GLÁUCIO FERREIRA LOUREIRO

SISTEMA PARA PROGRAMAÇÃO REMOTA DE VELOCIDADE DE MOTO-BOMBA DE PIVÔ CENTRAL ACIONADA POR INVERSOR DE FREQUÊNCIA

LAVRAS-MG

2017

GLÁUCIO FERREIRA LOUREIRO

SISTEMA PARA PROGRAMAÇÃO REMOTA DE VELOCIDADE DE MOTO-BOMBA DE PIVÔ CENTRAL ACIONADA POR INVERSOR DE FREQUÊNCIA

Tese apresentada à Universidade Federal de Lavras, como parte das exigências do Programa de Pós-Graduação em Recursos Hídricos em Sistemas Agrícolas, área de concentração Irrigação e Drenagem, para a obtenção do título de Doutor.

Prof. Dr. Alberto Colombo

Orientador

LAVRAS-MG

2017

Ficha catalográfica elaborada pelo Sistema de Geração de Ficha Catalográfica da Biblioteca Universitária da UFLA, com dados informados pelo(a) próprio(a) autor(a).

Loureiro, Gláucio Ferreira.
Sistema para programação remota de velocidade de moto-bomba de pivô central acionada por inversor de frequência / Gláucio Ferreira Loureiro. - 2017.
107 p. : il.

Orientador: Alberto Colombo.
Tese (doutorado) - Universidade Federal de Lavras, 2017.
Bibliografia.

1. Automação. 2. Pivô Central. 3. Rendimento Global. I. Colombo, Alberto. II. Título.

GLÁUCIO FERREIRA LOUREIRO

SISTEMA PARA PROGRAMAÇÃO REMOTA DE VELOCIDADE DE MOTO-BOMBA DE PIVÔ CENTRAL ACIONADA POR INVERSOR DE FREQUÊNCIA

SYSTEM FOR THE REMOTE PROGRAMMING OF THE SPEED OF A CENTRAL PIVOT PUMP MOTOR ACTIVATED BY VARIABLE FREQUENCY DRIVE

Tese apresentada à Universidade Federal de Lavras, como parte das exigências do Programa de Pós-Graduação em Recursos Hídricos em Sistemas Agrícolas, área de concentração Irrigação e Drenagem, para a obtenção do título de Doutor.

APROVADA em 14 de setembro de 2017.

Prof. Dr. Adriano Valentim Diotto	UFLA
Prof. Dr. Giovanni Francisco Rabelo	UFLA
Prof. Dr. Fábio Ponciano de Deus	UFLA
Prof. Dr. Paulo Henrique Cruz Pereira	CEFET-MG

Prof. Dr. Alberto Colombo
Orientador

LAVRAS-MG

2017

Aos meus pais Águida Aparecida Ferreira Loureiro (*in memoriam*) e Edson Farias Loureiro (*in memoriam*) por me estimular desde criança a almejar sempre mais conhecimento e sabedoria e pelo apoio dado em meus primeiros passos nos estudos.

À minha esposa Viviane de Castro Freire Loureiro que me apoia desde sempre na caminhada e pelo companheirismo nos momentos mais turbulentos de minha vida.

À minha filha Bia Freire Loureiro por ser mais um motivo para acordar de manhã e fazer o dia valer a pena.

DEDICO

AGRADECIMENTOS

A Deus, por me permitir seguir por caminhos cada vez mais desafiadores e, mesmo tropeçando, me permitir levantar.

À Universidade Federal de Lavras pelo espaço e recursos cedidos e pelo conhecimento adquirido durante esse tempo de estudo.

Ao Programa de Pós-Graduação em Recursos Hídricos em Sistemas Agrícolas pelo apoio técnico-científico que contribuíram para que este estudo fosse desenvolvido.

Ao Prof. Dr. Alberto Colombo que desde o princípio incentivou-me com diversos recursos, desde financeiro até de conhecimento para que este projeto fosse desenvolvido, e pelos momentos de conversas e interação que me permitiram desenvolver como aluno e orientado.

Aos colegas Daniel Araújo, José Henrique, Victor Buono, Ana Sártito, Tiago e Isabela pelos momentos de convivência e auxílio de conhecimento que contribuíram para o meu maior conhecimento e satisfação.

À CAPES pela bolsa de estudos concedida.

Olhe para dentro, para as suas profundezas, aprenda primeiro a se conhecer.

Sigmund Freud

RESUMO GERAL

O consumo de energia deve ser levado em consideração no que se refere à irrigação. O uso de inversor de frequência reduz significativamente o consumo de energia na irrigação, pois adequa a rotação do eixo do motor à necessidade do sistema hidráulico. A sazonalidade altera o nível de água dos reservatórios, impactando na pressão da água bombeada. Em diversas situações não é possível ir a campo para fazer as alterações necessárias no inversor de frequência. Nesse contexto objetivou-se desenvolver um sistema capaz de controlar o inversor de frequência remotamente com o uso de um Raspberry pi e um Arduino associado a um potenciômetro. O sistema foi validado a partir de dados de frequência de sete pivôs centrais e foram comparados os valores inseridos no sistema e os valores fornecidos pelo inversor de frequência. Foi utilizado o teste estatístico t de Student com nível de significância de 5% para fins de comparação. O sistema pode contribuir com alterações dos valores de frequência a distância. Os valores de frequência inseridos no sistema não apresentaram diferença significativa quando comparado aos valores fornecidos pelo inversor. O consumo de energia pode ser estimado pelo uso de equações de rendimento do inversor ou da bomba, interferindo no rendimento global e na potência elétrica ativa. Em algumas situações, tem-se a dificuldade em determinar o rendimento do inversor de frequência o que torna necessário lançar mão de equações de estimativas. Nesse sentido, foi desenvolvido um estudo de comparação entre equações propostas pelos autores Bernier e Bourreout (1999), Sarbu e Borza (1998) e Wallbom-Carlson (1998) para verificar se há diferenças entre os rendimentos propostos pelos autores e o impacto das diferenças no cálculo da potência elétrica ativa. Para efeito de comparação, foi desenvolvida uma análise de variância com teste F de Snadecor em nível de significância de 5%. Observou-se que somente a rotações mais baixas não houve diferença significativa para as equações propostas por Bernier e Bourreout (1999) e Wallbom-Carlson (1998). O cálculo do rendimento global utilizando da equação de rendimento de Bernier e Bourreout (1999) apresentou valores menores que as demais equações, essas características são devido à utilização de maior quantidade de parâmetros.

Palavras-chave: Inversor de frequência. Pivô central. Consumo específico.

GENERAL ABSTRACT

Energy consumption must be taken into consideration when regarding irrigation. The use of a Variable Frequency Drive (VFD) significantly reduces energy consumption during irrigation, by adjusting the rotation of the motor axis to the need of the hydraulic system. Seasonality alters water levels in the reservoir, affecting water pressure. In many occasions, it is impossible to go to field for the necessary adjustments to the frequency inverter. In this context, the objective was to develop a system capable of remotely controlling the VFD, using a Raspberry pi and an Arduino, associated to a potentiometer. The system was validated from the frequency data of seven central pivots, later comparing the values inserted into the system and those provided by the frequency inverter. The t Student statistical test was used with a significance level of 5%, for comparative ends. The system can contribute with changes to the frequency values at a distance. The frequency values inserted into the system presented no significant difference when compared to those provided by the VFD. Energy consumption can be estimated by using efficiency equations for the VFD or pump, affecting global efficiency and active electric power. In some cases, it is difficult to determine the efficiency of the variable frequency drive, demanding the use of estimation equations. In this sense, a comparison study between the equations proposed by Bernier & Bourreout (1999), Wallbom-Carlson (1998) and Sarbu & Borza (1998) was developed to verify the existence of differences between the efficiencies proposed by these authors and the impact of said differences over the calculations of the active electric power. For comparative effect, a variance analysis was developed using the Snedecor F test at a significance level of 5%. There was no significant difference in the low rotations for the equations proposed by Bernier & Bourreout (1999) and Wallbom-Carlson (1998). The calculation of global efficiency used for the Bernier & Bourreout (1999) efficiency equation presented values lower to those of the remaining equations, due to the use of a higher number of parameters.

Keywords: VFD. Center pivot. Specific consumption.

LISTA DE FIGURAS

CAPÍTULO 2

CAPÍTULO 3

LISTA DE TABELAS

CAPÍTULO 2

CAPÍTULO 3

SUMÁRIO

CAPÍTULO 1 INTRODUÇÃO GERAL

1 INTRODUÇÃO

O uso de inversor de frequência permite uma redução significativa dos custos de energia de bombeamento. A adequação das frequências às necessidades do sistema hidráulico permite que o conjunto moto-bomba trabalhe de forma a consumir a energia necessária para o funcionamento do pivô central.

Em diversas situações o nível do reservatório de captação sofre alteração devido à sazonalidade. A variação do nível da água do reservatório pode alterar a pressão da água captada impactando no funcionamento adequado do pivô central. Neste contexto objetivou-se o desenvolvimento de um sistema capaz de alterar as frequências do inversor de forma remota viabilizando alterações do inversor de frequência a distância quando a pressão for detectada inadequada.

O sistema consiste de uma placa microcomputador Raspberry pi configurada como servidor remoto conectado a um Arduino Uno que controla um servo motor responsável pela condução de um potenciômetro de 200kΩ. O potenciômetro foi conectado ao inversor da marca WEG modelo CFW-05 pelas portas analógicas 19, 20 e 22, em que a porta 19 e 22 alimentam o potenciômetro com tensão de 10V e a porta 20 recebe a tensão fornecida pelo potenciômetro. Os dados para avaliação do sistema foram baseados em valores obtidos a partir do funcionamento de sete moto-bombas alimentando sete pivôs centrais.

Foi desenvolvido o teste estatístico t de Student a nível de significância de 5% para verificação das diferenças entre os valores de alimentação do sistema e os valores obtidos a partir do uso do sistema.

Os dados programados no sistema e os dados obtidos através do sistema não apresentaram diferença significativa a nível de 5% no teste estatístico t de

Student. O sistema pode ser útil para alterações de frequência quando não é possível ir a campo.

O cálculo do rendimento global é necessário para a determinação da potência elétrica ativa e do consumo específico em relação a um volume de água aplicado na irrigação. Em muitas situações não é possível conhecer os valores que envolvem o rendimento do inversor.

Diversos autores propõem equações matemáticas para a determinação do rendimento do inversor a partir do conjunto moto-bomba-inversor.

Objetivou-se comparar três equações propostas por Bernier e Bourreout (1999), Sarbu e Borza (1998) e Wallbom-Carlson (1998) visando avaliar se há diferenças significativas entre o rendimento global calculado a partir dos valores de rendimento do conjunto moto-bomba-inversor e o impacto destas diferenças no cálculo da potencie elétrica ativa.

Os dados para essa comparação foram obtidos a partir de sete conjuntos moto-bomba-inversor alimentando sete pivôs centrais.

Foi desenvolvido uma análise de variância com teste F de Snadecor em nível de significância de 5% para verificar se houve diferença significativa entre as equações.

Observou-se que somente a comparação entre as equações de Bernier e Bourreout (1999) e Wallbom-Carlson (1998) não obtiveram uma diferença significativa entre os dados dos pivôs 2 que apresentavabaixas rotações, demonstrando que essas equações não apresentam uma diferença significativa a nível de 5% nesta faixa de valores. As demais comparações apresentaram diferença significativa.

Tem-se que as equações apresentam diferenças nos cálculos de rendimentos impactando nos valores de potência elétrica ativa calculada.

Os valores de rendimento global calculado a partir da equação do rendimento do inversor proposta por Bernier e Bourreout (1999), com o

rendimento da bomba e do motor tem valores menores que as demais equações de rendimento, devido as características de levar em consideração maior número de parâmetros para os cálculos.

CAPÍTULO 2 DESENVOLVIMENTO DE UM SISTEMA PARA CONTROLE A DISTÂNCIA DE INVERSORES DE FREQUÊNCIA

RESUMO

O uso de tecnologia na irrigação é um parâmetro a ser considerado quando se tem a necessidade de economia de energia e maior controle dos processos. No Brasil, o uso de automação na irrigação, ainda, é incipiente, tendo em vista que a falta de mão de obra qualificada associada ao custo para automatizar inviabiliza, em muitos casos, a sua adoção. Em diversas situações, tem-se a necessidade de adequar, durante o funcionamento do sistema de irrigação, as rotações do eixo do motor às características do sistema hidráulico. O sistema de pivô central, dependendo da região em que está instalado, pode ter entre seus ângulos de giro diferentes cotas de relevo. O consumo de energia é maior em pontos de maior cota e menor em pontos de menor cota. O uso do inversor de frequência altera a frequência de alimentação da moto-bomba permitindo a adequação da rotação do eixo do motor às necessidades do sistema hidráulico. Em diversas situações, devido à sazonalidade há uma alteração no nível de água do reservatório de captação o que torna necessária a adequação da rotação do eixo do motor para que a pressão bombeada não sofra alteração.Em muitos casos, não é possível ir a campo efetuar essa alteração. Nesse contexto, objetivou-se desenvolver um sistema capaz de permitir um controle a distância do inversor de frequência que controla a moto-bomba que alimenta o pivô central. O sistema foi desenvolvido a partir de uma placa microcomputador Arduino modelo Uno e uma placa microcontrolador Raspberry pi controlando um potenciômetro de 200k Ω. Foram utilizados dados de sete moto-bombas conectados, alimentando sete pivôs centrais para validação do sistema, e o teste estatístico t de Student a 5% de nível de significância, para verificar se há diferença significativa entre os dados inseridos no sistema e os dados obtidos a partir do inversor de frequência. Observou-se que não houve diferença significativa entre os dados do sistema e os dados obtidos a partir do inversor de frequência. O uso do sistema pode ser uma alternativa quando não se pode ir a campo para alterar os parâmetros do inversor de frequência.

Palavras-chave: Raspberry pi. Arduino. Controle de Frequência

ABSTRACT

The use of technology in irrigation is a parameter to be considered when it is necessary to save energy and greater control of the processes. In Brazil, the use of automation in irrigation is still incipient, since the lack of skilled labor associated with the cost to automate makes it in many cases impossible to adopt. In several situations, it is necessary to adjust, during the operation of the irrigation system, the rotations of the motor shaft to the characteristics of the hydraulic system. The central pivot system, depending on the region in which it is installed, can have different relief dimensions between its angles of rotation. The energy consumption is higher in points of higher share and lower in points of lower quota. The use of the variable frequency drives (VFD) changes the frequency of the motor pump to allow the rotation of the motor shaft to suit the needs of the hydraulic system. In several situations, due to seasonality there is a change in the water level of the catchment reservoir which makes it necessary to adjust the rotation of the engine shaft so that the pumped pressure does not change. In many cases, it is not possible to go to the field this change. In this context, the objective was to develop a system capable of allowing remote control of the VFD that controls the motor pump that feeds the central pivot. The system was developed from an Arduino model One microcomputer and a Raspberry pi microcontroller board controlling at 200kΩ potentiometer. And the statistical test of Student at 5% level of significance, to verify if there is significant difference between the inserted data in the system and the data obtained from the VFD. It was observed that there was no significant difference between the system data and the data obtained from the VFD. The use of the system can be an alternative when you can not go to the field to change the VFD parameters.

Keywords: System. Frequency control. VFD.

1 INTRODUÇÃO

Diversos recursos têm sido utilizados na modernização e redução dos custos de bombeamento. No Brasil, uma das maiores dificuldades encontradas está relacionada ao uso de mão de obra qualificada e o conhecimento para a adoção de equipamentos de baixo custo, o que torna, muitas vezes, inviável a adoção de métodos de automação e controle da irrigação.

Para monitoramento e controle automático faz-se necessário o uso de uma placa seja para a obtenção de dados ou para o envio de informação para determinado sensor. Uma alternativa com custo relativamente baixo é o uso do arduino, que de acordo com Susmethaa et al. (2015) é uma placa de aquisição de dados open source, de fácil utilização e faz uso do microcontrolador ATmega 316, a qual, em suas diversas conexões disponíveis, pode-se conectar a um sensor de umidade do solo ou um sensor de nível de água.

O uso de monitoramento e o uso de sensores no campo são uma das tecnologias que podem contribuir para um uso adequado do sistema de irrigação e para a redução dos custos de bombeamento. Além dos estudos que envolvem a instrumentação, destacam-se também análises de dados e desenvolvimento de sistemas que permitem o uso do sistema de irrigação baseado no consumo ótimo.

Daccache et al. (2014) desenvolveram algoritmos para analisar a relação entre a uniformidade de aplicação com o nível de pressão em hidrantes, este é um importante parâmetro, pois descreve estratégias de redução do consumo de energia com a uniformidade de aplicação associadas à colheita e produção.

Moreno et al. (2012) desenvolveram um software capaz de correlacionar o design ideal do pivô central e do movimento de sua lateral considerando uma relação teórica entre as características e eficiência das curvas de bomba.

Carrión et al. (2013, 2014) desenvolveram uma ferramenta para a tomada de decisão que envolve o tamanho do conjunto do aspersor com o custo mínimo total por unidade de área irrigada.

A modernização dos meios de produção reduz o consumo de água no cultivo, aumentam a produção por unidade de área e volume de água e contribui para a redução do consumo de energia. A automação com uso de informações e feedback sobre o que ocorre na fazenda contribui na redução do consumo de energia, água e fertilizantes e reduz o impacto da irrigação no meio ambiente (TARJUELO et al., 2015).

Além das questões que envolvem modelagem e instrumentação, tem-se, ao longo dos anos, uma melhoria nas características dos motores de indução elétrica presente nas bombas no que se refere ao design, confiabilidade, além da redução no tamanho, e em seu custo. Apesar das melhorias, motores de indução conectados diretamente à rede elétrica atuam com rotação constante, devido à frequência de alimentação ser constante, o que acarreta um maior consumo de energia elétrica. O inversor de frequência é capaz de fornecer diferentes ajustes de frequência e tensão permitindo ao motor trabalhar com a rotação necessária a fornecer a pressão adequada ao sistema hidráulico (CAMOIRANO; DELLEPIANE, 2005).

Em sistemas de irrigação do tipo pivô central, tem-se, no Brasil, diferentes condições topográficas no terreno, onde em locais de aclive o pivô precisará de uma maior pressão e em locais de declive essa pressão pode ser reduzida. Nesse contexto, Campana (2000), analisou o comportamento do índice de carregamento e rendimento dos motores elétricos de três tipos de pivô, média, baixa e muito baixa pressão. Observou-se que acréscimos de rendimento das ordens de 3,71%, 5,97% e 2,19%, respectivamente, quando se utiliza o inversor de frequência.

A viabilidade econômica de pivôs centrais associados ao uso de inversores de frequência apresenta uma boa relação de custo benefício. Campana et al. (2003) calcularam para os pivôs operando entre 3500 e 4500 horas por ano, a taxa interna de retorno média de 3,3% e 7,8%, respectivamente. Já no cenário que considera a redução no preço do inversor de 10% e 20% tem-se a taxa de retorno interna média nos valores de 7,9% e 13,4%. Assim, tem-se que a viabilidade econômica com o uso do inversor é função das características hidráulicas e elétricas do pivô e do sistema de bombeamento.

O uso do inversor de frequência permite uma redução significativa no consumo de energia, pois adequa o motor às rotações necessárias às características que envolvem o sistema hidráulico.

O projeto de irrigação permite que o pivô central seja dimensionado de forma adequada a consumir a energia necessária ao seu funcionamento, contudo, em algumas situações, devido à sazonalidade, há uma alteração no nível do reservatório de captação podendo impactar na pressão de água bombeada.

A necessidade de alteração dos parâmetros do inversor, em algumas situações, pode ser inviável, devido à dificuldade de ir a campo em determinado momento.

A partir disso, objetivou-se com esse estudo desenvolver um sistema capaz de permitir um controle remoto do inversor de frequência. O sistema foi alimentado a partir dos dados de frequência de sete conjuntos de moto-bomba alimentando sete pivôs centrais.

Os dados inseridos no sistema foram comparados aos dados fornecidos pelo inversor de frequência por meio de um teste F em nível de significância de 5% e o coeficiente de correlação de Pearson. Foi observado que houve diferença significativa para quatro conjuntos moto-bomba-inversor, contudo todos os conjuntos apresentaram uma forte correlação positiva. O sistema pode ser uma

ferramenta importante no controle e no auxílio da redução do consumo de energia de bombeamento.

2 MATERIAIS E MÉTODOS

Este trabalho foi desenvolvido no setor de Engenharia de Água e Solo do Departamento de Engenharia da Universidade Federal de Lavras, e consistiu de desenvolver um equipamento que permitisse a alteração da rotação da bomba através do envio sinais com tensão que variou de 0 a 10V a um inversor da marca WEG modelo CFW-05 e o software de coleta fornecido pelo fabricante SuperDrive v.5.70.

2.1 Desenvolvimento dos hardwares e softwares do sistema

O sistema é composto de um Arduíno modelo Uno rev.3 (FIGURA 1), conectado e controlando um servo motor tuning modelo TG9E (FIGURA 3) acoplado a um potenciômetro logarítmico de resistência de 200kΩ.

Foi utilizado um Raspberry pi 3 modelo B (FIGURA 2) configurado por via do software XRDP para atuar como um servidor remoto.

Para conexão com a rede, foi utilizado um roteador 3G portátil da marca TP-link modelo MR3020 e um modem 3G da marca D-link, em que foi configurado um servidor ddns a partir do site Weaved.

Figura 1 - Arduíno modelo Uno.

Fonte: Arduino (2017).

Figura 2 - Raspberry pi 3 modelo B.

Fonte: Raspberry Pi Foundation (2017).

Figura 3 - Servo motor Turnigy modelo TG9e.

Fonte: Hobbyking (2017).

A comunicação entre usuário e sistema foi estruturada em duas vertentes, a comunicação desenvolvida para atuar no Raspberry pi e comunicar com o usuário e o firmware desenvolvido para controlar o servo motor a partir do Arduino.

A comunicação com o usuário foi desenvolvida em linguagem Python (FIGURA 5) e a programação do firmware foi desenvolvida na linguagem Arduino (FIGURA 4).

A comunicação com o usuário baseia-se na inserção de 36 valores de frequência, em que é possível salvar o conjunto de valores via arquivo com extensão txt, caso seja necessário carregá-lo posteriormente.

Figura 4 - Trecho da programação do firmware do Arduino no qual os loops aumentam ou diminuem a rotação do servo motor.

```
if (auxpos < maxgiro){


for(pos = auxpos; pos <= maxgiro; pos = pos+1){
s.write(pos);
Serial.println(pos);
delay(100);
}
}

if (auxpos > maxgiro){
for(pos = auxpos; pos >= maxgiro; pos = pos-1){
Serial.println(pos);
s.write(pos);
delay(100);
}
}
```

Fonte: Do autor (2017).

Figura 5 - Trecho da programação da comunicação com o usuário desenvolvido em linguagem Python.

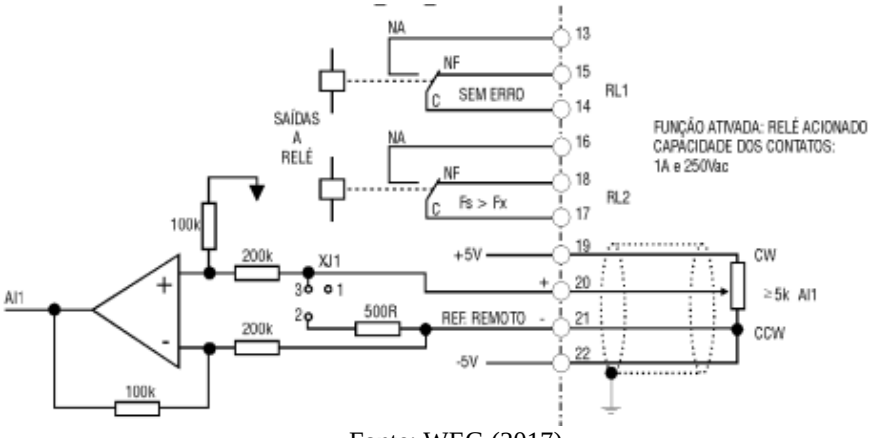
```
t = time.localtime()
auxtemp = t[4] + temp   #fica em um loop pelo tempo de giro do pivo
if auxtemp > 59:
    auxtemp = 60 - t[4]
    auxtemp = temp - auxtemp
while t[4] != auxtemp:
    x = 1
    t = time.localtime()
ser.write((self.Entrada2.get()).encode()) # para que o programa no Arduino identifique a informação
self.Entrada2.delete(0, END) # Limpa os valores dos campos de entrada
```

Fonte: Do autor (2017).

2.2 Comunicação do sistema com o inversor

Para a conexão com o inversor foi utilizado um Keystone modelo RJ-11 e um cabo de rede modelo cat5e com conectores RJ-45 conectados no inversor via portas analógicas 19, 20 e 22. As portas 19 e 22 são, respectivamente, tensão fornecida pelo inversor de +5V e -5V, respectivamente. A porta 20 é a porta de recepção de tensão advinda do potenciômetro com variação de 0 a 10V (FIGURA 6).

Figura 6 - Demonstração das portas analógicas 19, 20 e 22 do inversor WEG CFW-05.



Fonte: WEG (2017).

O potenciômetro apresenta giro máximo de 180°. A partir disso, para fins de calibração foi desenvolvido um laço de programação no Arduino, no qual os ângulos de rotação do servo motor foi decrementado, a partir da angulação máxima até a angulação mínima.

Foram verificadas as leituras de frequência, voltagem de controle, fornecidas pelo inversor de frequência, por meio do programa Super Drive com o objetivo de associar o ângulo de rotação do servo motor com a tensão fornecida pelo potenciômetro.

O inversor de frequência atua a partir de uma frequência mínima e máxima pré-estabelecida, estabelecidas na programação P133 e P134, respectivamente. Com a tensão de corrente contínua fornecida pelo inversor é possível, a partir da Equação 1, os parâmetros da equação que associa a tensão de controle analógico do inversor com a frequência de atuação do inversor (WEG, 2017).

Neste estudo, a frequência mínima foi estabelecida de 3Hz e a máxima de 60Hz e determinados os parâmetros da Equação 2.

$$F = \{[(F_{max} - F_{min}) * v_{max}^{-1}] * v\} + F_{min} \quad (1)$$

Em que:

F_{max}: frequência máxima configurada no inversor de frequência(hz)

F_{min}: frequência mínima configurada no inversor de frequência (hz)

v_{max}: tensão de alimentação fornecida ao potenciômetro (v)

v: tensão no qual o potenciômetro fornece após a corrente passar pela resistência (v)

Como os valores de $F_{min,}$ F_{max} foram determinados antes dos testes e o valor de v_{max} é fornecido por padrão pelo inversor, tem-se a seguinte equação determinada (WEG, 2017):

$$F = 5{,}7 * v + 3 \quad (2)$$

2.3 Calibração do sistema com o inversor de frequência

Os dados de tensão para o controle analógico do inversor de frequência necessitaram de uma calibração para associar a informação de frequência informada pelo usuário intermediada pela conversão em tensão de controle remoto analógico do inversor de frequência e, por fim, o giro do potenciômetro para obter-se esta tensão.

Os parâmetros da equação de calibração, entre o conjunto servo motor e potenciômetro, foram calculados a partir de uma regressão linear. Nos eixos das ordenadas foram expressos os valores de ângulo de rotação do conjunto servo motor e potenciômetro e no eixo das abscissas foram expressos os valores de tensão analógica de controle do inversor.

Com os dados, foi possível observar que houve uma diferença no giro crescente do conjunto servo motor e potenciômetro em detrimento ao giro decrescente. Além disso, valores de tensão de controle que estavam associados à frequência do inversor acima de 48Hz também apresentaram diferenças. Nesse sentido, foram desenvolvidas três regressões: valores crescentes de frequência, valores decrescentes de frequência e valores nos quais a frequência é maior que 48Hz.

2.4 Caracterização dos dados utilizados na simulação do sistema

Os dados de frequência para simulação do sistema foram obtidos a partir dos valores de sete conjuntos moto-bomba-inversor (MBI).

Cada conjunto de frequência continha 37 valores de frequência, com exceção ao conjunto MBI7 que continha 27 valores de frequência (TABELA 1).

Tabela 1 - Dados de frequência de alimentação do sistema para validação.

(Continua)

MBI 1	MBI 2	MBI 3	MBI 4	MBI 5	MBI 6	MBI 7
Frequência (Hz)						
55,3	42,9	58,2	57,5	52,6	52,5	56,1
55,5	43,4	58,2	58,2	51,3	52,3	56,3
55,6	44,0	58,1	58,7	50,2	52,1	56,5
55,8	44,8	58,0	59,1	49,4	51,9	56,6
56,1	45,9	57,8	59,1	49,1	51,8	56,8
56,5	47,2	57,3	58,7	49,2	51,7	56,9
57,1	48,7	56,6	58,2	49,4	51,9	57,0
57,9	50,1	55,8	57,5	49,4	52,6	56,9
58,5	51,6	54,9	57,0	49,2	53,8	56,7
58,5	53,2	54,1	56,5	48,9	55,4	56,6
58,1	54,6	53,6	56,1	48,7	56,4	56,5
57,5	55,6	53,3	55,8	48,6	56,2	56,2
56,7	56,0	53,2	55,6	48,7	55,4	56,0
56,1	55,7	53,1	55,5	48,7	55,0	55,7
55,5	54,9	53,2	55,5	48,7	54,5	55,4
54,9	53,7	53,2	55,5	48,7	53,6	55,0
54,4	52,3	53,2	55,5	48,7	52,7	54,7
54,1	50,6	53,2	55,5	48,7	52,0	54,6
53,9	49,0	53,2	55,6	48,7	51,9	54,7
53,8	47,6	53,2	55,6	48,7	52,3	54,7
53,8	46,7	53,2	55,8	48,7	53,1	54,7
53,8	46,2	53,2	55,9	48,7	54,4	54,7
53,7	45,7	53,2	56,0	49,3	55,1	54,7
53,7	45,0	53,1	56,1	50,7	54,8	54,7
53,7	44,0	53,2	56,2	52,5	53,9	54,7
53,7	42,8	53,4	56,3	54,1	52,9	54,7
53,7	41,8	54,0	56,3	55,4	51,9	54,7

Tabela 1 - Dados de frequência de alimentação do sistema para validação.

(Conclusão)

MBI 1	MBI 2	MBI 3	MBI 4	MBI 5	MBI 6	MBI 7
Frequência (Hz)						
53,7	41,2	54,8	56,2	56,4		54,7
53,8	40,8	55,6	56,1	57,1		54,7
54,0	40,4	56,4	56,1	57,5		54,7
54,1	40,2	57,2	56,0	57,6		54,7
54,1	40,2	58,0	56,0	57,5		54,7
54,2	40,3	58,6	56,1	57,0		54,8
54,3	40,5	58,6	56,3	55,9		55,1
54,6	40,9	58,4	56,6	54,7		55,6
55,0	41,8	58,3	57,0	53,6		55,9
55,3	42,9	58,2	57,5	52,6		56,1

Fonte: Azevedo (2003).

2.5 Caracterização dos testes

Na condução dos testes foi utilizado o software SuperDrive v.5.70 em que foram inseridos os valores de frequência no sistema desenvolvido e verificados os valores obtidos a partir do SuperDrive 5.70.

Os parâmetros e a denominação na programação do inversor obtidos por meio do SuperDrive 5.70 foram, respectivamente, frequência do motor (P005), tensão de saída (P007), potência de saída (P010) e entrada analógica Al1 (P018), a entrada Al1 é o parâmetro no qual tem-se a tensão de controle analógico do inversor.

Os valores obtidos a partir do software SuperDrive 5.70 foram comparados com os valores inseridos.

Para fins de comparação foi realizado o teste estatístico F de Snadecor a nível de significância de 5%.

3 RESULTADOS E DISCUSSÕES

3.1 Caracterização do sistema

O sistema apresenta em sua característica a possibilidade de inserção de 36 valores de frequência.

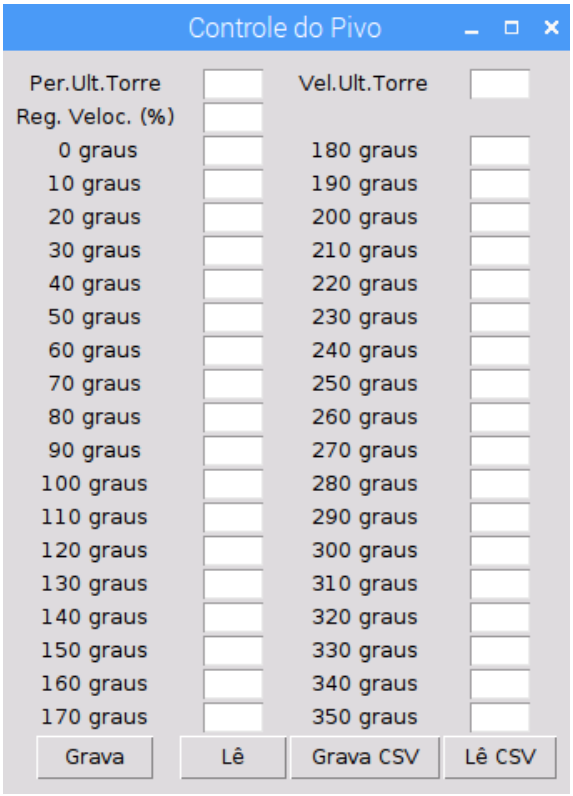
O sistema contém um temporizador sincronizado com o relógio do Raspberry pi, em que, é possível determinar qual o tempo que se deseja que haja mudança dos valores de frequência.

A alteração nos valores de frequência é enviada ao Arduino por controlar o servo motor.

O sistema permite que sejam salvos o conjunto de valores de frequência. O arquivo é salvo na extensão csv. Para essa funcionalidade, tem-se o botão “Grava csv”. Os dados salvos em csv também podem ser carregados no sistema, para essa funcionalidade, tem-se o botão “Lê csv” (FIGURA 7).

Para se ter acesso de forma remota ao sistema, tem-se a necessidade do acesso ao site de ddns weaved. O site disponibiliza um link, no qual, preenchendo este link em um programa de acesso remoto, permite o acesso à tela do Raspberry pi e permite que o sistema seja executado.

Figura 7 - Tela de controle do pivô do sistema desenvolvida em Python.



Fonte: Do autor (2017).

3.2 Validação do sistema

Com o conjunto de dados de frequência alimentado no sistema, obteve-se um conjunto de dados fornecidos pelo inversor controlado pelo sistema. A partir disso, foi possível plotar as curvas de cada conjunto moto-bomba-inversor (MBI) comparando os dados alimentados e dados obtidos a partir do inversor de frequência (FIGURA 8).

Figura 8 - Gráficos dos dados de frequência entrada no sistema e de resposta do inversor de frequência em relação ao tempo de teste para os sete conjuntos moto-bomba-inversor.

(Continua)

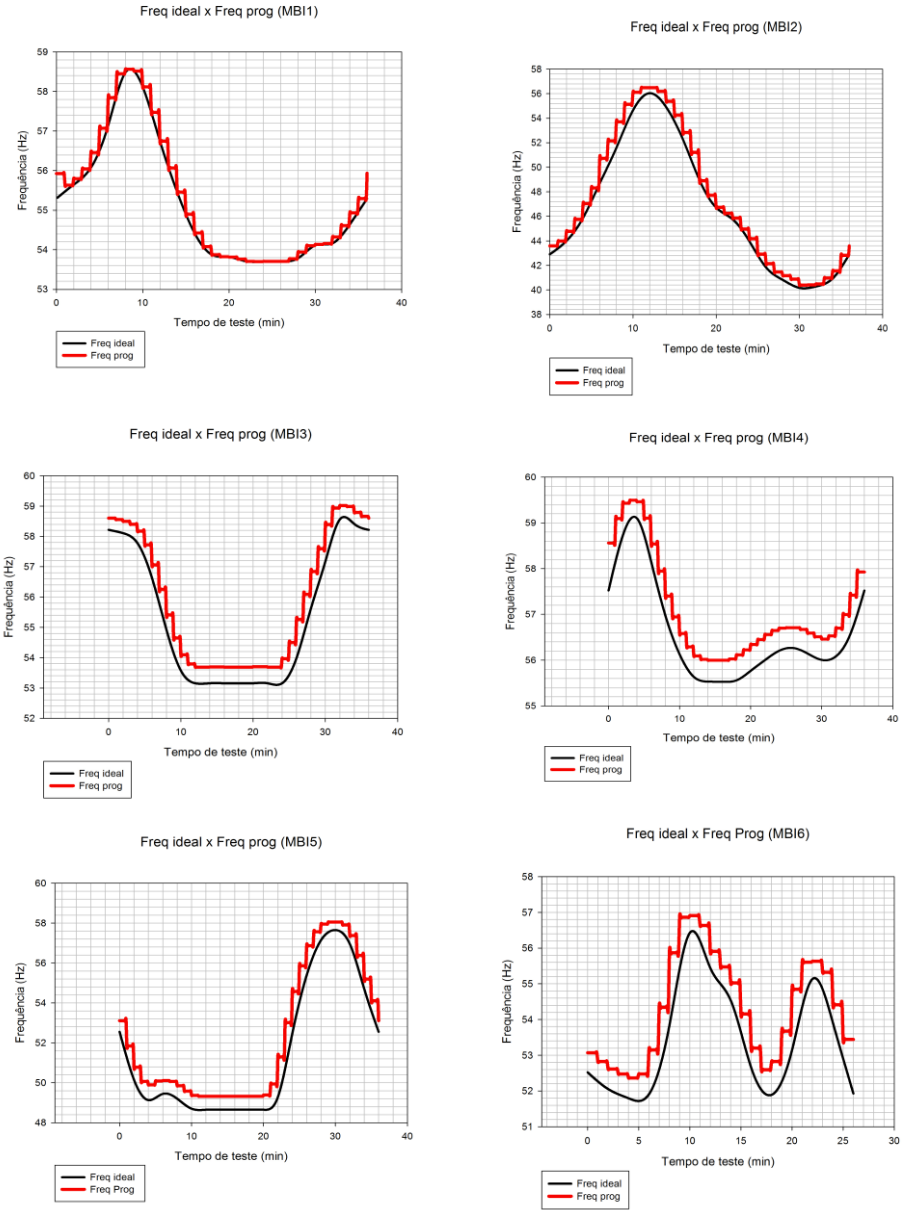
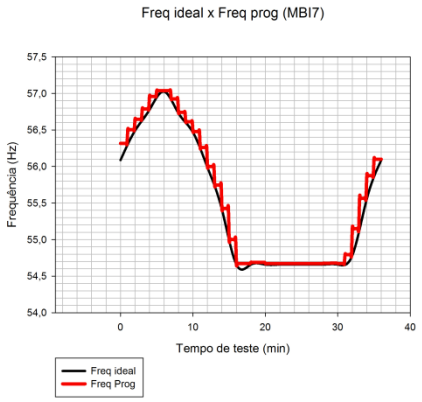


Figura 8 - Gráficos dos dados de frequência entrada no sistema e de resposta do inversor de frequência em relação ao tempo de teste para os sete conjuntos moto-bomba-inversor.

(Conclusão)



Fonte: Do autor (2017).

A partir dos gráficos é possível perceber uma maior diferença nos conjuntos MBI3, MBI4, MBI5, MBI6. Nesse sentido, foi desenvolvido o teste F em nível de significância de 5%, comparando os dados inseridos no sistema com os dados obtidos a partir do inversor de frequência. Foram também obtidos os valores do coeficiente de correlação de Pearson (TABELA 2).

Tabela 2 - Resultado do teste F e do coeficiente de correlação de Pearson entre os dados inseridos no sistema e os dados obtidos no inversor de frequência.

Teste F				
Conjunto	F	Fcrítico a 5%	Signif	Correlação
MBI1	1,391451	3,854407	NS	0,9953
MBI2	2,709835	3,8544	NS	0,997
MBI3	14,8798	3,8544	S	0,9955
MBI4	44,23	3,8544	S	0,9941
MBI5	11,58	3,8544	S	0,9948
MBI6	41,39	3,8544	S	0,9786
MBI7	1,4627	3,8544	NS	0,9957

Fonte: Do autor (2017).

Por meio do teste F de Snadecor em nível de significância de 5%, observou-se que os conjuntos MBI3, MBI4, MBI5, MBI6 apresentaram diferença significativa, contudo, todos os conjuntos obtiveram forte correlação positiva. Demonstrando que o sistema responde às alterações de frequência solicitadas pelo usuário.

4 CONCLUSÃO

O desenvolvimento do sistema permitiu que o acesso remoto fosse feito para alterar a frequência do inversor de acordo com a necessidade do sistema hidráulico.

O sistema apresentou uma diferença significativa em quatro conjuntos moto-bomba-inversor avaliados, contudo, em todos obteve uma forte correlação positiva.

O sistema demonstrou a possibilidade, de alteração de forma remota, o uso do inversor de frequência controlando um conjunto moto-bomba, contudo necessita de ser aprimorado.

Como desenvolvimento futuro, sugere-se o uso de um modelo de Arduino que permita a inserção de tensão de forma direta no inversor, via porta analógica de saída do Arduíno, eliminando o uso do potenciômetro.

REFERÊNCIAS

ARDUINO. **Arduino UNO REV3**. Disponível em: <https://store.arduino.cc/usa/arduino-uno-rev3>. Acesso em: 31 ago. 2017.

AZEVEDO, E. B. **Viabilidade do uso do inversor de frequência em sistema de irrigação do tipo pivô central**. 2003. 77 p. Dissertação (Mestrado em Engenharia Agrícola)-Universidade Federal de Lavras, Lavras, 2003.

BENIER, M. A.; BOURRET, B. Pumping energy and variable frequency drivers. **ASHRAE Journal**, Atlanta, v. 12, p. 37-40, 1999.

CAMOIRANO, R.; DELLEPIANE, G. Variable frequency drives for MSF desalination plant and associated pumping stations. **Desalination**, Amsterdam, v. 182, n. 1/3, p. 53-65, 2005.

CAMPANA, S. **Racionalização do uso da energia elétrica em sistemas de irrigação tipos pivô central e por aspersão convencional**. 2000. 110 p. Dissertação (Mestrado em Engenharia Agrícola)-Universidade Federal de Viçosa, Viçosa, MG, 2000.

CAMPANA, S. et al. Inversores de freqüência: uma alternativa para racionalização do uso da energia elétrica em sistemas de irrigação pivô central. In: ENCONTRO DE ENERGIA NO MEIO RURAL, 3., 2003, Campinas. **Anais…** Campinas: Ed. UNICAMP, 2003. Disponível em: <http://www.proceedings.scielo.br/scielo.php?pid=MSC0000000022000000200029&script=sci_arttext>. Acesso em: 10 mar. 2017.

CARRIÓN, F. et al. Design of microirrigation subunit of minimum cost with proper operation. **Irrigation Science**, New York, v. 31, n. 5, p. 1199-1211, 2013.

CARRIÓN, F. et al. Design of sprinkler irrigation subunit of minimum cost with proper operation: application at corn crop in Spain. **Water Resources Management**, Oxford, v. 28, n. 14, p. 5073-5089, 2014.

DACCACHE, A. et al. Water and energy footprint of irrigated agriculture in the Mediterranean region. **Environmental Research Letters**, Bristol, v. 9, n. 12, 2014. Disponível em: <http://iopscience.iop.org/article/10.1088/1748-9326/9/12/124014/meta>. Acesso em: 10 mar. 2017.

HOBBYKING. **Servo motor turnigy 9e**. Disponível em: <https://hobbyking.com/en_us/turnigytm-tg9e-eco-micro-servo-1-5kg-0-10sec-9g.html?___store=en_us>. Acesso em: 31 ago. 2017.

MORENO, M. A. et al. Optimal design of center pivot systems with water supplied from wells. **Agricultural Water Management**, Amsterdam, v. 107, p. 112-121, 2012.

RASPBERRY PI FOUNDATION. **Raspberry PI 3 model B**. Disponível em: <https://www.raspberrypi.org/products/raspberry-pi-3-model-b/>. Acesso em: 31 ago. 2017.

SARBU, I.; BORZA, I. Energetic optimization of water pumping in distribution systems. **Periodica Polytechnica.** Mechanical Engineering, Budapest, v. 42, n. 2, p. 141-152, 1998.

SUSMETHAA, S. R. et al. Hybrid water pumping control system for irrigation using Arduino. **International Journal of Engineering Research and Technology**, Bremen, v. 4, n. 3, p. 858-863, Mar. 2015.

TARJUELO, J. M. et al. Efficient water and energy use in irrigation modernization: lessons from Spanish case studies. **Agricultural Water Management**, Amsterdam, v. 162, p. 67-77, 2015.

WALLBOM-CARLSON, A. Energy comparison. VFD vs. on-off controlled pumping stations. **Scientific Impeller**, Stockholm, v. 136, n. 5, p. 29-32, 1998.

WEG. **Manual do inversor de frequência CFW-05**. Disponível em: <http://www.manutensul.com.br/downloads/Weg/Manual%20CFW05.pdf>. Acesso em: 28 ago. 2017.

CAPITULO 3 USO DE EQUAÇÕES DE RENDIMENTO PARA ESTIMATIVA DO CONSUMO ESPECÍFICO DE MOTOBOMBA

RESUMO

O consumo de energia é um parâmetro a ser levado em consideração para a redução dos custos com o cultivo. O uso de inversores de frequência permite uma redução significativa dos custos de bombeamento. Em muitas situações, não têm-se disponíveis os dados referentes ao rendimento do inversor. Na atualidade, existem diversas equações disponíveis para estimar o rendimento que envolve o conjunto moto-bomba-inversor. Nesse sentido, objetivou-se com esse estudo calcular, a partir dos dados de sete conjuntos moto-bombas-inversor, o rendimento global, a potência elétrica ativa e o consumo específico para os autores Bernier e Bourreout (1999), Sarbu e Borza (1998) e Wallbom-Carlson (1998). Foi calculado o consumo específico para o uso do inversor e sem o uso do inversor. Para fins verificação se há diferença significativa entre as equações, foi desenvolvido um teste F de Snadecor a 5%. O estudo demonstrou não haver uma diferença significativa entre os autores propostos. O uso do inversor permitiu uma economia de 20% em um dos pivôs estudados.

Palavras-chave: Consumo específico. Inversor de frequência. Rendimento Global.

ABSTRACT

Energy consumption is a parameter to be taken into account for the reduction of crop costs. The use of variable frequency drives (VFD) allows a significant reduction of pumping costs. At present, there are several equations available to estimate the efficiency of the motor-pump-vfd assembly. Bernier and Bourreout (1999), Sarbu and Borza (1998) and Wallbom-Carlson (1998), the global efficiency, the electric power and the specific consumption for the authors were calculated from the informations of seven center pivots. Specific consumption was calculated for the use of the inverter and without the use of the inverter. The study demonstrated there is no significant difference between the proposed authors. The use of the inverter allows the saving of 20% in one of the pivots studied.

Keywords: Specific Comsuption. VFD. Global Efficiency.

1 INTRODUÇÃO

O consumo de energia é um importante parâmetro no que se refere às questões dos custos de bombeamento. De acordo com Singh, Mishra e Nahar (2002), estima-se que, do consumo de energia existente, cerca de 23 a 48% estão relacionadas às questões que envolvem a produção de alimentos em fazendas que necessitam de bombeamento de água. Estatísticas atestam que 46% da energia gerada no mundo está relacionada à questão de motores elétricos (WAIDE; BRUNER, 2011). No Brasil, estima-se que 8% da energia gerada e distribuída são utilizadas pela área rural, e que, ainda há, nessa estatística, as agroindústrias que se enquadram no consumo industrial (BRASIL, 2017).

O uso do sistema de bombeamento em sua capacidade máxima acarreta um custo elevado. Os inversores de frequência é uma alternativa quando há necessidade de alterar a rotação da bomba de acordo com a variação no fluxo de distribuição do sistema.

Brar et al. (2017) analisou, por meio de dados de topografia obtidas por sistemas de informação geográfica, a influência da topografia de pivôs instalados no estado do Nebraska nos EUA, visando a obter um modelo capaz de reduzir o consumo de energia na irrigação com pivô central. Ao desenvolver o modelo e utilizar os dados de pressão total requerida, potência total requerida, potência hidráulica e analisar as cotas de elevação nas quais a torre do pivô estava era submetida, permitiu o cálculo kWh por volume bombeado, bem como, do valor em dólares por volume bombeado. Além disso, o uso do inversor foi analisado por via do capital investido no inversor em relação ao capital economizado com seu uso. O uso dos cálculos associado ao inversor de frequência permitiu em um dos casos de estudos economia em torno de 9,6% no custo com energia elétrica.

O sistema de irrigação por pivô central consiste de uma linha de aspersores localizados entre torres de sustentação com movimento circular

disposto acima da área de cultivo tendo como suplemento para funcionamento a energia elétrica e a água. Com o uso de inversores de frequência tem-se uma redução significativa do consumo de energia no uso do sistema de irrigação em pivô central.

Shankar et al. (2016) utilizaram em seus estudos a equação citada por Bernier Bourret (1999) como um dos cálculos para a determinação do rendimento total do sistema de irrigação. O estudo propôs analisar os parâmetros que influenciam no consumo de energia de bombeamento demonstrando conceitos desde a escolha da bomba ideal e na adoção de critérios para funcionamento da bomba, tais como a altura manométrica requerida e o uso de inversores de frequência na adequação desta altura manométrica requerida e dimensionamento adequado do sistema de captação da água. Os autores concluem que de 15 a 25% da energia pode ser economizada no dimensionamento adequado do sistema de captação e no contexto geral, até 35TWh podem ser economizados com uma regulagem adequada da bomba consistindo no aumento de sua eficiência.

Fu, Cai e Wu (2006) utilizou das equações propostas por Bernier Bourret (1999) no contexto do cálculo do consumo de energia de bombeamento, bem como, na estimativa da energia elétrica de bombeamento consumida. Os autores adotaram a equação para cálculo do rendimento da bomba com inversor e do rendimento do motor e citaram que o uso de metodologias com admiensionalização devem ser observadas para faixas de velocidade específica de 15 a 190 e vazão entre 5 a 200$m^3.s^{-1}$, desde que o erro de cálculo não ultrapasse o valor de 6,55%. Foi avaliado a influência da rotação do eixo do rotor da bomba na economia de energia, bem como, essa variação através do uso de inversor de frequência. Os autores observaram que a economia de energia diminui à medida que as características do sistema exigem maior altura

manométrica e aumenta à medida que o fluxo está mais próximo de seu destino final.

Bene e Hos (2011) demonstram uma técnica para o preenchimento de reservatórios de água com base no menor custo de bombeamento. Foi calculada a altura manométrica de referência a partir de uma vazão de referência e calculou-se um rendimento de referência a partir de uma de um rendimento máximo da bomba. Os dados de potência de entrada da bomba foram calculados a partir das equações propostas por Wallbom-Carlson (1998), em que, a energia específica transmitida ao fluido foi estimada a partir da potência de entrada da bomba. O estudo consistiu em determinar um modelo capaz de reduzir o consumo de energia no enchimento de reservatórios. Foi utilizado o conceito de Série Local Ótima e Série Global Ótima para calcular os valores com base nos tempos de enchimento do reservatório variando o tempo e o rendimento da bomba em relação ao rendimento temporal. Observou-se que a medida que se aumenta o tempo de enchimento há um aumento no consumo de energia específico para todo o processo, tanto na Série Local Ótima quanto na Série Global, contudo esse aumento é, relativamente, baixo. O estudo demonstrou ser um bom parâmetro para o cálculo da economia de energia de bombeamento no enchimento de reservatórios, contudo os autores consideram necessários estudos mais direcionados no que se refere à questão do uso de maior quantidade de reservatórios e na utilização de conjuntos de bombas. A técnica de otimização perde sua eficiência se houver alteração na forma de cobrança tarifária de energia elétrica ou mudança de consumo da bomba.

Burt et al. (2008) desenvolveram estudos para determinar o desempenho de motores sob velocidades variáveis quando são induzidos por velocidades variáveis de rotação e, além disso, fornecer dados suficientes a projetos que envolvam projetos com o uso de bombas. Os autores utilizaram a equação proposta por Wallbom-Carlson (1998) para o cálculo de consumo de energia de

bombeamento, calcularam a eficiência do motor com o inversor e o rendimento do motor, além de analisar a influência do fator de potência no consumo de energia. Observou-se que, comercialmente, existem inversores que melhoram a questão do fator de potência dos motores. A eficiência do inversor de frequência apresenta uma ligeira alteração impactada pelo motor que controla. O inversor pode ser indicado quando o motor é submetido a cargas variáveis.

Este estudo baseou-se na comparação entre os cálculos propostos por Bernier e Bournout (1999), Sarbu e Borza (1998) e Wallbom-Carlson (1998) para o rendimento global do conjunto moto-bomba-inversor e cálculo da potência elétrica ativa do conjunto moto-bomba e consumo específico.

2 MATERIAIS E MÉTODOS

Para o desenvolvimento deste estudo, foram utilizados do conjunto de dados de sete bombas que alimentam sete pivôs centrais e controladas por inversor de frequência.

Foram utilizadas as equações propostas por Bernier e Bourreout (1999), Sarbu e Borza (1998) e Wallbom-Carlson (1998) para o cálculo do rendimento do inversor e o cálculo do rendimento global.

A partir dos dados de rendimento global, foi possível determinar a potência elétrica ativa e o consumo específico de cada bomba e compará-los entre valores obtidos a partir de cada equação. Foi verificado se houve diferença significativa entre o cálculo das equações a partir do teste estatístico F de Snadecor em nível de significância de 5%.

2.1 Caracterização das bombas utilizadas

Neste estudo foram utilizadas sete bombas distintas, cada bomba é utilizada em um pivô central. Dessa forma, tem-se na Tabela 1 as características de cada bomba.

Tabela 1 - Dados das bombas utilizadas.

MBI	Marca/Modelo Bomba	φ (mm)	Q (m³/h)	η	HMT (mca)	Pot. Abs. Eixo (cv)
1	KSB WKL 100/5	252	130	0,75	141	91,21
2	KSB/MEGANORM 125-315	328	138,52	0,78	104	68,64
3	KSB / ETA 100-50/2	328	138,52	0,78	104	68,64
4	KSB / WKL 100/4	252	183	0,72	88	84
5	KSB/MEGANORME 80-400	399	146,29	0,695	70,5	54,96
6	KSB / WKL 100/7	265	187,39	0,71	167	163,78
7	KSB / WKL 40/3	140	20,03	0,6	90	11,39

Fonte: KSB (2017).

Tem-se as seguintes características de cada motor associado ao conjunto moto-bomba.

Tabela 2 - Dados dos motores associados aos conjuntos moto-bombas.

MBI	Motor/Marca	Rotação nominal (rpm)	Nº Polos	Pot. Nominal (cv)	Tensão (v)
1	WEG	1775	4	100	380
2	WEG	1770	4	75	380
3	WEG	1770	4	75	380
4	WEG	1775	4	100	220/380
5	WEG	1775	4	60	220
6	WEG	1780	4	200	380
7	WEG	3540	2	20	220

Fonte: WEG (2017).

2.2 Cálculo da altura manométrica

Pereira, Colombo e Rabelo (2011) utilizou-se de uma equação cúbica para a determinação da HMT com base nos parâmetros da bomba: velocidade angular, vazão, diâmetro do rotor e número de rotores, assim disposta:

$$HMT = \frac{Nr \cdot \varphi^2 \cdot \omega^2}{g}\left[a_1 + b_1 \cdot \frac{Q}{\varphi^3 \cdot \omega} + c_1 \cdot \left(\frac{Q}{\varphi^3 \cdot \omega}\right)^2 + d_1 \cdot \left(\frac{Q}{\varphi^3 \cdot \omega}\right)^3\right] \quad (1)$$

Em que:

HMT: altura manométrica total (mca)

Nr: Número de rotores

φ: diâmetro do rotor (m)

ω: velocidade angular (rad.s^{-1})

g: aceleração da gravidade (m.s^{-2})

Q: vazão da bomba (m³.s^{-1})

a_1, b_1, c_1 e d_1: parâmetros de ajuste da equação

Em que:

$$\omega = \frac{2 \cdot \pi \cdot n}{60} \quad (2)$$

n: Rotação do eixo do rotor (rpm)

2.3 Cálculo do rendimento da bomba

Pereira, Colombo e Rabelo (2011) utilizaram de ajuste de uma equação cúbica para determinar o rendimento da bomba a partir da velocidade angular do eixo do rotor (Equação 3).

$$n_b = a_2 + b_2 \cdot \frac{Q}{\varphi^3 \cdot \omega} + c_2 \cdot \left(\frac{Q}{\varphi^3 \cdot \omega}\right)^2 + d_2 \cdot \left(\frac{Q}{\varphi^3 \cdot \omega}\right)^3 \quad (3)$$

2.4 Cálculo do rendimento do motor

Para o cálculo de rendimento do motor (η_M), foi utilizada a metodologia citada por Bernier e Bournout (1999) e Zhenjun e Shengwey (2009) que baseia-se em uma equação exponencial em função do carregamento do motor:

$$\eta_M = a \cdot \left(1 - e^{b \cdot x}\right) \quad (4)$$

Em que:

η_M : Rendimento do motor (%)

x: carregamento do motor (%)

a e b: parâmetros da equação

Para determinar o carregamento do motor, foi utilizada a potência nominal do motor sobre a potência no eixo associado a cada rotação, assim, o carregamento pode ser expresso da seguinte forma, conforme Zhenjun e Shengwey (2009):

$$x = \frac{Pot_{eixo}}{Pot_{nominal}} \cdot 100 \quad (5)$$

Em que:

x: carregamento do motor (%)

Pot_{eixo}: Potência no eixo (cv)

$Pot_{nominal}$: Potência nominal do motor

Com base na Eq. 1, foi possível através da ferramenta Solver do Microsoft Excel, determinar os parâmetros a e b para cada motor associado a cada bomba (TABELA 5).

2.5 Cálculo do rendimento do inversor a partir de Bernier e Bournout (1999)

Bernier e Bournout (1999) ajustaram uma equação para calcular o rendimento do inversor (η_V):

$$\eta_{VB} = 50{,}87 + 1{,}283 \cdot K - 0{,}0142 \cdot K^2 + 5{,}834 \cdot 10^{-6} \cdot K^3 \quad (6)$$

Em que os valores de K são calculados a partir da equação:

$$K = \frac{n_{atual}}{n_{max}} \cdot 100 \quad (7)$$

Em que:

n_{atual}: rotação do eixo do rotor atualmente (RPM)

n_{max}: rotação máxima do eixo do rotor (RPM)

2.6 Cálculo do rendimento do motor a partir de Wallbom-Carlson (1998)

Wallbom-Carlson (1998) propôs por meio de seus estudos, o η_M a partir dos parâmetros da eficiência do motor a 100% e de um fator de eficiência do inversor de frequência. O fator de eficiência baseia-se na porcentagem de frequência na qual o inversor está atuando, dessa forma, a equação proposta é disposta da seguinte maneira:

$$n_{MW} = Fator\ do\ Inversor\ \cdot Eficiência\ do\ motor\ a\ 100\% \qquad (8)$$

Em que o fator do inversor está relacionado à eficiência do inversor de acordo com a Figura 1.

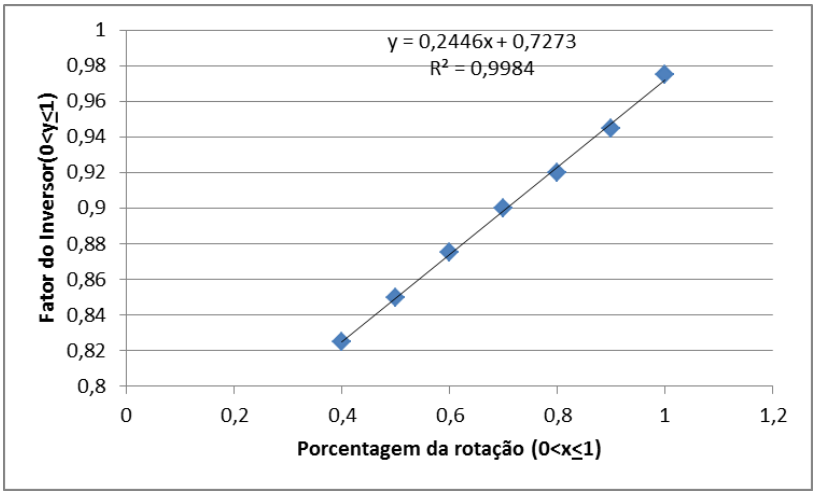
Figura 1 - Tabela de % de Frequência do Motor x eficiência do inversor de frequência.

% of Rated Motor Frequency	VFD Efficiency Factor
100	.97
90	.945
80	.92
70	.90
60	.875
50	.85
40	.825

Fonte: Wallbom-Carlson (1998).

Para ajuste da equação da porcentagem de frequência do motor em relação ao fator de eficiência do inversor, foi utilizada uma regressão linear pelo Microsoft Office Excel.

Figura 2 - Gráfico de regressão da porcentagem de rotação em relação ao fator do inversor proposto para a equação proposta por Wallbom-Carlson (1998).



Fonte: Do autor (2017).

A partir da regressão linear, foi possível calcular o rendimento do conjunto moto-bomba e inversor, e utilizar a equação proposta por Wallbom-Carlson (1998).

2.7 Cálculo do rendimento global a partir de Sarbu & Borza (1998)

Sarbu e Borza (1998) determinaram em seus estudos uma equação na qual é possível calcular o novo rendimento do conjunto moto-bomba a partir do rendimento máximo do conjunto e da rotação máxima da bomba. Essa equação baseou-se nos conceitos de semelhanças mecânicas:

$$\eta_S = 1 - (1 - \eta_1) \cdot \left(\frac{K}{100}\right)^{0,1} \quad (9)$$

Em que:

η_S: rendimento do conjunto moto-bomba atual ($0 < \eta_2 \leq 1$)

η_1: rendimento do conjunto moto-bomba máximo ($0 < \eta_1 \leq 1$)

Com base nos dados do rendimento do motor, foi possível calcular o valor da potência consumida em cada posição e bem como determinar o consumo de energia por volume de água aplicado pelos pivôs centrais.

2.8 Cálculo da potência no eixo

Marchi e Simpson (2012) calculam a potência do eixo pela equação:

$$Pot_{eixo} = \frac{Q \cdot HMT \cdot \rho \cdot g}{\eta_B \cdot 1000} \quad (10)$$

Em que:

Pot_{eixo}: Potência no eixo (kW)

Q: vazão que sai da bomba ($m^3.s^{-1}$)

ρ: peso específico do líquido ($kgf.m^{-3}$)

g: aceleração da gravidade ($m.s^{-2}$)

HMT: altura manométrica total (mca)

η_B: rendimento da bomba variando de $0 < \eta_B \leq 1$

A partir da potência no eixo é possível calcular a potência elétrica ativa total.

2.9 Cálculo da Potência Elétrica Ativa com base Bernier & Bournout (1999)

$$Pot_{ativa} = \frac{Pot_{eixo}}{\eta_M . \eta_V} \quad (11)$$

Em que:

Pot_{ativa} : Potência elétrica ativa (kW)

η_M: rendimento da bomba variando de $0 < \eta_M \leq 1$

Para o cálculo do $kWh.m^{-3}$ foi utilizado à equação:

2.10 Cálculo da Potência Elétrica Ativa com base Wallbom-Carlson (1998)

$$Pot_{ativa} = \frac{Pot_{eixo}}{\eta_{MW}} \quad (12)$$

2.11 Cálculo da Potência Elétrica Ativa com base Sarbu e Borza (1998)

$$Pot_{ativa} = \frac{Pot_{eixo}}{\eta_S} \quad (13)$$

2.12 Cálculo do Consumo Específico

Brar et al. (2017) calcularam o consumo específico utilizando-se da equação:

$$Consumo\ Específico = \frac{Pot_{ativa}}{Q_P} \quad (14)$$

Em que:

Consumo: consumo específico ($kWh.m^{-3}$)

Q_P: Vazão do pivô ($m^3.h^{-1}$)

3 RESULTADOS E DISCUSSÕES

Com base na regressão cúbica da equação, de HMT (Equação 1) e do rendimento da bomba (Equação 3), foi possível ajustar os parâmetros de HMT (TABELA 1) e de rendimento da bomba (TABELA 2). Além disso, foi possível determinar os parâmetros a e b (TABELA 3) para o cálculo do rendimento do motor (Equação 4).

Tabela 3 - Parâmetros da equação cúbica de HMT.

		HMT			
MBI	Bomba	a_1	b_1	c_1	d_1
1	KSB WKL 100/5	0,1689	-1,504	-115,93	-1435,15
2	KSB/MEGANORM 125-315	0,15291	0,395608	-75,0693	-4077,2
3	KSB / ETA 100-50/2	0,122318	-1,40165	-646,388	-103709
4	KSB / WKL 100/4	0,170541	-1,68263	-99,8354	-3355,43
5	KSB/MEGANORME 80-400	0,148543	1,75241	-1165,25	-167429
6	KSB / WKL 100/7	0,170541	-1,72977	-83,4373	-4539,7
7	KSB / WKL 40/3	0,130403	-1,14288	-84,7937	-44345,3

Fonte: Do autor (2017).

Tabela 4 - Parâmetros da equação cúbica de Rendimento da bomba.

		Rendimento			
MBI	Bomba	a_2	b_2	c_2	d_2
1	KSB WKL 100/5	0,03845	119,72	-5951,19	68662,8
2	KSB/MEGANORM 125-315	0,141426	114,0547	-5815,19	81217,91
3	KSB / ETA 100-50/2	0,043408	384,6215	-62495,3	2102652
4	KSB / WKL 100/4	-0,05132	153,2902	-8715,95	127688,2
5	KSB/MEGANORME 80-400	0,268368	188,3909	-10279,4	-2305563
6	KSB / WKL 100/7	-0,01016	134,7565	-6701,34	65023,7
7	KSB / WKL 40/3	0,109118	133,2532	-5947,29	-322243

Fonte: Do autor (2017).

Tabela 5 - Parâmetros de ajuste para a equação de η_M cada bomba utilizada nos pivôs centrais estudados.

MBI	Bomba	A	B
1	KSB WKL 100/5	0,925	7,2075
2	KSB/MEGANORM 125-315	0,919	8,0719
3	KSB / ETA 100-50/2	0,925	7,2075
4	KSB / WKL 100/4	0,925	7,2075
5	KSB/MEGANORME 80-400	0,9156	7,7724
6	KSB / WKL 100/7	0,9326	6,9925
7	KSB / WKL 40/3	0,9115	6,8706

Fonte: Do autor (2017).

Com base nos dados de potência calculados para Bernier e Bournout (1999), Sarbu e Borza (1998) e Wallbom-Carlson (1998) foi possível determinar o consumo $kW.h.m^{-3}$ para cada autor e verificar as diferenças entre os consumos calculados. Além disso, foi calculado o consumo $kW.h.m^{-3}$.

Tabela 6 - Comparação entre os valores calculados de consumo específico em porcentagem média entre os autores e em porcentagem em ralação a pior situação.

Pivô	% MD	%D S/I e MC
1	4,57	13,66
3	5,98	9,81
4	4,18	6,78
5	3,92	20,32
6	4,92	6,78
7	4,77	18,59
8*	4,65	9,54

Legenda: %MD: porcentagem média das diferenças entre os autores %D S/I e PS: Porcentagem das diferenças entre o sem inversor com a porcentagem da pior situação Bernier e Bournout (1999).
*Comparação do pivô 8 foi desconsiderando o Bernier e Bournout (1999) e baseando em Wallbom-Carlson (1998).

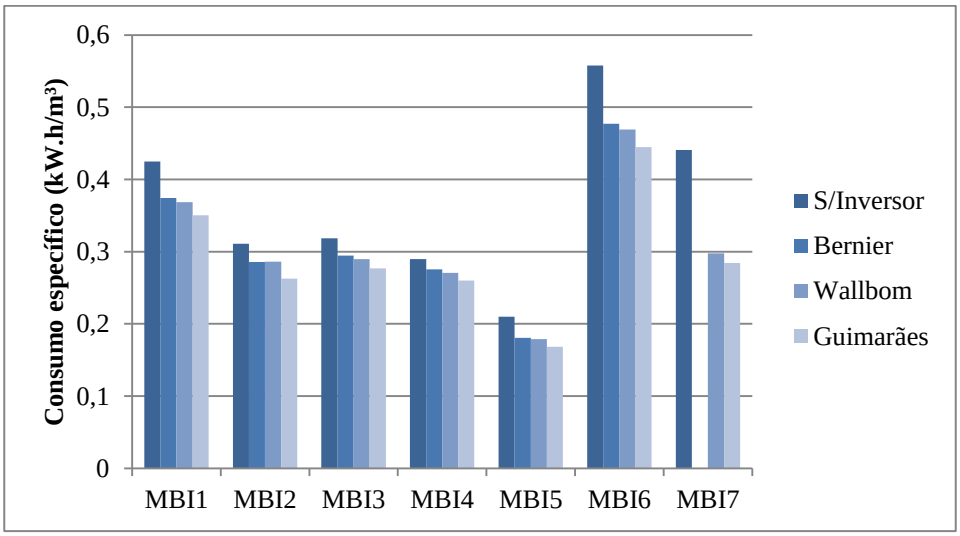
Tabela 7 - Dados do teste estatístico F de Snadecor a nível de significância de 5%.

		B e W		W e S		B e S	
MBI	Kmédio	F	$F_{crítico}$	F	$F_{crítico}$	F	$F_{crítico}$
1	0,92	9,59	3,85	113,69	3,85	184,47	3,85
2	0,78	0,38*	3,85	909,54	3,85	719,45	3,85
3	0,93	6,89	3,85	121,78	3,85	121,78	3,85
4	0,95	24,40	3,85	335,36	3,85	523,99	3,85
5	0,87	18,56	3,85	30,53	3,85	92,31	3,85
6	0,91	8,63	3,85	89,04	3,85	149,26	3,85
7	0,93	3793,36	3,85	271,63	3,85	6318,14	3,85

Fonte: Do autor (2017).

Legenda: B: Bernier, W: Wallbom, S: Sarbu.

Figura 3 - Gráfico de Consumo específico médio em kW.h/m³ para cada pivô estudado e cada equação.



Fonte: Do autor (2017).

A Tabela 4 expressa a média das diferenças em porcentagem entre Bernier e Bournout (1999), Sarbu e Borza (1998) e Wallbom-Carlson (1998) e a porcentagem de diferença entre a pior situação, no caso de comparação com

Bernier e Bournout (1999) para os conjuntos MBI de 1 a 6 e com Wallbom-Carlson (1998) para o conjunto MBI 7.

A porcentagem média das diferenças entre Bernier e Bournout (1999), Sarbu e Borza (1998) e Wallbom-Carlson (1998) apresentou sempre menor que a porcentagem do cálculo mais restritivo em relação ao conjunto moto-bomba sem inversor.

Observou-se também uma pequena diferença entre aos autores, em que Bernier e Bournout (1999) apresenta um maior consumo de específico e Sarbu e Borza (1998) apresenta um menor valor de consumo específico.

O teste estatístico F de Snadecor em nível de significância de 5% demonstrou que há diferença significativa na comparação entre todas as equações com exceção ao conjunto MBI2, em que, a equação de Wallbom-Carlson (1998) e a equação de Bernier e Bournout (1999) não apresentaram diferença significativa. Observou-se que o valor de K médio, ou seja, o valor da rotação do eixo do motor em relação a rotação nominal, foi o menor valor quando comparado aos demais conjunto moto-bomba-inversor, em que este pode ser uma razão para que não haja diferença significativa entre as equações.

Tem-se através dos dados uma pequena diferença Bernier e Bournout (1999), Sarbu e Borza (1998) e Wallbom-Carlson (1998), em que, ao utilizar Bernier e Bournout (1999), o cálculo considera a pior situação, ou seja, exige uma maior potência, pois considera o rendimento total menor. A maior potência impacta em um maior consumo de energia.

No caso de uma situação em que considera-se uma melhor situação, pode-se utilizar Sarbu e Borza (1998), pois este considera um maior rendimento e consequentemente uma menor necessidade de potência, e um menor consumo de energia. No caso Wallbom-Carlson (1998), este exige uma menor quantidade de dados e apresenta ser mais equilibrado em comparação Bournout (1999) e Sarbu e Borza (1998).

4 CONCLUSÃO

A equação de Bernier e Bournout (1999) compõem-se de mais elementos para os cálculos, assim, uma equação que apresenta menor rendimento e exige maior potência e consequentemente maior consumo.

Sarbu e Borza (1998) demonstrou ter menor quantidade de parâmetros na equação, apresentando maior valor de rendimento e menor potência no cálculo de sua equação e apresentando menor consumo.

Wallbom-Carlson (1998) demonstrou ser mais equilibrado em comparação a Bernier e Bournout (1999) e Sarbu e Borza (1998).

Há diferença significativa entre todas equações, com exceção, da equação Bernier e Bournout (1999) e Wallbom-Carlson (1998) no conjunto MBI2, em que, essa não diferença pode ser efeito da baixa rotação em que esse conjunto atua.

REFERÊNCIAS

BENE, J. G.; HŐS, C. J. Finding least-cost pump schedules for reservoir filling with a variable speed pump. **Journal of Water Resources Planning and Management**, Reston, v. 138, n. 6, p. 682-686, 2011.

BENIER, M. A.; BOURRET, B. Pumping energy and variable frequency drivers. **ASHRAE Journal**, Atlanta, v. 12, p. 37-40, 1999.

BRAR, D. et al. Energy conservation using variable-frequency drives for center-pivot irrigation. In: ANNUAL CENTRAL PLAINS IRRIGATION CONFERENCE, 29., 2017, Burlington. **Proceedings...** Burlington, 2017. p. 155-166.

BRASIL. Agência Nacional de Energia Elétrica. **Geração distribuída**. Brasília, DF, 2017. Disponível em: <http://www2.aneel.gov.br/scg/gd/GD_Classe.asp>. Acesso em: 10 set. 2017.

BURT, C. M. et al. Electric motor efficiency under variable frequencies and loads. **Journal of Irrigation and Drainage Engineering**, New York, v. 134, n. 2, p. 129-136, 2008.

FU, Y.; CAI, Y.; WU, K. E. Q. I. Forecasting the energy-saving benefits of variable-speed pumps. **Task Quarterly**, Gdansk, v. 10, n. 1, p. 27-33, 2006.

KSB. **Catálogo de produtos/Irrigação/Irrigação por aspersão**. Disponível em: <http://www.ksb.com.br/ksb-br-pt/tipos.php?codtipo=1&codgrupo=1&codaplicacao=47>. Acesso em: 9 set. 2017.

MARCHI, A.; SIMPSON, A. R. Correction of the EPANET inaccuracy in computing the efficiency of variable speed pumps. **Journal of Water Resources Planning and Management**, Reston, v. 139, n. 4, p. 456-459, 2012.

PEREIRA, P. H. C.; COLOMBO, A.; RABELO, G. F. O uso da rede Modbus em sistema de irrigação por pivô central auxiliando na eficiência energética através do controle do conjunto moto-bomba. In: LATIN-AMERICAN CONGRESS ON ELECTRICITY GENERATION AND TRANSMISSION: CLAGTEE, 9., 2011, Mar del Plata. **Proceedings…** Guaratinguetá: FUNDEB, 2011. v. 1, p. 1-7.

SARBU, I.; BORZA, I. Energetic optimization of water pumping in distribution systems. **Periodica Polytechnica.** Mechanical Engineering, Budapest, v. 42, n. 2, p. 141-152, 1998.

SHANKAR, V. K. A. et al. A comprehensive review on energy efficiency enhancement initiatives in centrifugal pumping system. **Applied Energy**, London, v. 181, p. 495-513, 2016.

SINGH, H.; MISHRA, D.; NAHAR, N. M. Energy use pattern in production agriculture of a typical village in arid zone, India: part I. **Energy Conversion and Management**, Oxford, v. 43, n. 16, p. 2275-2286, 2002.

WAIDE, P.; BRUNNER, C. U. **Energy-efficiency policy opportunities for electric motor-driven systems**. Paris: IEA, 2011. 132 p. (International Energy Agency Energy Efficiency Series).

WALLBOM-CARLSON, A. Energy comparison. VFD vs. on-off controlled pumping stations. **Scientific Impeller**, Stockholm, v. 136, n. 5, p. 29-32, 1998.

WEG. **Catálogo de motores**. Disponível em: <http://catalogo.weg.com.br/tec_cat/tech_motor_sel_web.asp>. Acesso em: 9 set. 2017.

ZHENJUN, M. A.; WANG, S. Energy efficient control of variable speed pumps in complex building central air-conditioning systems. **Energy and Buildings**, Lausanne, v. 41, n. 2, p. 197-205, 2009.

ANEXO A - CÓDIGO DE COMUNICAÇÃO DO RASPBERRY PI COM O ARDUINO

```
import os
import csv
from tkinter import *
import tkinter.messagebox
import serial
import time

ser = serial.Serial('/dev/ttyACM0', 115200, timeout=0)
# Definir a porta na qual está conectado o Arduino (no caso, COM9)

class GUIFramework(Frame):

 def __init__(self,master=None):

   self.root = Tk()
   Frame.__init__(self,master)
   self.master.title("Controle do Pivo")
   self.grid(padx=10,pady=10)
   self.CreateWidgets()
```

```
 def CreateWidgets(self):



   self.Entrada0 = Entry(self, width=4) # Campo de entrada 1
   self.Entrada0.grid(row=0, column=1)

   self.Entrada01 = Entry(self, width=4) # Campo de entrada 1
   self.Entrada01.grid(row=0, column=3)

   self.Entrada02 = Entry(self, width=4) # Campo de entrada 1
   self.Entrada02.grid(row=1, column=1)

   self.Entrada1 = Entry(self, width=4) # Campo de entrada 1
   self.Entrada1.grid(row=2, column=1)
   self.Entrada2 = Entry(self, width=4)# Campo de entrada 2
   self.Entrada2.grid(row=3, column=1)
   self.Entrada3 = Entry(self, width=4)# Campo de entrada 2
   self.Entrada3.grid(row=4, column=1)
   self.Entrada4 = Entry(self, width=4)# Campo de entrada 2
   self.Entrada4.grid(row=5, column=1)
   self.Entrada5 = Entry(self, width=4)# Campo de entrada 2
```

```
self.Entrada5.grid(row=6, column=1)
self.Entrada6 = Entry(self, width=4)# Campo de entrada 2
self.Entrada6.grid(row=7, column=1)
self.Entrada7 = Entry(self, width=4)# Campo de entrada 2
self.Entrada7.grid(row=8, column=1)
self.Entrada8 = Entry(self, width=4)# Campo de entrada 2
self.Entrada8.grid(row=9, column=1)
self.Entrada9 = Entry(self, width=4)# Campo de entrada 2
self.Entrada9.grid(row=10, column=1)
self.Entrada10 = Entry(self, width=4)# Campo de entrada 2
self.Entrada10.grid(row=11, column=1)
self.Entrada11 = Entry(self, width=4)# Campo de entrada 2
self.Entrada11.grid(row=12, column=1)
self.Entrada12 = Entry(self, width=4)# Campo de entrada 2
self.Entrada12.grid(row=13, column=1)
self.Entrada13 = Entry(self, width=4)# Campo de entrada 2
self.Entrada13.grid(row=14, column=1)
self.Entrada14 = Entry(self, width=4)# Campo de entrada 2
self.Entrada14.grid(row=15, column=1)
self.Entrada15 = Entry(self, width=4)# Campo de entrada 2
self.Entrada15.grid(row=16, column=1)
self.Entrada16 = Entry(self, width=4)# Campo de entrada 2
```

```
    self.Entrada16.grid(row=17, column=1)
    self.Entrada17 = Entry(self, width=4)# Campo de entrada 2
    self.Entrada17.grid(row=18, column=1)
    self.Entrada18 = Entry(self, width=4)# Campo de entrada 2
    self.Entrada18.grid(row=19, column=1)
    self.Entrada19 = Entry(self, width=4)# Campo de entrada 2
    self.Entrada19.grid(row=2, column=3)
    self.Entrada20 = Entry(self, width=4)# Campo de entrada 2
    self.Entrada20.grid(row=3, column=3)
    self.Entrada21 = Entry(self, width=4)# Campo de entrada 2
    self.Entrada21.grid(row=4, column=3)
    self.Entrada22 = Entry(self, width=4)# Campo de entrada 2
    self.Entrada22.grid(row=5, column=3)
    self.Entrada23 = Entry(self, width=4)# Campo de entrada 2
    self.Entrada23.grid(row=6, column=3)
    self.Entrada24 = Entry(self, width=4)# Campo de entrada 2
    self.Entrada24.grid(row=7, column=3)
    self.Entrada25 = Entry(self, width=4)# Campo de entrada 2
    self.Entrada25.grid(row=8, column=3)
    self.Entrada26 = Entry(self, width=4)# Campo de entrada 2
    self.Entrada26.grid(row=9, column=3)
    self.Entrada27 = Entry(self, width=4)# Campo de entrada 2
```

```
self.Entrada27.grid(row=10, column=3)
self.Entrada28 = Entry(self, width=4)# Campo de entrada 2
self.Entrada28.grid(row=11, column=3)
self.Entrada29 = Entry(self, width=4)# Campo de entrada 2
self.Entrada29.grid(row=12, column=3)
self.Entrada30 = Entry(self, width=4)# Campo de entrada 2
self.Entrada30.grid(row=13, column=3)
self.Entrada31 = Entry(self, width=4)# Campo de entrada 2
self.Entrada31.grid(row=14, column=3)
self.Entrada32 = Entry(self, width=4)# Campo de entrada 2
self.Entrada32.grid(row=15, column=3)
self.Entrada33 = Entry(self, width=4)# Campo de entrada 2
self.Entrada33.grid(row=16, column=3)
self.Entrada34 = Entry(self, width=4)# Campo de entrada 2
self.Entrada34.grid(row=17, column=3)
self.Entrada35 = Entry(self, width=4)# Campo de entrada 2
self.Entrada35.grid(row=18, column=3)
self.Entrada36 = Entry(self, width=4)# Campo de entrada 2
self.Entrada36.grid(row=19, column=3)
```

```
    # Note que aqui utilizando o gerenciador .grid, no qual especificamos as
linhas e colunas onde desejamos posicionar os widgets.


    #Label(self, text = "CONTROLE PIVO", justify = RIGHT).grid (row=0,
column=1)

    Label(self, text = "Per.Ult.Torre", justify = RIGHT).grid (row=0, column=0)

    Label(self, text = "Vel.Ult.Torre", justify = RIGHT).grid (row=0, column=2)

    Label(self, text = "Reg. Veloc. (%)", justify = RIGHT).grid (row=1,
column=0)


    Label(self, text = "10 graus", justify = RIGHT).grid (row=2, column=0)

    Label(self, text = "20 graus", justify = LEFT).grid (row=3, column=0)

    Label(self, text = "30 graus", justify = LEFT).grid (row=4, column=0)

    Label(self, text = "40 graus", justify = LEFT).grid (row=5, column=0)

    Label(self, text = "50 graus", justify = LEFT).grid (row=6, column=0)

    Label(self, text = "60 graus", justify = LEFT).grid (row=7, column=0)

    Label(self, text = "70 graus", justify = LEFT).grid (row=8, column=0)

    Label(self, text = "80 graus", justify = LEFT).grid (row=9, column=0)

    Label(self, text = "90 graus", justify = LEFT).grid (row=10, column=0)

    Label(self, text = "100 graus", justify = LEFT).grid (row=11, column=0)

    Label(self, text = "110 graus", justify = LEFT).grid (row=12, column=0)

    Label(self, text = "120 graus", justify = LEFT).grid (row=13, column=0)

    Label(self, text = "130 graus", justify = LEFT).grid (row=14, column=0)
```

```
Label(self, text = "140 graus", justify = LEFT).grid (row=15, column=0)
Label(self, text = "150 graus", justify = LEFT).grid (row=16, column=0)
Label(self, text = "160 graus", justify = LEFT).grid (row=17, column=0)
Label(self, text = "170 graus", justify = LEFT).grid (row=18, column=0)
Label(self, text = "180 graus", justify = LEFT).grid (row=19, column=0)

Label(self, text = "190 graus", justify = LEFT).grid (row=2, column=2)
Label(self, text = "200 graus", justify = LEFT).grid (row=3, column=2)
Label(self, text = "210 graus", justify = LEFT).grid (row=4, column=2)
Label(self, text = "220 graus", justify = LEFT).grid (row=5, column=2)
Label(self, text = "230 graus", justify = LEFT).grid (row=6, column=2)
Label(self, text = "240 graus", justify = LEFT).grid (row=7, column=2)
Label(self, text = "250 graus", justify = LEFT).grid (row=8, column=2)
Label(self, text = "260 graus", justify = LEFT).grid (row=9, column=2)
Label(self, text = "270 graus", justify = LEFT).grid (row=10, column=2)
Label(self, text = "280 graus", justify = LEFT).grid (row=11, column=2)
Label(self, text = "290 graus", justify = LEFT).grid (row=12, column=2)
Label(self, text = "300 graus", justify = LEFT).grid (row=13, column=2)
Label(self, text = "310 graus", justify = LEFT).grid (row=14, column=2)
Label(self, text = "320 graus", justify = LEFT).grid (row=15, column=2)
Label(self, text = "330 graus", justify = LEFT).grid (row=16, column=2)
Label(self, text = "340 graus", justify = LEFT).grid (row=17, column=2)
```

```
    Label(self, text = "350 graus", justify = LEFT).grid (row=18, column=2)

    Label(self, text = "360 graus", justify = LEFT).grid (row=19, column=2)




    self.Write

    Botao_Grava = Button(self, text=" Grava ", command=self.Write) # Botão
para gravar os valores na EEPROM do Arduino

    Botao_Grava.grid(row=20, column=0)

    Botao_Le = Button(self, text="  Lê  ", command=self.Read)  # Botão para ler
os valores da EEPROM do Arduino

    Botao_Le.grid(row=20, column=1)


    Botao_grava_csv = Button(self, text="Grava CSV",
command=self.WriteCSV)  # Botão para ler os valores da EEPROM do Arduino

    Botao_grava_csv.grid(row=20, column=2)


    Botao_le_csv = Button(self, text="Lê CSV", command=self.ReadCSV)  #
Botão para ler os valores da EEPROM do Arduino

    Botao_le_csv.grid(row=20, column=3)
```

```
def WriteCSV(self): # Salva os valores em formato .csv

  filename = tkinter.filedialog.asksaveasfilename()
  output_file = open(filename+".csv", 'w', newline='')
  data = csv.writer(output_file)

  valor = self.Entrada1.get()
  send_value = (chr(ord('a') ) + (valor))
  data.writerow([send_value])
  valor = self.Entrada2.get()
  send_value = (chr(ord('a')+1 ) + (valor))
  data.writerow([send_value])
  valor = self.Entrada3.get()
  send_value = (chr(ord('a')+2 ) + (valor))
  data.writerow([send_value])
  valor = self.Entrada4.get()
  send_value = (chr(ord('a')+3 ) + (valor))
  data.writerow([send_value])
  valor = self.Entrada5.get()
  send_value = (chr(ord('a')+4 ) + (valor))
  data.writerow([send_value])
  valor = self.Entrada6.get()
```

```
    send_value = (chr(ord('a')+5 ) + (valor))
    data.writerow([send_value])
    valor = self.Entrada7.get()
    send_value = (chr(ord('a')+6 ) + (valor))
    data.writerow([send_value])
    valor = self.Entrada8.get()
    send_value = (chr(ord('a')+7 ) + (valor))
    data.writerow([send_value])
    valor = self.Entrada9.get()
    send_value = (chr(ord('a')+8 ) + (valor))
    data.writerow([send_value])
    valor = self.Entrada10.get()
    send_value = (chr(ord('a')+9 ) + (valor))
    data.writerow([send_value])
    valor = self.Entrada11.get()
    send_value = (chr(ord('a')+10 ) + (valor))
    data.writerow([send_value])
    valor = self.Entrada12.get()
    send_value = (chr(ord('a')+11 ) + (valor))
    data.writerow([send_value])
    valor = self.Entrada13.get()
    send_value = (chr(ord('a')+12 ) + (valor))
```

```
data.writerow([send_value])
valor = self.Entrada14.get()
send_value = (chr(ord('a')+13 ) + (valor))
data.writerow([send_value])
valor = self.Entrada15.get()
send_value = (chr(ord('a')+14 ) + (valor))
data.writerow([send_value])
valor = self.Entrada16.get()
send_value = (chr(ord('a')+15) + (valor))
data.writerow([send_value])
valor = self.Entrada17.get()
send_value = (chr(ord('a')+16 ) + (valor))
data.writerow([send_value])
valor = self.Entrada18.get()
send_value = (chr(ord('a')+17 ) + (valor))
data.writerow([send_value])
valor = self.Entrada19.get()
send_value = (chr(ord('a')+18 ) + (valor))
data.writerow([send_value])
valor = self.Entrada20.get()
send_value = (chr(ord('a')+19 ) + (valor))
data.writerow([send_value])
```

```
    valor = self.Entrada21.get()
    send_value = (chr(ord('a')+20 ) + (valor))
    data.writerow([send_value])
    valor = self.Entrada22.get()
    send_value = (chr(ord('a')+21 ) + (valor))
    data.writerow([send_value])
    valor = self.Entrada23.get()
    send_value = (chr(ord('a')+22 ) + (valor))
    data.writerow([send_value])
    valor = self.Entrada24.get()
    send_value = (chr(ord('a')+23) + (valor))
    data.writerow([send_value])
    valor = self.Entrada25.get()
    send_value = (chr(ord('a')+24 ) + (valor))
    data.writerow([send_value])
    valor = self.Entrada26.get()
    send_value = (chr(ord('a')+25 ) + (valor))
    data.writerow([send_value])
    valor = self.Entrada27.get()
    send_value = (chr(ord('a')+26 ) + (valor))
    data.writerow([send_value])
    valor = self.Entrada28.get()
```

```
send_value = (chr(ord('a')+27 ) + (valor))
data.writerow([send_value])
valor = self.Entrada29.get()
send_value = (chr(ord('a')+28 ) + (valor))
data.writerow([send_value])
valor = self.Entrada30.get()
send_value = (chr(ord('a')+29 ) + (valor))
data.writerow([send_value])
valor = self.Entrada31.get()
send_value = (chr(ord('a')+30 ) + (valor))
data.writerow([send_value])
valor = self.Entrada32.get()
send_value = (chr(ord('a')+31 ) + (valor))
data.writerow([send_value])
valor = self.Entrada33.get()
send_value = (chr(ord('a')+32 ) + (valor))
data.writerow([send_value])
valor = self.Entrada34.get()
send_value = (chr(ord('a')+33 ) + (valor))
data.writerow([send_value])
valor = self.Entrada35.get()
send_value = (chr(ord('a')+34 ) + (valor))
```

```
    data.writerow([send_value])
    valor = self.Entrada36.get()
    send_value = (chr(ord('a')+35 ) + (valor))
    data.writerow([send_value])




    tkinter.messagebox.showwarning("Upload", "O Arquivo foi salvo com
sucesso!")


  def ReadCSV(self): # Lê os valores do arquivo .csv


    self.Entrada1.delete(0, END) # Limpa os valores dos campos de entrada
    self.Entrada2.delete(0, END)
    self.Entrada3.delete(0, END)
    self.Entrada4.delete(0, END)
    self.Entrada5.delete(0, END)
    self.Entrada6.delete(0, END)
    self.Entrada7.delete(0, END)
    self.Entrada8.delete(0, END)
    self.Entrada9.delete(0, END)
    self.Entrada10.delete(0, END)
```

```
self.Entrada11.delete(0, END)
self.Entrada12.delete(0, END)
self.Entrada13.delete(0, END)
self.Entrada14.delete(0, END)
self.Entrada15.delete(0, END)
self.Entrada16.delete(0, END)
self.Entrada17.delete(0, END)
self.Entrada18.delete(0, END)
self.Entrada19.delete(0, END)
self.Entrada20.delete(0, END)
self.Entrada21.delete(0, END)
self.Entrada22.delete(0, END)
self.Entrada23.delete(0, END)
self.Entrada24.delete(0, END)
self.Entrada25.delete(0, END)
self.Entrada26.delete(0, END)
self.Entrada27.delete(0, END)
self.Entrada28.delete(0, END)
self.Entrada29.delete(0, END)
self.Entrada30.delete(0, END)
self.Entrada31.delete(0, END)
self.Entrada32.delete(0, END)
```

```
    self.Entrada33.delete(0, END)
    self.Entrada34.delete(0, END)
    self.Entrada35.delete(0, END)
    self.Entrada36.delete(0, END)




    filename = tkinter.filedialog.askopenfilename()
    input_file = open(filename, 'r', newline='')
    data = csv.reader(input_file, delimiter='\t', quoting=csv.QUOTE_NONE)

    line = next(data)
    var = line[0]
    self.Entrada1.insert (END, var[1:])
    line = next(data)
    var = line[0]
    self.Entrada2.insert (END, var[1:])
    line = next(data)
    var = line[0]
    self.Entrada3.insert (END, var[1:])
    line = next(data)
```

```
var = line[0]
self.Entrada4.insert (END, var[1:])
line = next(data)
var = line[0]
self.Entrada5.insert (END, var[1:])
line = next(data)
var = line[0]
self.Entrada6.insert (END, var[1:])
line = next(data)
var = line[0]
self.Entrada7.insert (END, var[1:])
line = next(data)
var = line[0]
self.Entrada8.insert (END, var[1:])
line = next(data)
var = line[0]
self.Entrada9.insert (END, var[1:])
line = next(data)
var = line[0]
self.Entrada10.insert (END, var[1:])
line = next(data)
var = line[0]
```

```
    self.Entrada11.insert (END, var[1:])
    line = next(data)
    var = line[0]
    self.Entrada12.insert (END, var[1:])
    line = next(data)
    var = line[0]
    self.Entrada13.insert (END, var[1:])
    line = next(data)
    var = line[0]
    self.Entrada14.insert (END, var[1:])
    line = next(data)
    var = line[0]
    self.Entrada15.insert (END, var[1:])
    line = next(data)
    var = line[0]
    self.Entrada16.insert (END, var[1:])
    line = next(data)
    var = line[0]
    self.Entrada17.insert (END, var[1:])
    line = next(data)
    var = line[0]
    self.Entrada18.insert (END, var[1:])
```

```
line = next(data)
var = line[0]
self.Entrada19.insert (END, var[1:])
line = next(data)
var = line[0]
self.Entrada20.insert (END, var[1:])
line = next(data)
var = line[0]
self.Entrada21.insert (END, var[1:])
line = next(data)
var = line[0]
self.Entrada22.insert (END, var[1:])
line = next(data)
var = line[0]
self.Entrada23.insert (END, var[1:])
line = next(data)
var = line[0]
self.Entrada24.insert (END, var[1:])
line = next(data)
var = line[0]
self.Entrada25.insert (END, var[1:])
line = next(data)
```

```
    var = line[0]
    self.Entrada26.insert (END, var[1:])
    line = next(data)
    var = line[0]
    self.Entrada27.insert (END, var[1:])
    line = next(data)
    var = line[0]
    self.Entrada28.insert (END, var[1:])
    line = next(data)
    var = line[0]
    self.Entrada29.insert (END, var[1:])
    line = next(data)
    var = line[0]
    self.Entrada30.insert (END, var[1:])
    line = next(data)
    var = line[0]
    self.Entrada31.insert (END, var[1:])
    line = next(data)
    var = line[0]
    self.Entrada32.insert (END, var[1:])
    line = next(data)
    var = line[0]
```

```
    self.Entrada33.insert (END, var[1:])

    line = next(data)

    var = line[0]

    self.Entrada34.insert (END, var[1:])

    line = next(data)

    var = line[0]

    self.Entrada35.insert (END, var[1:])

    line = next(data)

    var = line[0]

    self.Entrada36.insert (END, var[1:])


  def Write(self): #Através da bibliteca serial, envia os valores dos campos de
entrada para o Arduino


    temp = 0

     #verificar um erro do campo

    if self.Entrada1.get() == ("") or self.Entrada2.get() == "" or self.Entrada3.get()
== "" or self.Entrada4.get() == "" or self.Entrada5.get() == "" or
self.Entrada6.get() == "" or self.Entrada7.get() == "" or self.Entrada8.get() == ""
or self.Entrada9.get() == "" or self.Entrada10.get() == "" or self.Entrada11.get()
== "" or self.Entrada12.get() == "" or self.Entrada13.get() == "" or
self.Entrada14.get() == "" or self.Entrada15.get() == "" or self.Entrada16.get()
== "" or self.Entrada17.get() == "" or self.Entrada18.get() == "" or
self.Entrada19.get() == "" or self.Entrada20.get() == "" or self.Entrada21.get()
== "" or self.Entrada22.get() == "" or self.Entrada23.get() == "" or
self.Entrada24.get() == "" or self.Entrada25.get() == "" or self.Entrada26.get()
```

```
== "" or self.Entrada27.get() == "" or self.Entrada28.get() == "" or
self.Entrada29.get() == "" or self.Entrada30.get() == "" or self.Entrada31.get()
== "" or self.Entrada32.get() == "" or self.Entrada33.get() == "" or
self.Entrada34.get() == "" or self.Entrada35.get() == "" or self.Entrada36.get()
== "":

    tkinter.messagebox.showwarning("Aviso", "Campo vazio") #Se os campos
estão vazios, exibe um aviso

  else:

    ser.write((self.Entrada1.get()).encode()) # Envia os valores precedidos por
letras

    self.Entrada1.delete(0, END) # Limpa os valores dos campos de entrada



    t = time.localtime()

    auxtemp = t[4] + temp  #fica em um loop pelo tempo de giro do pivo

    if auxtemp > 59:

      auxtemp = 60 - t[4]

      auxtemp = temp - auxtemp

    while t[4] != auxtemp:

      x = 1

      t = time.localtime()

    ser.write((self.Entrada2.get()).encode()) # para que o programa no Arduino
identifique a informação

    self.Entrada2.delete(0, END) # Limpa os valores dos campos de entrada



    t = time.localtime()
```

```
auxtemp = t[4] + temp  #fica em um loop pelo tempo de giro do pivo
if auxtemp > 59:
  auxtemp = 60 - t[4]
  auxtemp = temp - auxtemp
while t[4] != auxtemp:
  x = 1
  t = time.localtime()

ser.write((self.Entrada3.get()).encode())
self.Entrada3.delete(0, END) # Limpa os valores dos campos de entrada

t = time.localtime()
auxtemp = t[4] + temp  #fica em um loop pelo tempo de giro do pivo
if auxtemp > 59:
  auxtemp = 60 - t[4]
  auxtemp = temp - auxtemp
while t[4] != auxtemp:
  x = 1
  t = time.localtime()

ser.write((self.Entrada4.get()).encode())
self.Entrada4.delete(0, END) # Limpa os valores dos campos de entrada
```

```
t = time.localtime()
auxtemp = t[4] + temp  #fica em um loop pelo tempo de giro do pivo
if auxtemp > 59:
  auxtemp = 60 - t[4]
  auxtemp = temp - auxtemp
while t[4] != auxtemp:
  x = 1
  t = time.localtime()

ser.write((self.Entrada5.get()).encode())
self.Entrada5.delete(0, END) # Limpa os valores dos campos de entrada

t = time.localtime()
auxtemp = t[4] + temp  #fica em um loop pelo tempo de giro do pivo
if auxtemp > 59:
  auxtemp = 60 - t[4]
  auxtemp = temp - auxtemp
while t[4] != auxtemp:
  x = 1
  t = time.localtime()
ser.write((self.Entrada6.get()).encode())
```

```
self.Entrada6.delete(0, END) # Limpa os valores dos campos de entrada


t = time.localtime()
auxtemp = t[4] + temp  #fica em um loop pelo tempo de giro do pivo
if auxtemp > 59:
  auxtemp = 60 - t[4]
  auxtemp = temp - auxtemp
while t[4] != auxtemp:
  x = 1
  t = time.localtime()
ser.write((self.Entrada7.get()).encode())
self.Entrada7.delete(0, END) # Limpa os valores dos campos de entrada



t = time.localtime()
auxtemp = t[4] + temp  #fica em um loop pelo tempo de giro do pivo
if auxtemp > 59:
  auxtemp = 60 - t[4]
  auxtemp = temp - auxtemp
while t[4] != auxtemp:
  x = 1
  t = time.localtime()
```

```
ser.write((self.Entrada8.get()).encode())
self.Entrada8.delete(0, END) # Limpa os valores dos campos de entrada



t = time.localtime()
auxtemp = t[4] + temp  #fica em um loop pelo tempo de giro do pivo
if auxtemp > 59:
  auxtemp = 60 - t[4]
  auxtemp = temp - auxtemp
while t[4] != auxtemp:
  x = 1
  t = time.localtime()
ser.write((self.Entrada9.get()).encode())
self.Entrada9.delete(0, END) # Limpa os valores dos campos de entrada

t = time.localtime()
auxtemp = t[4] + temp  #fica em um loop pelo tempo de giro do pivo
if auxtemp > 59:
  auxtemp = 60 - t[4]
  auxtemp = temp - auxtemp
```

```
while t[4] != auxtemp:
  x = 1
  t = time.localtime()
ser.write((self.Entrada10.get()).encode())
self.Entrada10.delete(0, END) # Limpa os valores dos campos de entrada

t = time.localtime()
auxtemp = t[4] + temp  #fica em um loop pelo tempo de giro do pivo
if auxtemp > 59:
  auxtemp = 60 - t[4]
  auxtemp = temp - auxtemp
while t[4] != auxtemp:
  x = 1
  t = time.localtime()
ser.write((self.Entrada11.get()).encode())
self.Entrada11.delete(0, END) # Limpa os valores dos campos de entrada

t = time.localtime()
auxtemp = t[4] + temp  #fica em um loop pelo tempo de giro do pivo
if auxtemp > 59:
  auxtemp = 60 - t[4]
  auxtemp = temp - auxtemp
```

```
while t[4] != auxtemp:
  x = 1
  t = time.localtime()
ser.write((self.Entrada12.get()).encode())
self.Entrada12.delete(0, END) # Limpa os valores dos campos de entrada


t = time.localtime()
auxtemp = t[4] + temp  #fica em um loop pelo tempo de giro do pivo
if auxtemp > 59:
  auxtemp = 60 - t[4]
  auxtemp = temp - auxtemp
while t[4] != auxtemp:
  x = 1
  t = time.localtime()
ser.write((self.Entrada13.get()).encode())
self.Entrada13.delete(0, END) # Limpa os valores dos campos de entrada



t = time.localtime()
auxtemp = t[4] + temp  #fica em um loop pelo tempo de giro do pivo
if auxtemp > 59:
  auxtemp = 60 - t[4]
```

```
  auxtemp = temp - auxtemp
while t[4] != auxtemp:
  x = 1
  t = time.localtime()
ser.write((self.Entrada14.get()).encode())
self.Entrada14.delete(0, END) # Limpa os valores dos campos de entrada



t = time.localtime()
auxtemp = t[4] + temp  #fica em um loop pelo tempo de giro do pivo
if auxtemp > 59:
  auxtemp = 60 - t[4]
  auxtemp = temp - auxtemp
while t[4] != auxtemp:
  x = 1
  t = time.localtime()
ser.write((self.Entrada15.get()).encode())
self.Entrada15.delete(0, END) # Limpa os valores dos campos de entrada


t = time.localtime()
auxtemp = t[4] + temp  #fica em um loop pelo tempo de giro do pivo
if auxtemp > 59:
```

```
  auxtemp = 60 - t[4]
  auxtemp = temp - auxtemp
while t[4] != auxtemp:
  x = 1
  t = time.localtime()
ser.write((self.Entrada16.get()).encode())
self.Entrada16.delete(0, END) # Limpa os valores dos campos de entrada

t = time.localtime()
auxtemp = t[4] + temp  #fica em um loop pelo tempo de giro do pivo
if auxtemp > 59:
  auxtemp = 60 - t[4]
  auxtemp = temp - auxtemp
while t[4] != auxtemp:
  x = 1
  t = time.localtime()
ser.write((self.Entrada17.get()).encode())
self.Entrada17.delete(0, END) # Limpa os valores dos campos de entrada


t = time.localtime()
auxtemp = t[4] + temp  #fica em um loop pelo tempo de giro do pivo
```

```
if auxtemp > 59:
  auxtemp = 60 - t[4]
  auxtemp = temp - auxtemp
while t[4] != auxtemp:
  x = 1
  t = time.localtime()
ser.write((self.Entrada18.get()).encode())
self.Entrada18.delete(0, END) # Limpa os valores dos campos de entrada


t = time.localtime()
auxtemp = t[4] + temp  #fica em um loop pelo tempo de giro do pivo
if auxtemp > 59:
  auxtemp = 60 - t[4]
  auxtemp = temp - auxtemp
while t[4] != auxtemp:
  x = 1
  t = time.localtime()
ser.write((self.Entrada19.get()).encode())
self.Entrada19.delete(0, END) # Limpa os valores dos campos de entrada



t = time.localtime()
```

```
auxtemp = t[4] + temp  #fica em um loop pelo tempo de giro do pivo
if auxtemp > 59:
  auxtemp = 60 - t[4]
  auxtemp = temp - auxtemp
while t[4] != auxtemp:
  x = 1
  t = time.localtime()
ser.write((self.Entrada20.get()).encode())
self.Entrada20.delete(0, END) # Limpa os valores dos campos de entrada


t = time.localtime()
auxtemp = t[4] + temp  #fica em um loop pelo tempo de giro do pivo
if auxtemp > 59:
  auxtemp = 60 - t[4]
  auxtemp = temp - auxtemp
while t[4] != auxtemp:
  x = 1
  t = time.localtime()
ser.write((self.Entrada21.get()).encode())
self.Entrada21.delete(0, END) # Limpa os valores dos campos de entrada
```

```
t = time.localtime()
auxtemp = t[4] + temp  #fica em um loop pelo tempo de giro do pivo
if auxtemp > 59:
  auxtemp = 60 - t[4]
  auxtemp = temp - auxtemp
while t[4] != auxtemp:
  x = 1
  t = time.localtime()
ser.write((self.Entrada22.get()).encode())
self.Entrada22.delete(0, END) # Limpa os valores dos campos de entrada
```


```
t = time.localtime()
auxtemp = t[4] + temp  #fica em um loop pelo tempo de giro do pivo
if auxtemp > 59:
  auxtemp = 60 - t[4]
  auxtemp = temp - auxtemp
while t[4] != auxtemp:
  x = 1
  t = time.localtime()
```

```
ser.write((self.Entrada23.get()).encode())
self.Entrada23.delete(0, END) # Limpa os valores dos campos de entrada



t = time.localtime()
auxtemp = t[4] + temp  #fica em um loop pelo tempo de giro do pivo
if auxtemp > 59:
  auxtemp = 60 - t[4]
  auxtemp = temp - auxtemp
while t[4] != auxtemp:
  x = 1
  t = time.localtime()
ser.write((self.Entrada24.get()).encode())
self.Entrada24.delete(0, END) # Limpa os valores dos campos de entrada



t = time.localtime()
auxtemp = t[4] + temp  #fica em um loop pelo tempo de giro do pivo
if auxtemp > 59:
  auxtemp = 60 - t[4]
  auxtemp = temp - auxtemp
while t[4] != auxtemp:
```

```
  x = 1
  t = time.localtime()
ser.write((self.Entrada25.get()).encode())
self.Entrada25.delete(0, END) # Limpa os valores dos campos de entrada

t = time.localtime()
auxtemp = t[4] + temp  #fica em um loop pelo tempo de giro do pivo
if auxtemp > 59:
  auxtemp = 60 - t[4]
  auxtemp = temp - auxtemp
while t[4] != auxtemp:
  x = 1
  t = time.localtime()
ser.write((self.Entrada26.get()).encode())
self.Entrada26.delete(0, END) # Limpa os valores dos campos de entrada

t = time.localtime()
auxtemp = t[4] + temp  #fica em um loop pelo tempo de giro do pivo
if auxtemp > 59:
  auxtemp = 60 - t[4]
  auxtemp = temp - auxtemp
while t[4] != auxtemp:
```

```
  x = 1
  t = time.localtime()
ser.write((self.Entrada27.get()).encode())
self.Entrada27.delete(0, END) # Limpa os valores dos campos de entrada


t = time.localtime()
auxtemp = t[4] + temp  #fica em um loop pelo tempo de giro do pivo
if auxtemp > 59:
  auxtemp = 60 - t[4]
  auxtemp = temp - auxtemp
while t[4] != auxtemp:
  x = 1
  t = time.localtime()
ser.write((self.Entrada28.get()).encode())
self.Entrada28.delete(0, END) # Limpa os valores dos campos de entrada



t = time.localtime()
auxtemp = t[4] + temp  #fica em um loop pelo tempo de giro do pivo
if auxtemp > 59:
  auxtemp = 60 - t[4]
  auxtemp = temp - auxtemp
```

```
while t[4] != auxtemp:
  x = 1
  t = time.localtime()
ser.write((self.Entrada29.get()).encode())
self.Entrada29.delete(0, END) # Limpa os valores dos campos de entrada



t = time.localtime()
auxtemp = t[4] + temp  #fica em um loop pelo tempo de giro do pivo
if auxtemp > 59:
  auxtemp = 60 - t[4]
  auxtemp = temp - auxtemp
while t[4] != auxtemp:
  x = 1
  t = time.localtime()
ser.write((self.Entrada30.get()).encode())
self.Entrada30.delete(0, END) # Limpa os valores dos campos de entrada



t = time.localtime()
auxtemp = t[4] + temp  #fica em um loop pelo tempo de giro do pivo
```

```
    if auxtemp > 59:
      auxtemp = 60 - t[4]
      auxtemp = temp - auxtemp
    while t[4] != auxtemp:
      x = 1
      t = time.localtime()
    ser.write((self.Entrada31.get()).encode())
    self.Entrada31.delete(0, END) # Limpa os valores dos campos de entrada

    t = time.localtime()
    auxtemp = t[4] + temp  #fica em um loop pelo tempo de giro do pivo
    if auxtemp > 59:
      auxtemp = 60 - t[4]
      auxtemp = temp - auxtemp
    while t[4] != auxtemp:
      x = 1
      t = time.localtime()
    ser.write((self.Entrada32.get()).encode())
    self.Entrada32.delete(0, END) # Limpa os valores dos campos de entrada

    t = time.localtime()
    auxtemp = t[4] + temp  #fica em um loop pelo tempo de giro do pivo
```

```
if auxtemp > 59:
  auxtemp = 60 - t[4]
  auxtemp = temp - auxtemp
while t[4] != auxtemp:
  x = 1
  t = time.localtime()
ser.write((self.Entrada33.get()).encode())
self.Entrada33.delete(0, END) # Limpa os valores dos campos de entrada

t = time.localtime()
auxtemp = t[4] + temp  #fica em um loop pelo tempo de giro do pivo
if auxtemp > 59:
  auxtemp = 60 - t[4]
  auxtemp = temp - auxtemp
while t[4] != auxtemp:
  x = 1
  t = time.localtime()
ser.write((self.Entrada34.get()).encode())
self.Entrada34.delete(0, END) # Limpa os valores dos campos de entrada

t = time.localtime()
auxtemp = t[4] + temp  #fica em um loop pelo tempo de giro do pivo
```

```
    if auxtemp > 59:
      auxtemp = 60 - t[4]
      auxtemp = temp - auxtemp
    while t[4] != auxtemp:
      x = 1
      t = time.localtime()
    ser.write((self.Entrada35.get()).encode())
    self.Entrada35.delete(0, END) # Limpa os valores dos campos de entrada



    t = time.localtime()
    auxtemp = t[4] + temp  #fica em um loop pelo tempo de giro do pivo
    if auxtemp > 59:
      auxtemp = 60 - t[4]
      auxtemp = temp - auxtemp
    while t[4] != auxtemp:
      x = 1
      t = time.localtime()
    ser.write((self.Entrada36.get()).encode())
    self.Entrada36.delete(0, END) # Limpa os valores dos campos de entrada
```

```
    tkinter.messagebox.showwarning("Upload", "O upload foi realizado com
sucesso!")


 def Read(self):


   self.Entrada1.delete(0, END) # Limpa os valores dos campos de entrada

   self.Entrada2.delete(0, END)


   ser.write(("r").encode())

   time.sleep (3)


   var = ser.readline().decode()

   valor = var[1:].rstrip() # Extrai os valores a partir da 2a posição e tira o espaço
do final

   self.Entrada1.insert(END, valor)


   var = ser.readline().decode()

   valor = var[1:].rstrip()

   self.Entrada2.insert(END, valor)


   tkinter.messagebox.showwarning("Read", "O arquivo foi lido com sucesso!")
```

```
if __name__ == "__main__":
                                    GUIFramework().mainloo
```

Firmware de recebimento dos dados advindo do Raspberry pi e controle do servo motor

```
#include <Servo.h>

#define SERVO 6 // Porta Digital 6 PWM

Servo s; // Variável Servo
int pos; // Posição Servo
float input; //usuário digita a regulagem
int control;
int auxpos = 180;
int maxgiro = 180;
float ang;
int auxinicio = 0;


void setup ()
{
```

```
 s.attach(SERVO);
 Serial.begin(115200);
 //Serial.println("aqui");
 s.write(180); // Inicia motor posição zero
}

void loop()
{
 if (Serial.available()>0){
   input=Serial.parseFloat();
  ang = (input-58.469)/-0.2437; //Calcula o angulo a partir da equação ang = freq-58.469/-0.2437 obtida pelo excel
  Serial.println(ang);

  for (int i=0; i<=180; i++){
    float aux1 = i - ang;



    if (aux1 > 0) {
     if (aux1 <1){ //Coloca o angulo com valor inteiro, por exemplo ang = 12,8 fica 12 graus
      maxgiro = aux1 + (ang - 1);
      i = 181;
```

```
    }
   }

 }


Serial.println(maxgiro);


if (auxpos < maxgiro){
for(pos = auxpos; pos <= maxgiro; pos = pos+1){
s.write(pos);
Serial.println(pos);
delay(100);
}
}


if (auxpos > maxgiro){
for(pos = auxpos; pos >= maxgiro; pos = pos-1){
Serial.println(pos);
```

```
s.write(pos);
delay(100);
}

}

 }
auxpos = maxgiro;
}
```