo o circuito de proteção para evitar danos ao microcontrolador, e possui baixa corrente de operação.

3.1.4.2 Multiplexadores

Na eletrônica, um multiplexador (ou MUX) é um dispositivo que seleciona um dos vários sinais de entrada analógicos ou digitais e encaminha a entrada selecionada para uma única linha. É um dispositivo comumente utilizado em todo ramo da eletrônica, sua relevância nessa proposta se dá por permitir que uma única entrada lógica de identificação possa receber informação de diversas chaves de sensores de nível, tornando possível o dimensionamento para sistemas de 100 tubulações ou mais.

O MUX utilizado para o desenvolvimento do protótipo foi o HC4051. O HC4051 é um multiplexador analógico de 8 canais com três entradas de endereço (A, B e C), uma entrada de habilitação BAIXA ativa (In), oito entradas / saídas independentes (IO0 a IO7) e uma entrada / saída comum (Z).

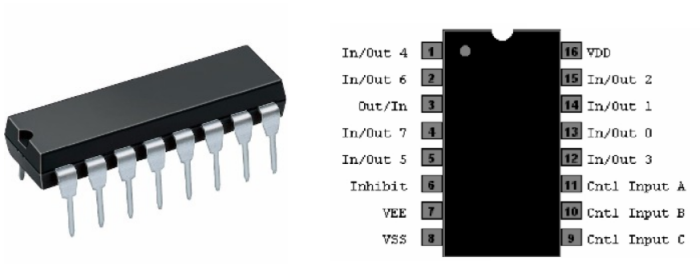

Figura 7: MUX HC4051 com a respectiva pinagem. Fonte: (FILIPEFLOP, Acesso em 10 de Outubro de 2018).

3.2 Métodos

Aqui serão apresentados os métodos necessários a fim de que seja possível recriar o protótipo elaborado para a identificação e controle dos dutos hidropônicos.

3.2.1 Instalação e Configuração

A fundação Arch Linux ARM oferece um tutorial de instalação do sistema operacional. Um importante pré-requisito é o acesso a um computador capaz de ler e alterar um cartão microSD com uma distribuição Linux previamente instalada.

A partir deste computador, basta inserir o cartão microSD, identificá-lo e seguir os seguintes passos em um terminal(ARM, Acesso em 8 de Outubro de 2018):

1. Abra o fdisk para particionar o micro SD (troque sdX pelo nome do dispositivo que aparecer no computador): fdisk /dev/sdx
2. No prompt do fdisk delete qualquer partição anterior e crie uma nova:
   - **a.** Digite o para deletar qualquer partição já existente.
   - **b.** Digite p para listar as partições e confirme que não há nenhuma.
   - **c.** Digite n, depois p para primário, 1 para selecionar a primeira partição, pressione ENTER para aceitar a configuração padrão e por fim digite +100M para o último setor.
   - **d.** Digite t, depois c para selecionar o tipo da primeira partição como W95 FAT32 (LBA).

e. Digite n, depois p para primário, 2 para selecionar a primeira partição, pressione ENTER duas vezes seguidas para aceitar as configurações padrões.

f. Escreva na memória as novas partições e saída do fdisk digitando w.

3. Crie monte o sistema de arquivos FAT32: 32 mkfs.vfat /dev/sdX1 (primeira partição do micro SD) mkdir boot mount /dev/sdX1 boot

4. Crie e monte o sistema de arquivos ext4: mkfs.ext4 /dev/sdX2 (segunda partição do micro SD) mkdir root mount /dev/sdX2 root

5. Faça o download e extraia o sistema de arquivos root do Arch Linux (esse comando precisa ser executado como root do terminal, e não via sudo): wget http://os.archlinuxarm.org/os/ArchLinu rpi-2-latest.tar.gz bsdtar -xpf ArchLinuxARM-rpi-2-latest.tar.gz -C root sync

6. Desmonte as duas partições: unmount boot root

7. Insira o micro SD no Raspberry Pi 3 e conecte ela à rede de energia.

A partir deste ponto, o Raspberry já deve inicializar o sistema operacional, devendo ser configurado via um monitor e teclado.

3.2.1.1 Configuração para Acesso Remoto

Uma ferramenta importante para a programação remota do RaspberryPi é o SSH, que permitirá ao usuário implementar códigos e alterações sem que haja a necessidade de um acesso físico.

No passo a passo de um possível primeiro acesso, será solicitado um usuário e senha. Os valores pré definidos pela instalação são "alarm"e "alarm".

Com a credencial liberada, os comandos necessários para habilitar o funcionamento do SSH são os seguintes:

```
systemctl start systemd-networkd.service
systemctl start systemd-resolved.service
```

Desta maneira, o sistema inicializará com SSH habilitado. Com auxílio do editor de texto *nano*, presente na instalação básica do Arch Linux, edite o arquivo presente na pasta do SSH da seguinte maneira:

```
nano /etc/systemd/network/eth0
```

E altere os valores presentes para:

```
[Match]
Name=eth0
[Network]
Address = 192.168.0.110/24
Gateway = 192.168.0.110
```

Apertando "Ctrl+w"para sair e salvar. O IP exemplificado é uma sugestão de valor para o IP fixo dentro do local de rede pela qual o RaspberryPi está conectado, seu valor deve ser verificado para caso já esteja em uso por outro dispositivo, respeitando o protocolo interno do roteador. (exemplo: alterar somente o último número pontuado, para 192.168.0.67)

Existe a possibilidade desta configuração não corresponder aos valores do roteador, para este caso, consulte as informações fornecidas pelo fabricante do equipamento via selos ou manual.

A porta padrão de conexão ao SSH é a 22, que deve ser preferencialmente alterada por motivos de segurança. Para o exemplo em questão, foi escolhida a porta 8085, que pode ser estabelecida da seguinte maneira:

```
nano etc/ssh/sshd_config
```

Procure a inscrição comentada como #Port 22 e altere para:

```
Port 8085
```

Removendo o "#".

3.2.1.2 Configuração de Roteador Doméstico

Também é necessário a configuração do encaminhamento de porta do roteador, que terá de ser estabelecida por meio de um computador pessoal com acesso ao equipamento.

Seguindo os seguintes passos:

1. No computador, abra um navegador web e digite o IP interno do roteador (exemplo: 192.168.0.1)
2. Geralmente os valores padrões de *login* e senha estão disponíveis em um selo colado no próprio dispositivo.

3. Será apresentado o firmware de controle, dentre as opções avançadas, encontre o item de encaminhamento de porta, ou *Port Fowarding*.

4. Dois encaminhamentos precisarão ser adicionados, estabeleça a configuração do primeiro com os seguintes valores:

   ```
   Description: SSH RasPi
   Protocol: Both
   Application & Port: User Defined
   Local: 192.168.0.110 8085/8085
   External: 0.0.0.0 22/22
   Activate: YES
   ```

   Que será a configuração para o acesso em SSH. Em seguida, entre com a segunda entrada para permitir o acesso ao webserver que será implementado:

   ```
   Description: WebServer
   Protocol: Both
   Application & Port: User Defined
   Local: 192.168.0.110 5000/5000
   External: 0.0.0.0 8000/8000
   Activate: YES
   ```

5. Ainda dentro do acesso do roteador, identifique o IP pela qual ele está designado nas configurações básicas. No caso exemplificado, o IP doméstico era de 177.180.188.165. Este valor será obrigatoriamente diferente para qualquer outro equipamento conectado a internet e pode alterar com certa frequência dependendo do provedor de internet.

6. Reinicie o roteador

   Reinicie o RaspberryPi através do comando

```
su
alarm
reboot
```

Para acessar a máquina de outro computador com uma distribuição Linux, digite:

```
ssh alarm@xxx.xxx.xxx.xxx
```

Aonde "xxx.xxx.xxx.xxx"é o IP local do roteador.

3.2.1.3 Python 3

Tendo configurado a placa para acesso remoto, faz-se desnecessário o uso do cabo HDMI em conjunto com um monitor, nem do teclado para sua programação.

A distribuição não acompanha o pacote *sudo*, que permite aos usuários cadastrados realizarem tarefas de permissões especiais. A desvantagem é que, até então, todos os comandos que exigem permissão especial do sistemas tem de ser executados através do *root*, o que é perigoso. Por isso, é de grande interesse a instalação do pacote *sudo*. A instalação é bem simples, basta digitar no terminal:

```
pacman -S sudo
```

O próximo passo é instalar os pacotes e programas necessários para a construção do projeto. O primeiro deles é a instalação do compilador de Python. Ele pode ser obtido no repositório oficial do Arch Linux, podendo ser obtido através do gerenciador de pacotes pacman. Para tal, basta digitar no terminal:

```
sudo pacman -S python
```

Para facilitar a inclusão de pacotes e bibliotecas em *python*, que são requisitos para o funcionamento dos códigos, instalou-se o gerente de pacotes do próprio *python*, chamado de pip:

```
sudo pacman -S pip
```

3.2.2 Configuração da Placa Multiplexada

O protótipo para identificação de fluxo fez uso de somente 3 saídas do MUX, que deverá ser conectada ao RaspberryPi através de uma placa soldada, ou com o auxílio de uma placa de ensaio com a seguinte configuração:

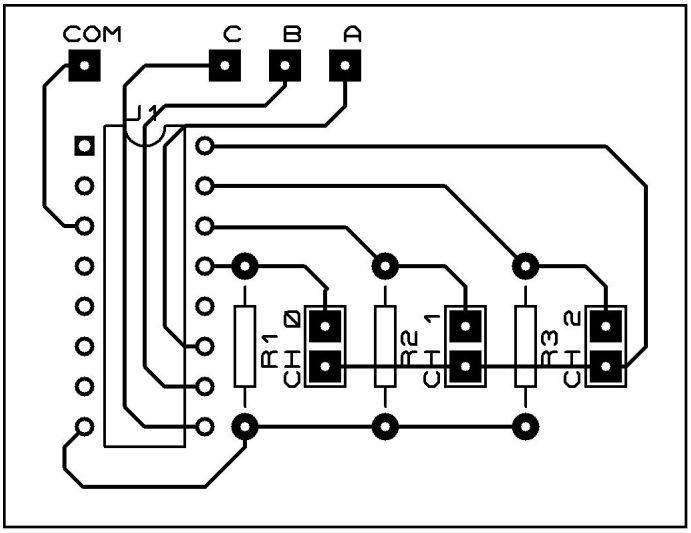

Figura 8: Circuito de ligação Raspberry-Multiplexador-Relé-Chaves.

3.2.3 Instalação do Protótipo

Foi construído em garrafa PET uma peça elaborada de forma que pudesse permitir um fluxo de água inferior ao presente na saída da tubulação e que também pudesse deixar fluir o restante de água represada. A figura a seguir detalha o que foi utilizado:

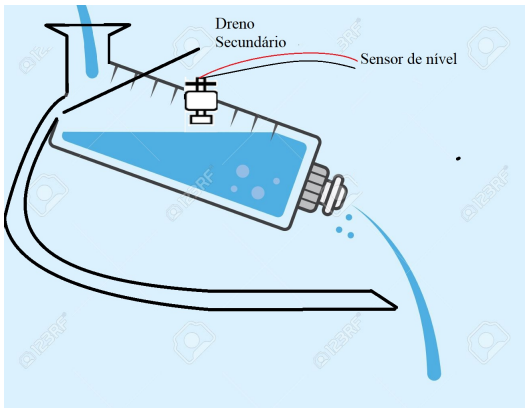

Figura 9: Protótipo para identificação de fluxo em dutos hidropônicos.

Esta peça deverá ser encaixada nas saídas dos dutos de forma que o sistema final tenha a seguinte característica:

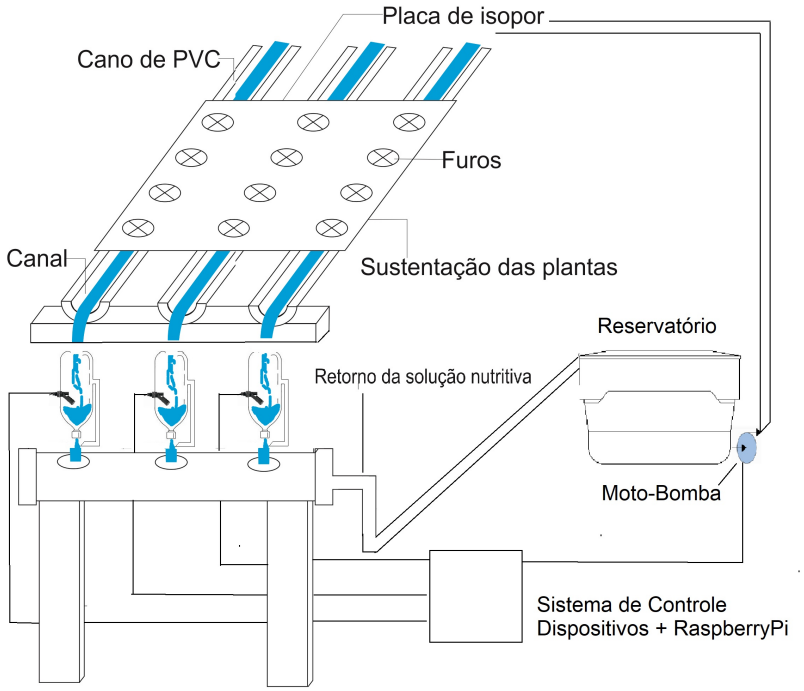

Figura 10: Sistema final prototipado.

Com o acesso ao Raspberry, foi clonado o repositório do sistema online *github* do autor, que continham todos os códigos desenvolvidos para o funcionamento do sistema.

Os comandos para instalação do *git* e o download dos códigos são:

```
sudo pacman -S git
git clone https://github.com/lawrencerb/Raspi-Hidromonitor.git
```

Acessando o repositório copiado no Raspberry, foi feito o download de todos os requisitos dos códigos através do comando:

```
sudo pip install -r requirements.txt
```

Que contém a lista de todas as bibliotecas pertinentes à execução do sistema.

Por fim, executou-se o arquivo *run*, que inicializará o sistema de controle:

```
sudo ./run
```

O sistema então pôde ser acessado via qualquer computador conectado à internet digitando no navegador o endereço xxx.xxx.xxx.xxx:8000, onde xxx.xxx.xxx.xxx é o IP da rede onde o Raspberry está conectado. Entrando com o login e senha *admin* e *admin*.

Tabela 2: Orçamento de um projeto para 96 tubulações. Fontes: (ALIEXPRESS, Acesso em 10 de Outubro de 2018)(FILIPEFLOP, Acesso em 10 de Outubro de 2018) (MERCADOLIVRE, Acesso em 10 de Outubro de 2018)

Objeto	Quantidade	Custo (R$)	Total(R$)
Raspberry Pi	1	220,00	220,00
Cabos e Conectores	*	125,00	125,00
Dispositivos de Identificação	100	2,96	296,00
Modulos de Relés	4	28,00	112,00
Componentes Eletrônicos	*	143,00	143,00
Placa soldada integrada	1	150,00	150,00
Material de Construção	*	95,00	95,00
Mão de Obra e Custos Adicionais	*	1500,00	1500,00
		TOTAL:	2641,00

3.2.4 Levantamento de Custo e Discussões

Dispondo dos materiais apresentados neste capítulo, foi feito um orçamento do custo de implantação para um sistema real de 96 tubulações. Este montagem seria o limite de entradas lógicas de identificação da GPIO do RaspberryPi no modelo que fora proposto. Se houver necessidade de identificar dutos adicionais, outra solução deverá ser elaborada.

O custo de implantação deste projeto está em torno do que fora dimensionado, restando analisar se é aplicável em termos de economia de recursos. O orçamento está disponível na tabela 2.

Os valores referentes à mão-de-obra e aos custos adicionais foram estimados sem dados empíricos. No que se referem, seriam valores destinados a compra de material para confecção dos microtanques de identificação e de serviços de manipulação dos materiais para implantação do sistema.

A fim de se obter uma resposta útil deste projeto, destaca-se a necessidade absoluta da padronização replicável do protótipo aqui estudado. Por quais meios e recursos, que tornam isto possível, fogem da análise deste trabalho. Portanto, serão debatidos os resultados encontrados com um olhar simplório na precificação.

4 RESULTADOS E DISCUSSÕES

4.1 Resultados

O produto final para o sistema de monitoramento desta proposta conta com um controle de bomba, o estado dos sensores de nível e, opcionalmente, um segmento de *stream* de câmera. A câmera não é um elemento absolutamente necessário, no entanto, ela facilitou a execução dos testes e pode oferecer ao usuário imagens em tempo real de algum ponto do sistema hidropônico. A *webpage* do que fora desenvolvido é ilustrado pela figura:

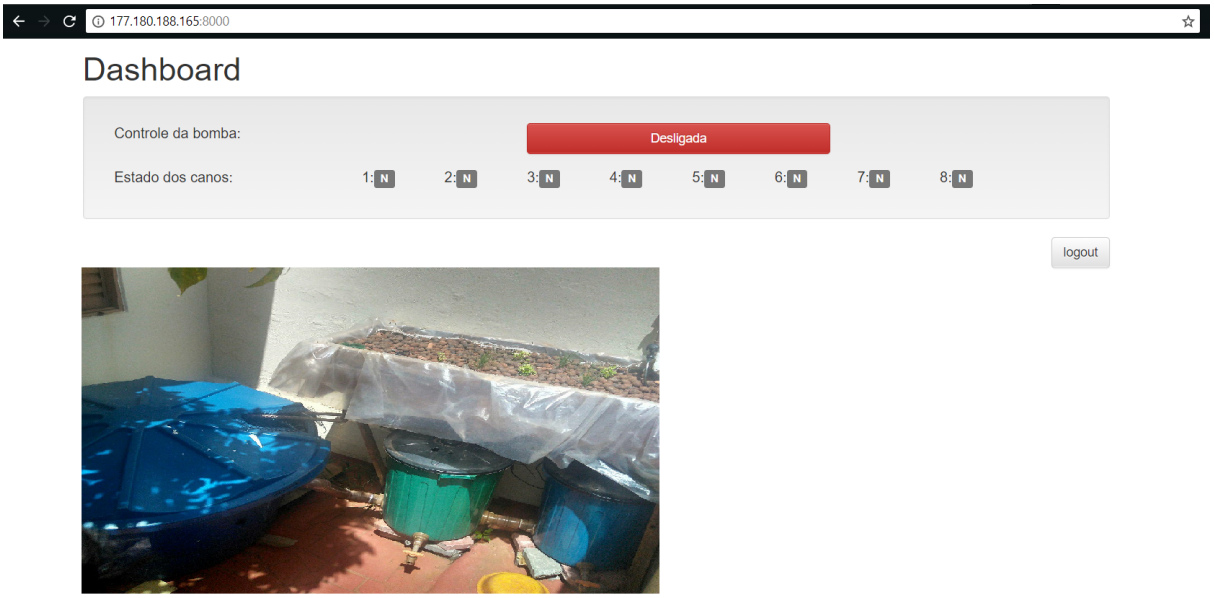

Figura 11: Tela do sistema de monitoramento desenvolvido.

O sistema apresentado para a escrita deste trabalho é um protótipo em testes, isto significa que não foram implementadas as temporizações possíveis para o funcionamento da bomba. Entretanto, um ícone de ilustração desenvolvido em *javascript* permitiu ao usuário acionar, desligar e conferir o estado da bomba. Utilizando uma lâmpada convencional, foram feitos testes com diversos computadores remotamente distantes e, para todos os casos, o sistema respondeu corretamente.

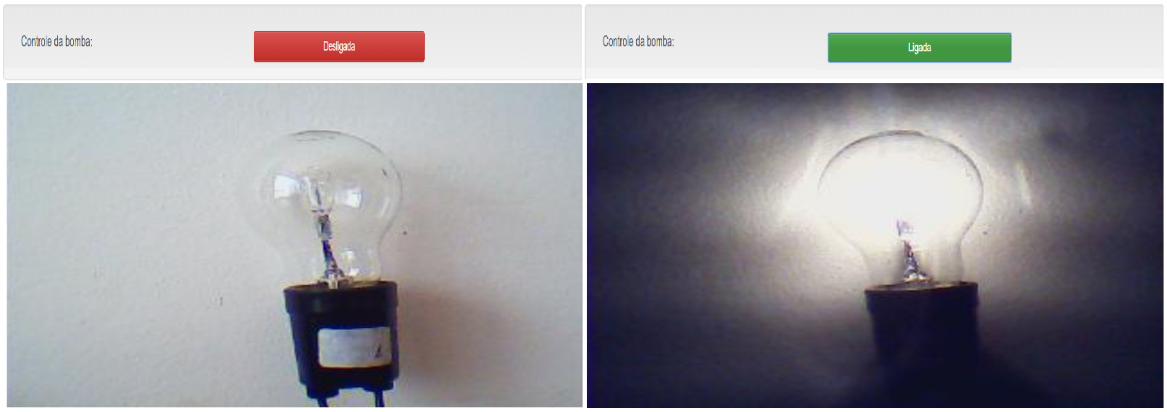

Figura 12: Teste de acionamento da bomba utilizando uma lâmpada convencional.

O servidor via web identificou corretamente o estado das chaves através de uma sinalização por botões. É feita uma varredura das chaves conectadas ao multiplexador e o servidor atualiza as informações via um PUSH de HTML de 2 em 2 segundos. A figura 13 demonstra os testes realizados com as chaves 1 e 3:

Figura 13: Teste de identificação dos sensores de nível 1 e 3.

Uma vez que o protótipo estava em funcionamento, fora programado um ícone HTML próximo à numeração do duto 1 para realizar-se um ensaio de avaria. Este ícone foi pré-configurado para alterar a cor e a descrição para o caso de uma demora maior que 15 segundos entre o acionamento da motobomba e a identificação do duto 1 cheio.

Figura 14: Teste de alerta do duto de nível 1

Este alerta seria o fim para qual esta proposta almeja apresentar. O usuário pode acessar o *webserver* através de qualquer aparelho conectado a internet que possua um

navegador, podendo identificar rapidamente um duto defeituoso apontado por uma demora ou no não acionamento da chave da bóia.

Para o funcionamento completo do sistema, o pico de processamento não ultrapassava os 60% em um núcleo singular, mesmo com múltiplos computadores conectados. O algoritmo em *python* também impede que mais de 2 usuários conectados simultaneamente assistam ao vídeo da câmera, o que diminui o grau de exigência processual. A figura 15 mostra uma colagem do *htop*, ilustrando o consumo total do sistema. O que é um excelente indicativo tanto de baixo consumo energético, quanto para o barateamento do sistema que poderia ser rodado através de um RaspberryPi Zero.

Figura 15: Uso total de CPU em pico do sistema

Três algoritmos foram escritos para o funcionamento da proposta. O primeiro é o de monitoramento das chaves e controle da motobomba, o segundo é a composição do *webserver* tendo partes em HTML, *javascript* e *json*, o último sendo o funcionamento da *webcam* USB.

Um dos grandes desafios foi o desenvolvimento de operações em *multithread*, que é um elemento necessário para a execução de múltiplas atividades simultaneamente. Para tal fim, a execução dos algoritmos separadamente produziu melhores resultados com a biblioteca *Gunicorn*. O *Gunicorn* é baseado no atendimento de modelos pré bifurcação. Isso significa que existirá um processo mestre central que gerencia um conjunto de diversos trabalhos diferentes e independentes. O mestre não comunica com os demais programas, apenas os gerencia e os mantém em atuação. Todos os pedidos e respostas são tratados completamente pelos processos de trabalho, dando liberdade para o programador compor separadamente os programas sem se preocupar com a execução simultânea e comunicação entre eles.

4.2 Aprimoramentos

Conforme fora comentado, o conjunto é uma versão simples para a realização de testes, sendo necessário o desenvolvimento de algoritmos mais sofisticados para que se possa, eficientemente, testar o sistema em um caso real. Destaca-se o advento de algoritmos de aprendizado ou a facilitação ao usuário na configuração tanto nos tempos de acionamento da motobomba quanto na interpretação dos intervalos de tempos da identificação da fluidez dos dutos.

Como um primeiro fator, um sistema hidropônico exige confiabilidade no acionamento da motobomba. O sistema em Arch Linux pode ser configurado para inicializar automaticamente o programa, sem que haja a necessidade de um usuário conectado ao Raspberry. Isto em conjunto com uma pré configuração padrão de T15 no período diurno e de TN2 para o período noturno, e também, uma disposição simplificada para alteração destes valores, seriam elementos exigidos para realização de testes reais.

Outro elemento que se deve notar, é que o código foi escrito para analisar somente 8 saídas multiplexadas, embora fora dimensionado o custo de implantação do projeto em uma cultivo de 96 canais, o *script* precisa ser ampliado para analisar efetivamente todos as tubulações de um caso real. Há de se adicionar para o caso de muitas saídas, onde é considerável o advento de microcontroladores simples, encarregados dos gerenciamento do estado das chaves, o que aliviaria o número de saídas utilizadas no Raspberry, permitindo a conexão de outros instrumentos.

Identificar o bom estado dos canos não poderia apenas estar resumido em estados binários. A partir da diferença do tempo em que a motobomba é acionada, e que a chave é corretamente identificada como fechada, o programa pode armazenar e aprender este tempo a fim de que se possa identificar possíveis fluxos deficientes. Estes tempos seriam únicos para cada saída singular da tubulação, onde o usuário teria um acionamento do modo de treinamento que armazenaria estes momentos adicionando uma faixa de segurança, e informa ao usuário para o caso de uma demora excessiva, ou até do não acionamento, que configura uma avaria no sistema de alimentação das plantas.

A partir de um sistema embarcado como o Raspberry, instrumentação adicional como sondas de condutividade, pHmetros, termômetros e sensores de luminosidade poderiam compor um sistema de monitoramento completo para hidroponia, facilitando o controle e melhorando a eficiência do cultivo.

5 CONCLUSÕES

A proposta teve êxito em demonstrar a possibilidade de controle e inspeção remotos de dispositivos conectados a um sistema embarcado via a criação de um micro servidor de gerenciamento. Uma plantação hidropônica comercial faz uso de uma área consideravelmente grande, sua inspeção é uma tarefa trabalhosa que ainda não possui uma solução genericamente aceita e difundida, principalmente para os períodos noturnos, onde operários e agricultores não costumam estar presentes.

A plataforma para desenvolvimento do protótipo, o RaspberryPi 3 Modelo B, foi escolhida por já haver possessão prévia do autor. No entanto, foi constatado que muitos dos recursos adicionais presentes neste dispositivo eram dispensáveis, permitindo a escolha de um *hardware* de custo ainda menor, o que passa a ser um atrativo na composição de um produto para o mercado.

O sistema operacional Arch Linux ARM demonstrou ser rápido e eficiente, abafando o questionamento da necessidade de se compor um SO dedicado para a aplicação. Sua inicialização leva menos de 10 segundos e ele permite controle total sobre inicializações e execuções internas, o que pode torná-lo ainda mais eficiente no gerenciamento e aprimoramento para os fins de monitoramento de dispositivos. Adiciona-se, em conjunto, o comentário de que o SO Arch Linux é robusto e constantemente atualizado, possuindo excelente nível de segurança de operação cobrindo avarias de software e ataques maliciosos.

Uma das questões levantadas no decorrer do desenvolvimento desta proposta é a dispensabilidade de um sistema embarcado na composição do projeto. Claramente, existem trabalhos que detalham a utilização de microcontroladores simples no monitoramento de cultivos hidropônicos(SOUZA, 2010), entretanto, a comodidade de acessar remotamente os identificadores através de um navegador, combinados com o baixo custo destes micro computadores encapsulados, a possibilidade de diversos aprimoramentos, a alta capacidade de processamento em baixo consumo energético e facilidade em replicar a programação são elementos que claramente caracterizam a necessidade desta escolha.

Python foi a linguagem utilizada para a composição do *software*, a linguagem era desconhecida pelo autor até o início deste projeto, e demonstrou por que é considerada como a "lingua do futuro"(ALMEIDA, 2015). Embora tenha uma exigência computacional maior, ela possui extensas bibliotecas e seus códigos são intuitivos e de fácil aprendizagem. Com poucos dias de uso, o desenvolvimento já se mostrava rápido, e qualquer dúvida era rapidamente sanada pela extensa comunidade e material disponível *online*.

Existem obrigatoriedades em elaborações de material físico, como a padronização

dos microtanques e a estrutura de apoio e conexão do sistema, que precisam ser projetadas afim de que se possa realizar um teste de campo para esta proposta. Também há muito espaço para aprimoramentos, elucidados nesta proposta. O autor enxerga estas ressalvas como estímulo para continuidade deste projeto, visto que fora demonstrada a possibilidade de sanar uma dificuldade existente neste tipo de cultivo através de materiais de baixo custo monetário.

REFERÊNCIAS

AITA A.; LONDERO, F. A. A. **Custo de produção de alface hidropônica**. 2000. Santa Maria: Imprensa Universitária.

ALIEXPRESS. **Busca de compra**. Acesso em 10 de Outubro de 2018. <https:/https://www.aliexpress.com/>.

ALMEIDA, A. G. D. de. **Conhecendo o Raspberry Pi: Possibilidades de uso em contextos educacionais e profissionais**. 2015. <https://docente.ifrn.edu.br/andrealmeida>.

ALMEIDA, D. N. **Python, a linguagem do futuro**. 2015. <http://www.odiarioonline.com.br/noticia/45952/PYTHON-A-LINGUAGEM-DO-FUTURO>.

ANDRADE, M. **Histórico dos Computadores**. Acesso em 7 de Outubro de 2018. Notas de aula SEL0415-Introdução à Organização de Computadores <http://iris.sel.eesc.usp.br/>.

ANDRIOLO, J. **Fisiologia das culturas protegidas.** 1999. Santa Maria: UFSM.

ARM. **website oficial da ARM holdings**. Acesso em 13 de Outubro de 2018. <https://arm.com>.

ARM, A. L. **Info**. Acesso em 8 de Outubro de 2018. <https://archlinuxarm.org>.

BARROS, A. **Botânica 15a edição**. 1970. São Paulo: Livraria Nobel S. A.

CARRIJO O. A.; MAKISHIMA, N. **Principios de hidroponia**. 2000. <https://www.embrapa.br/>.

FILIPEFLOP. **Busca de compra**. Acesso em 10 de Outubro de 2018. <https://www.filipeflop.com/>.

MERCADOLIVRE. **Busca de compra**. Acesso em 10 de Outubro de 2018. <https://www.mercadolivre.com.br/>.

NETO EGÍDIO B.; BARRETO, L. P. **As Técnicas de Hidroponia**. 2011. <http://www.journals.ufrpe.br>.

PILAU, F. G. **Intervalo entre irrigações na produção de alface hidropônica**. 2002. Mestrado em Agronomia - Universidade Federal de Santa Maria.

RAGHAVAN, P. **Embedded Linux System Design and Development.** 2006. [S.l.]: Taylor & Francis Group.

RESH, H. **Cultivos hidropônicos**. 1997. Madrid: Mundi-Prensa.

RPIFOUNDATION. **Website oficial da RaspberryPi Foundation**. Acesso em 8 de Outubro de 2018. <https://www.raspberrypi.org/>.

SILVA EDUARDO TEIXEIRA. SCHWONKA, F. **Viabilidade econômica para a produção de alface no sistema hidropônico em Colombo, região metropolitana de**. 2001. <http://revistas.ufpr.br/>.

SLOSS A.; SYMES, D. **ARM system developers guide: designing and optimizing system software**. 2004. [S.l.]: Morgan Kaufmann.

SOUZA, E. S. **Controle de sistema hidropônico utilizando a técnica de fluxo laminar de nutrientes**. 2010. Monografia de Conclusão de Curso <http://lyceumonline.usf.edu.br/salavirtual/documentos/1901.pdf>.

VENTURINI, T. L. **ebook Projetos em hidroponia**. 2014. <http://tudohidroponia.net/>.

Apêndices

APÊNDICE A – APÊNDICE

A.1 Códigos desenvolvidos pelo autor

Também disponível no repositório github.com/lawrencerb

Identificador do sensor de nível e controle de bomba

```
"""
Monitor e controle hidropônico
"""

import RPi.GPIO as GPIO

PIN_STATUS = 4
CONTROL_S0 = 17
CONTROL_S1 = 27
CONTROL_S2 = 22
ACIONAMENTO_BOMBA = 21

def bit_read(byte, bit):
"""Return bit from byte"""
return (byte >> bit) & 1

class HydroMonitor:
"""Classe base com observação dos tubos (has_water) e controle (turn_pump)."""

def __init__(self):
GPIO.setmode(GPIO.BCM)
GPIO.setup(PIN_STATUS, GPIO.IN)
GPIO.setup(CONTROL_S0, GPIO.OUT)
GPIO.setup(CONTROL_S1, GPIO.OUT)
GPIO.setup(CONTROL_S2, GPIO.OUT)
GPIO.setup(ACIONAMENTO_BOMBA, GPIO.OUT)
self.pump_state = 0

def has_water(self, channel):
"""Retorna True caso haja água passando pelo cano ligado ao canal."""
if channel < 0 or channel > 7:
```

```
raise ValueError("Canal não suportado pelo AMUX (CD4051).")
GPIO.output(CONTROL_S0, bit_read(channel, 0))
GPIO.output(CONTROL_S1, bit_read(channel, 1))
GPIO.output(CONTROL_S2, bit_read(channel, 2))
return GPIO.input(PIN_STATUS)

def is_pump_on(self):
return self.pump_state

def turn_pump(self, state=None):
""" Controle de acionamento da bomba.
Chamando turn_pump(), alterna o estado (aberto/fechado),
Chamando turn_pump(1), liga a bomba,
e turn_pump(0), desliga a mesma.
"""
if state is None:
self.pump_state = (self.pump_state and 1 or 0) ^ 1
else:
self.pump_state = state
GPIO.output(ACIONAMENTO_BOMBA, self.pump_state)

def __del__(self):
GPIO.cleanup()
```

Gerenciador Webserver

```
from flask import Flask, Response, redirect, render_template, request, session, json, js
from hydro import HydroMonitor
from camera import Camera

hw = HydroMonitor()
app = Flask(__name__)
app.secret_key = 'Wjj0nY7DhKd9XfTGlmWRmz7HodIN1Iz4afL5vciuDkxSkzPtM7xAwWnZ6Ap2BjPy'

REDIRECT_TO_GET = 303
```

```
def error(status, msg):
r = jsonify(status=status, error=msg)
r.status_code = status
return r

@app.route('/')
def index():
if not 'user' in session:
return redirect('/login')
else:
return render_template('dashboard.html')

@app.route('/login', methods=['GET','POST'])
def login():
if request.method == 'GET':
return render_template('login.html')
else:
p = request.form
if not p or not 'login' in p or not 'senha' in p:
return render_template('login.html', erro='Login ou senha faltando.')
usr = p['login']
pwd = p['senha']
if usr != 'admin' or pwd != 'admin':
return render_template('login.html', erro='Login ou senha incorretos.')
session['user'] = usr
return redirect('/', REDIRECT_TO_GET)

@app.route('/logout', methods=['POST'])
def logout():
session.pop('user', None)
return redirect('/login', REDIRECT_TO_GET)

@app.route('/hw', methods=['GET', 'POST'])
def main():
if not 'user' in session:
return error(401, 'AutenticaÃ§Ã£o requerida.')
if request.method == 'POST':
try:
if ('toggle' in request.form) and request.form['toggle']:
```

```
hw.turn_pump()
else:
hw.turn_pump(request.form['state'])
except:
return error(400, 'RequisiÃ§Ã£o invÃ¡lida: falta toggle ou state.')
return jsonify({
'pipe': [hw.has_water(p) for p in range(8)],
'pump': hw.is_pump_on()
})


def getFrame(camera):
while True:
frame = camera.get_frame()
yield (b'--frame\r\n'
b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')

@app.route('/video')
def video():
return Response(getFrame(Camera()),
mimetype='multipart/x-mixed-replace; boundary=frame')

if __name__ == '__main__':
app.run(host='0.0.0.0', debug=True)
```

Codigo HTML da tela de login

```
<!doctype html>
<html>
<head>
<meta charset=UTF-8>
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstr
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstr
<link rel=stylesheet href="static/css/style.css">
</head>
<body>
```

```
<script src="https://code.jquery.com/jquery-3.2.1.min.js" integrity="sha256-hwg4gsx
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js" i
<script src="static/js/app.js"></script>
<div class=container>
<form class=form-signin method=POST>
<h2 class="form-signin-heading">Monitorar Hidroponia</h2>
<fieldset>
<label for=iLogin class=sr-only>Login</label>
<input name=login type=text placeholder=Login
id=iLogin class=form-control required autofocus>
<label for=iSenha class=sr-only>Senha</label>
<input name=senha type=password placeholder=Senha
id=iSenha class=form-control required>
</fieldset>
{% if erro %}
<div class="alert alert-warning">{{erro}}</div>
{% endif %}
<button class="btn btn-lg btn-primary btn-block" type=submit>Ok</button>
</form>
</div>
</body>
</html>
```

Codigo HTML do Dashboard de Controle

```
<!doctype html>
<html>
<head>
<meta charset="UTF-8">
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bo
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bo
<link rel="stylesheet" href="static/css/style.css">
</head>
<body>
<script src="https://code.jquery.com/jquery-3.2.1.min.js" integrity="sha256-hwg4gsx
```

```
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js" integr
<script src="static/js/app.js"></script>
<div class="container dashboard">
<h1>Dashboard</h1>
<div class="well">
<div class=row>
<div class=col-sm-3>
Controle da bomba:
</div>
<div class=col-sm-2></div>
<div class=col-sm-4>
<button id=botaoBomba type=button onclick="update(1)"
class="btn btn-block btn-default">N/A</button>
</div>
</div>
<div class=row>
<div class=col-sm-3>
Estado dos canos:
</div>
<div class="col-sm-1 canos">
1:<span id=cano0 class="label label-default">N</span></div>
<div class=col-sm-1>
2:<span id=cano1 class="label label-default">N</span></div>
<div class=col-sm-1>
3:<span id=cano2 class="label label-default">N</span></div>
<div class=col-sm-1>
4:<span id=cano3 class="label label-default">N</span></div>
<div class=col-sm-1>
5:<span id=cano4 class="label label-default">N</span></div>
<div class=col-sm-1>
6:<span id=cano5 class="label label-default">N</span></div>
<div class=col-sm-1>
7:<span id=cano6 class="label label-default">N</span></div>
<div class=col-sm-1>
8:<span id=cano7 class="label label-default">N</span></div>
</div>
</div>
<div class=logout>
<form action="/logout" method="POST">
```

```
<button type=submit class="btn btn-small btn-default">logout</button>
</form>
</div>
<div class=video>
<img src="{{ url_for('video') }}" width=640 height=360>
</div>
</div>
</body>
</html>
```

Algoritmo para funcionamento da camera USB

```
import time
import threading
from io import BytesIO
from PIL import Image
from PyV4L2Camera.camera import Camera as V4L2Camera
try:
from greenlet import getcurrent as get_ident
except ImportError:
try:
from thread import get_ident
except ImportError:
from _thread import get_ident

class CameraEvent(object):
"""An Event-like class that signals all active clients when a new frame is
available.
"""
def __init__(self):
self.events = {}

def wait(self):
"""Invoked from each client's thread to wait for the next frame."""
ident = get_ident()
if ident not in self.events:
```

```
# this is a new client
# add an entry for it in the self.events dict
# each entry has two elements, a threading.Event() and a timestamp
self.events[ident] = [threading.Event(), time.time()]
return self.events[ident][0].wait()

def set(self):
"""Invoked by the camera thread when a new frame is available."""
now = time.time()
remove = None
for ident, event in self.events.items():
if not event[0].isSet():
# if this client's event is not set, then set it
# also update the last set timestamp to now
event[0].set()
event[1] = now
else:
# if the client's event is already set, it means the client
# did not process a previous frame
# if the event stays set for more than 5 seconds, then assume
# the client is gone and remove it
if now - event[1] > 5:
remove = ident
if remove:
del self.events[remove]

def clear(self):
"""Invoked from each client's thread after a frame was processed."""
self.events[get_ident()][0].clear()

class Camera(object):
video_source = '/dev/video0'
w = 640
h = 360
thread = None  # background thread that reads frames from camera
frame = None  # current frame is stored here by background thread
last_access = 0  # time of last client access to the camera
event = CameraEvent()
```

```
def __init__(self):
"""Start the background camera thread if it isn't running yet."""
if Camera.thread is None:
Camera.last_access = time.time()
Camera.thread = threading.Thread(target=self._thread)
Camera.thread.start()
while self.get_frame() is None:
time.sleep(0)

def get_frame(self):
"""Return the current camera frame."""
Camera.last_access = time.time()
Camera.event.wait()
Camera.event.clear()
return Camera.frame

@staticmethod
def set_video_source(source, w=640, h=360):
Camera.video_source = source
Camera.w = w
Camera.h = h

@staticmethod
def frames():
"""Generator that returns frames from the camera."""
cam = V4L2Camera(Camera.video_source, Camera.w, Camera.h)
while True:
buf = BytesIO()
img = Image.frombytes('RGB', (cam.width, cam.height), cam.get_frame())
img.save(buf, format='jpeg')
yield buf.getvalue()

@classmethod
def _thread(cls):
"""Camera background thread."""
print('Starting camera thread.')
frames_iterator = cls.frames()
for frame in frames_iterator:
Camera.frame = frame
```

```
Camera.event.set()
time.sleep(1)
# if there hasn't been any clients asking for frames in
# the last 10 seconds then stop the thread
if time.time() - Camera.last_access > 10:
frames_iterator.close()
print('Stopping camera thread due to inactivity.')
break
Camera.thread = None
```