MEC-SETEC
INSTITUTO FEDERAL DE MINAS GERAIS – Campus Formiga
Curso de Engenharia Elétrica

DESENVOLVIMENTO DE UM MÉTODO DE CONTROLE APLICADO A UM SISTEMA DE ENERGIA SOLAR VIA PADRÃO SERIAL RS-485

MARCO ANTONIO LAINI RODRIGUES

Orientador: Professor Me. Gustavo Lobato Campos

FORMIGA – MG
2015

MARCO ANTONIO LAINI RODRIGUES

DESENVOLVIMENTO DE UM MÉTODO DE CONTROLE APLICADO A UM SISTEMA DE ENERGIA SOLAR VIA PADRÃO SERIAL RS-485

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia Elétrica do Instituto Federal de Minas Gerais como requisito para obtenção do título de bacharel em Engenharia Elétrica.

Orientador: Prof. Me. Gustavo Lobato Campos

FORMIGA – MG

2015

Marco Antonio Laini Rodrigues

DESENVOLVIMENTO DE UM MÉTODO DE CONTROLE APLICADO A UM SISTEMA DE ENERGIA SOLAR VIA PADRÃO SERIAL RS-485

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia Elétrica do Instituto Federal de Minas Gerais como requisito para obtenção do título de bacharel em Engenharia Elétrica.

BANCA EXAMINADORA

_____

Me. Gustavo Lobato Campos
Orientador(a)

_____

Me. Carlos Renato Borges dos Santos
Avaliador(a)

_____

Me. Rafael Vinícius Tayette da Nobrega
Avaliador(a)

Formiga, dia 11 de Junho de 2015.

Dedico este trabalho em especial aos meus pais, que sempre me deram força para conquistar mais essa vitória em minha vida.

AGRADECIMENTOS

Agradeço primeiramente a Deus, serei grato sempre pelas bençõe recebidas. Em seguida agradeço aos meus pais por dedicarem suas vidas a mim e a meu irmão durante todos estes anos. Sei que o caminho de vocês não foi fácil para me proporcionar esta conquista e serei eternamente grato por isso.

Agradeço também a meu orientador Prof. Gustavo pela paciência e prontidão que sempre teve para me atender. Agradeço a todos os meus professores, colegas, técnicos e funcionários do Instituto Federal que foram fundamentais para minha formação, não só acadêmica, mas também pessoal.

RESUMO

A necessidade de investir em novas tecnologias para viabilizar o uso de fontes alternativas de energia, como a solar, fica evidente ao passo que a matriz energética brasileira (as hidrelétricas) sofrem consequências graves pela falta de chuvas que afetam os níveis dos reservatórios desde 2012. É pensando nisso que foi construída no Instituto Federal de Minas Gerais, através de um projeto de pesquisa aplicada, uma planta solar controlada por sensores, atuadores e um microcontrolador a fim de aumentar a eficiência dos aquecedores solares e também para reaproveitamento de água. Porém, esta planta solar possui alguns problemas e/ou pontos de melhoras que este trabalho tem como objetivo identificar e conjuntamente sugerir um novo método de controle que seja mais eficiente e que aumente a confiabilidade da mesma. A sugestão deste trabalho se baseia no modelo de controle mestre – escravos e através de microcontroladores Arduinos é feita a coleta de dados dos sensores pelos escravos, e enviado ao mestre através de um barramento de comunicação serial via padrão RS-485.

Palavras chave: Arduino, Comunicação Serial, Mestre-Escravo, Novas Tecnologias, Planta Solar, RS-485.

ABSTRACT

The necessity to invest in new technologies to enable the use of alternative energy sources, as the solar, is evidenced while the Brazilian energy matrix (the hydro powers) suffer serious consequences by the lack of rains that affect the reservoirs levels since 2012. And thinking about it, was built at the Institute Federal de Minas Gerais campus Formiga – (IFMG), through an applied research project, a solar plant controlled by sensors, actuators and one microcontroller in order to Increase the efficiency of solar heaters and for water reuse too. However, this solar plant has some problems and / or improvements points that this work aims to identify and jointly suggest a new control method which is more efficient and increases to the trustworthiness of it. The suggestion of this paper is based on the model Master - Slaves Control. Through Arduinos microcontrollers is made the data collect from the sensors by slaves, and sent to the master through a serial bus communication via RS-485 standard.

Keywords: Arduino, Master–Slave, New technologies, RS-485Serial Communication, Solar Plant, Serial Communication.

LISTA DE FIGURAS

FIGURA 1 COMPARAÇÃO ENTRE OS VOLUMES MENSAIS DOS RESERVATÓRIOS DE FURNAS NOS ANOS DE 2012 E 2014.....12
FIGURA 2 ESTRUTURA DE UM SDCD EM BARRAMENTO DUPLO. 19
FIGURA 3 SISTEMA DE CONTROLE CENTRALIZADO MESTRE-ESCRAVO. 21
FIGURA 4 TRANSMISSÃO PARALELA. 24
FIGURA 5 MODO SÍNCRONO DE COMUNICAÇÃO. 25
FIGURA 6 MODELO ASSÍNCRONO DE COMUNICAÇÃO. 25
FIGURA 7 COMUNICAÇÃO RS-232 27
FIGURA 8 SISTEMA RS-485 COM COMUNICAÇÃO HALF-DUPLEX.....29
FIGURA 9 VELOCIDADE DA COMUNICAÇÃO X COMPRIMENTO DO CABO.....30
FIGURA 10 ARDUINO UNO. 32
FIGURA 11 PORTAS ARDUINO UNO.....33
FIGURA 12 IDE ARDUINO. 34
FIGURA 13 EXEMPLO DE UMA MÁQUINA DE ESTADOS. 36
FIGURA 14 RESERVATÓRIO, BOMBA D'ÁGUA E VÁLVULA SOLENOIDES. 37
FIGURA 15 PLANTA SOLAR DO IFMG.....38
FIGURA 16 ARQUITETURA DA PLANTA SOLAR. 39
FIGURA 17 RUÍDOS OCASIONADOS PELA DISTÂNCIA ENTRE O LM35 E O CONTROLADOR. 42
FIGURA 18 SISTEMA CENTRALIZADO DE CONTROLE DA PLANTA SOLAR.....43
FIGURA 19 SENSOR DE NÍVEL MPX 5010.....44
FIGURA 20 PERFIL DO SENSOR DE NÍVEL MPX 5010.....45
FIGURA 21 CI MAX485 E BARRAMENTO E SINAIS DIFERENCIAIS.....48
FIGURA 22 PAR DE FIOS TRANÇADOS LIGANDO DOIS COMPONENTES.....49
FIGURA 23 LIGAÇÃO ENTRE OS MESTRES E ESCRAVOS PELO PADRÃO SERIAL RS-485. 51
FIGURA 24 MÁQUINA DE ESTADOS ARDUINO MESTRE.....53
FIGURA 25 MÁQUINA DE ESTADOS ARDUINOS ESCRAVOS. 54
FIGURA 26 PACOTE DE DADOS ENVIADO PELO CONTROLADOR MESTRES..55
FIGURA 27 PACOTE DE DADOS ENVIADO PELOS ESCRAVOS AO MESTRE....55

FIGURA 28 SHIELD DESENVOLVIDO PARA PADRÃO SERIAL RS-485 PARA ARDUINO UNO. 57
FIGURA 29 CONTROLADOR MESTRE (CENTRO) E ESCRAVOS CONECTADOS. 57
FIGURA 30 AFERIÇÃO DO SENSOR LM 35 PARA UM CABO DE 2 METROS DE COMPRIMENTO. 59
FIGURA 31 SHIELDS DO ESCRAVO 1 INSTALADOS NA PLANTA SOLAR DO IFMG. 59
FIGURA 32 ESCRAVO 2 INSTALADO. 60
FIGURA 33 ARDUINO ESCRAVO AGUARDANDO O ENDEREÇO. 61
FIGURA 34 COMPARAÇÃO ENTRE O SISTEMA ANTERIOR E O PROPOSTO. 62
FIGURA 35 CONTROLADOR MESTRE. 63
FIGURA 36 LEITURAS REQUISITADAS AO ESCRAVO 1. 64
FIGURA 37 LEITURAS DE TODOS OS DADOS E CONSIDERAÇÕES. 65
FIGURA 38 ACIONAMENTO DA BOMBA 1, APÓS ESVAZIAR O RESERVATÓRIO 1. 66

SUMÁRIO

1 INTRODUÇÃO 12

1.1 PROBLEMA 13

1.2 HIPÓTESE 14

1.3 JUSTIFICATIVA 15

2 OBJETIVOS 16

2.1 OBJETIVO GERAL 16

2.2 OBJETIVOS ESPECÍFICOS 16

3 REFERENCIAL TEÓRICO 18

3.1 REDES INDUSTRIAIS 18
3.1.1 ARQUITETURAS DE REDES 18
3.1.2 MESTRE E ESCRAVO 22
3.1.3 COMUNICAÇÃO SERIAL E COMUNICAÇÃO PARALELA 22
3.1.4 PRINCIPAIS PADRÕES DE INTERFACE SERIAL 26

3.2 MICROCONTROLADORES 31
3.2.1 ARDUINO UNO 31

3.3 MÁQUINA DE ESTADOS FINITOS 35

4 MATERIAIS E MÉTODOS 37

4.1 SISTEMA SOLAR IFMG-FORMIGA 37
4.1.1 ARQUITETURA DA PLANTA SOLAR 38
4.1.2 PROBLEMAS ENCONTRADOS 42

4.2 SENSORES 43
4.2.1 MPX 5010 44
4.2.2 SENSOR DE TEMPERATURA LM 35 45

4.3 METODOLOGIA DO TRABALHO 46
4.3.1 PADRÃO DE COMUNICAÇÃO SERIAL RS-485 46
4.3.2 BARRAMENTO DE COMUNICAÇÃO RS-485 48
4.3.3 PROGRAMAÇÃO 52

5 RESULTADOS E DISCUSSÃO 56

5.1 SHIELD RS-485 PARA ARDUINO UNO 56

5.2 ESCRAVOS 58
5.2.1 COMPORTAMENTO DE CONTROLE DOS ESCRAVOS 61

5.3 MESTRE 62
5.3.1 COMPORTAMENTO DE CONTROLE DO MESTRE 63

6 CONCLUSÕES.....68

7 TRABALHOS FUTUROS 69

8 REFERÊNCIAS BIBLIOGRÁFICAS.....70

Anexo A.....73

Anexo B.....82

Anexo C.....87

1 INTRODUÇÃO

Ao analisar a situação dos reservatórios de água para produção de energia elétrica nos últimos anos, nota-se uma acentuada queda dos seus níveis a partir do ano de 2012 (ONS, 2015). Nos primeiros meses deste ano, os volumes encontravam-se em seus estados ótimos e ao final, já se encontraram em situação crítica. As chuvas decorrentes dos anos que o sucederam até o presente não foram suficientes para reestabelecer a normalidade. Um exemplo, pode ser encontrado no comparativo feito pela ONS (Operador Nacional do Sistema Elétrico) do volume útil dos principais reservatórios de Furnas, nos anos de 2012 e 2014 como mostra a figura 1.

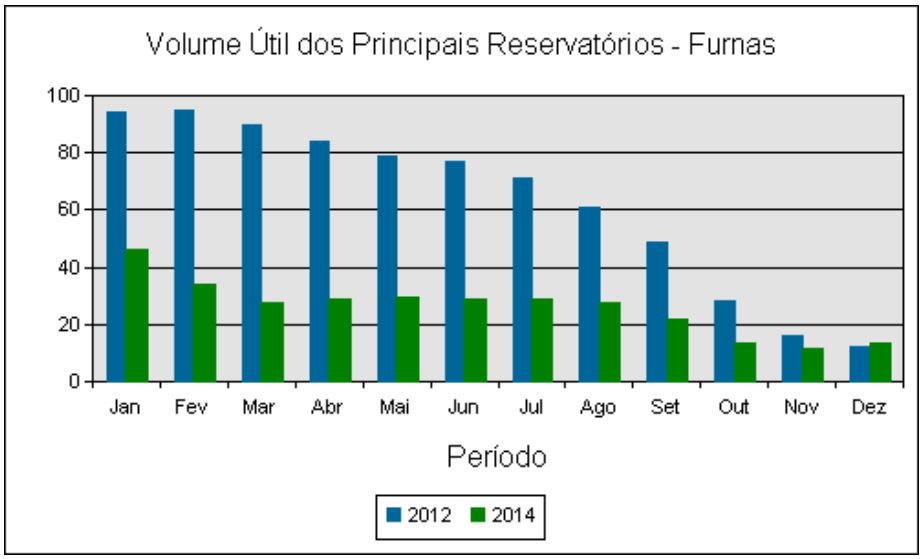

Figura 1 Comparação entre os volumes mensais dos reservatórios de furnas nos anos de 2012 e 2014.

Fonte: (ONS, 2014)

Em momentos como esse, a necessidade de se investir em energias alternativas, renováveis e até mesmo em novas tecnologias para aperfeiçoar as já existentes se fazem mais do que necessário.

Partindo deste ponto de vista, foi projetada e construída uma planta solar, pelo programa institucional de fomento à pesquisa aplicada do Instituto Federal de Minas Gerais (IFMG) – Campus Formiga, intitulado: “Estudo sobre a energia solar e suas aplicações à inclusão da população de baixa renda e ao programa ‘Luz Para Todos’”.

Este trabalho de conclusão de curso se baseia no estudo deste projeto de pesquisa, identificando seus problemas e propondo soluções pertinentes para cada um deles, a fim de melhorar sua eficiência e aumentar a confiabilidade do controle feito na planta.

1.1 PROBLEMA

A planta solar instalada no IFMG-Formiga, tem o objetivo de aumentar a eficiência dos aquecedores solares de baixa renda e comerciais e evitar o desperdício de água pelos mesmos. Para isso, são utilizados sensores de temperatura e nível, além de válvulas solenoides, bombas d’água, um reservatório para a água que seria desperdiçada e um controlador responsável por coordenar todos os instrumentos citados. Os detalhes desta planta serão Informados com maior riqueza ainda neste trabalho.

Estudando a planta solar e seu método de controle, foi possível detectar alguns problemas que podem comprometer o funcionamento da mesma, estes são:

- Um único controlador responsável por todo o processo da planta;
- Longas distâncias dos pontos de medição para o controlador, ocasionando ruídos nas medições.
- Excesso de cabeamento;
- Limitação nos números de sensores e atuadores;
- Confiabilidade.

Todos estes problemas identificados serão exemplificados e explicados neste texto.

1.2 HIPÓTESE

Ao identificar os problemas da planta solar, uma solução pode ser apresentada ao aplicar um modelo que permita o sistema diminuir os cabeamento e seus efeitos e, que ao mesmo tempo aumente a confiabilidade do sistema. A solução proposta neste trabalho se baseia no modelo de controle Mestre-Escravo.

O modelo mestre-escravo é caracterizado por possuir um controlador responsável pela tomada de decisões (mestre) e outro(s) responsável(eis) por executar as funções requeridas pelo primeiro controlador (escravos).

Os controladores mestres e escravos devem ser interligados por uma rede de comunicação serial. Por critérios de razões técnicas, que serão justificadas ainda neste trabalho, foi escolhido o padrão de comunicação serial RS-485.

Segundo (KRON, 2013), este padrão é destinado a comunicações de longas distâncias, cerca de 1200 metros, utilizando apenas um par de fios trançados e pode ter até 32 dispositivos conectados à rede. Este trabalho sugere o uso de um controlador mestre e outros dois controladores escravos, instalados em pontos estratégicos da planta para o seu controle.

Para que este método proposto seja possível, foi desenvolvido um protótipo que atenda a todas as necessidades da planta e ao mesmo tempo resolva os problemas encontrados na planta fruto do projeto de pesquisa aplicada.

1.3 JUSTIFICATIVA

Quando se utiliza apenas um controlador para todo o sistema, em caso de falha, todo o processo pode ser comprometido o que acarretaria em perda da eficiência. Ao distribui as responsabilidades, mesmo que um controlador escravo sofra algum dano, os outros continuariam a trabalhar.

O uso da metodologia mestre-escravo permite que a tomada de decisões continue centralizada, mas permite que os escravos estarão mais próximos dos sensores e atuadores, eliminando assim as perdas de informações e ruídos ocasionadas pela resistência dos longos cabos antes utilizados.

A distância para comunicação é importante para planejar os pontos em que os controladores estarão alocados. O padrão RS-485 permite que os escravos estejam distantes a 1200 metros do mestre (ALBUQUERQUE, 2009) sem perda de sinais.

2 OBJETIVOS

Este capítulo irá tratar dos objetivos deste trabalho de conclusão de curso, a fim de traçar metas e especificar o propósito do trabalho.

2.1 OBJETIVO GERAL

É importante ressaltar que o objetivo deste trabalho não é modificar a rotina de controle já existente da planta solar do IFMG-Formiga, e sim identificar os possíveis problemas provenientes do modelo de controle implantado e desenvolver um novo meio de comunicação que seja melhor do que o sistema linear instalado.

A solução proposta é utilizar um modelo baseado no sistema mestre-escravo. Para isso, deve-se encontrar um padrão de comunicação que utilize este modelo e que seja eficiente para suprir as necessidades de controle da planta e que solucione os problemas que foram identificados.

2.2 OBJETIVOS ESPECÍFICOS

Como objetivo específico do trabalho, foi realizado um levantamento bibliográfico para identificar ações de melhorias no método de controle da planta solar, estas são:

- Estudar a planta solar do IFMG – Formiga e identificar os possíveis problemas do método de controle da planta;
- Adotar e estudar o modelo de comunicação mestre-escravo;
- Adotar um protocolo de comunicação que seja mais eficiente do que o atualmente empregado;
- Estudar e sugerir o uso de um controlador específico;
- Adotar e estudar o algoritmo com ação em máquinas de estado;

Visando a melhoria do método de controle da planta solar, foi abordado um breve referencial teórico sobre redes Industriais, mais especificamente no modelo mestre-escravo, os padrões em que podem realizar a comunicação entre eles e a ação em máquinas de estados finitos para o novo algoritmo de controle. Um breve resumo sobre microcontroladores também foi abordado no capítulo seguinte, visando a troca do Arduino *mega*, pelo Arduino.

3 REFERENCIAL TEÓRICO

Este capítulo é destinado a um breve estudo sobre os itens que foram necessários para o desenvolvimento deste trabalho. Ele abordará os assuntos: Redes industriais, para uma nova metodologia de controle da planta solar; Microcontroladores, visando a troca do microcontrolador atual por um de menor custo; Metodologia de programação sequencial “Máquina de estados Finitos” recomendada para sistemas onde o sincronismo entre os componentes é extremamente exigido.

3.1 REDES INDUSTRIAIS

A comunicação é uma das maiores necessidades do ser humano desde os primórdios. Nos dias de hoje, a globalização nos conduz a uma nova forma de organização social, nos quais os impactos podem ser comparados aos da Revolução Industrial. É partindo deste contexto que novas tecnologias são exigidas dia-a-dia nos ambientes industriais, e por que não, na casa de qualquer indivíduo. (ALBUQUERQUE, 2009; TANENBAUM, 2003).

3.1.1 ARQUITETURAS DE REDES

No início, os sistemas de controle eram totalmente baseados em controladores com malha única de realimentação (*Single – Loop Controllers – SLC)*. Após a década de 1960, os sistemas baseados em minis e grandes computadores digitais, os DDCs (*DIrect Digital Controller*), eram os mais utilizados, porém estão quase extintos atualmente (ALBUQUERQUE, 2009).

Durante os anos 1990, ocorreram grandes desenvolvimentos da eficiência dos computadores, CLPs, sensores, atuadores e sistemas de comunicação, tornando os Sistemas Digitais de Controle Distribuído (SDCD) uma realidade não só nos

processos industriais, mas de praticamente todos os segmentos da automação (ALBUQUERQUE, 2009).

Os atuais SDCDs implementam as malhas de controle em pequenos grupos, cada grupo com seu próprio processamento (controlador). Os controladores são conectados via barra de comunicação de dados (*Data Highway – BUS*), que em alguns casos podem ser duplicados para aumentar a confiabilidade conforme a figura 2 (ALBUQUERQUE, 2009).

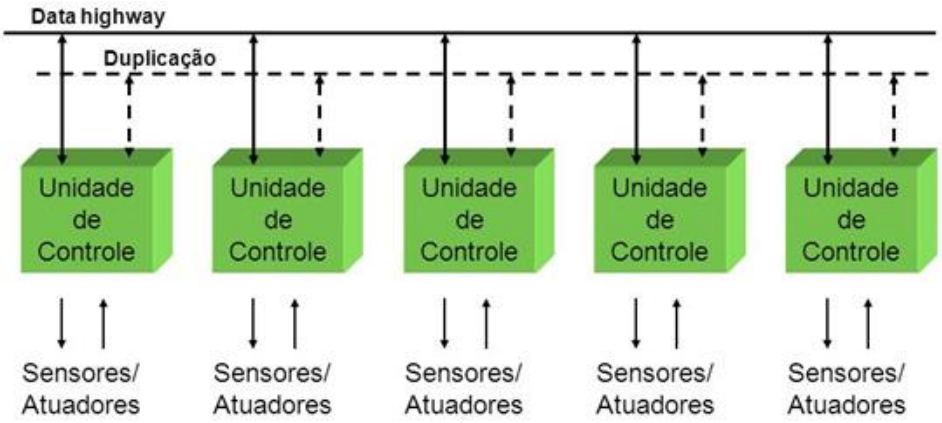

Figura 2 Estrutura de um SDCD em barramento Duplo.
Fonte: (ALBUQUERQUE, 2009) Modificado

Com o uso de um SDCD pode-se ter o controle das funções tão independente quanto se deseja. Pode-se ainda, monitorar e ajustar as funções de controle de forma centralizada e com fácil operação. Um dos sistemas que mais tem se desenvolvido nos últimos anos é o de multicomputadores. Estes sistemas são definidos como subsistemas com mais de um computador com interação *online.* Uma subcategoria que pode ser definida é o sistema de multiprocessadores que inclui, em partes, as subclasses dos sistemas a multiprocessador e sistema distribuídos (ALBUQUERQUE, 2009).

Como o nome já sugere, os sistemas multiprocessadores possuem diversos processadores, controlados por um único gerenciamento, com a capacidade de executar uma mesma tarefa compartilhada dinamicamente. No entanto cada processador possui seu próprio gerenciamento (programa aplicativo) e realiza funções especiais (ALBUQUERQUE, 2009).

Neste ponto do trabalho, são destacados dois modelos de arquitetura: o sistema de controle centralizado e o sistema de controle distribuído que serão explicados respectivamente a seguir.

4.1.1.1 Sistema de controle centralizado

O sistema centralizado é feito de forma a manter vários dispositivos na mesma sala. Desta forma, vários computadores compartilham um barramento comum. Os sistemas com barramentos paralelos são os com maior aplicação em processos que precisam de uma alta eficiência de processamento. Um sistema de multicomputadores é o adequado para preencher esta solicitação de eficiência. (ALBUQUERQUE, 2009).

Um sistema a multicomputadores usado como uma máquina de controle, consiste em um controle principal (*master)* e de controladores escravos (*slaves),* de acordo com a figura 3.

Para este tipo de atuação, a primeira etapa é separar as tarefas que requerem relativamente pouca intercomunicação. Na arquitetura mestre-escravo, o controlador mestre é responsável pela tomada do controle global e tomada de decisões, enquanto o escravo atua no nível do atuador (ALBUQUERQUE, 2009).

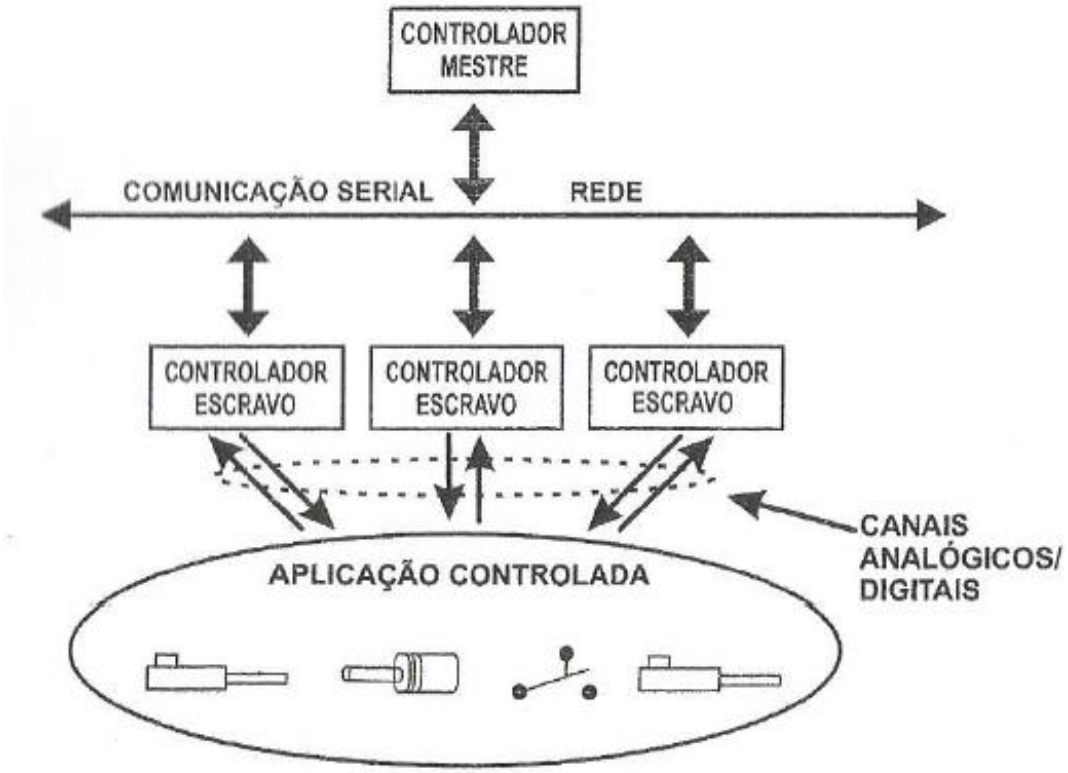

Figura 3 Sistema de controle centralizado Mestre-Escravo.

Fonte: (ALBUQUERQUE, 2009).

4.1.1.2 Sistema de Controle Distribuído

Este sistema é caracterizado pelos transdutores, atuadores e controladores espacialmente distribuídos nas plantas. A ideia principal é usar uma rede de comunicação serial para conectar as partes e minimizar a complexidade do cabeamento, principalmente em grandes instalações. Este barramento é conhecido como *fieldbus (*barramento de campo) (ALBUQUERQUE, 2009).

Um exemplo, pode ser observado na figura 3, onde o controlador coleta informações dos vários transdutores e controla um atuador ou grupo de atuadores. Em relação à distribuição das tarefas de controle, este sistema é completamente

centralizado. Somente as atuações de baixo nível e as informações dos sensores são distribuídas (ALBUQUERQUE, 2009).

A seguir será dada a definição do que significam os controladores mestres e escravos no sistema de controle distribuído.

3.1.2 MESTRE E ESCRAVO

Mestres e escravos possuem funções distintas dentro de uma rede de comunicação. O mestre tem como função principal controlar a rede de comunicação e concentrar todos os dados do sistema. Ele funciona também, como uma interface de operação remota e interface com sistemas de controle supervisórios, além de garantir comunicação com outros níveis da rede; realiza a interpretação dos comandos da rede e é responsável pelo sincronismo do sistema, bem como a coordenação da rede e cálculos matemáticos complexos (ALBUQUERQUE, 2009).

Os escravos por sua vez, possuem a função de receber a informação do mestre e executá-la da melhor maneira possível, atuando em tarefas localizadas. Além disso, pode realizar processamento de sinais, efetuar leituras e manipular eventos de forma pré-determinadas (ALBUQUERQUE, 2009).

Adotando o sistema de controle distribuído como modelo e utilizando o padrão mestre escravo como sugestão deste trabalho, deve-se definir uma rede de comunicação entre os controladores. A seguir será feito uma comparação entre as comunicações seriais e a paralela para determinar qual a melhor para ser utilizada.

3.1.3 COMUNICAÇÃO SERIAL E COMUNICAÇÃO PARALELA

Quando há a necessidade de se comunicar entre dois sistemas digitais, pode-se fazer uso da comunicação paralela. Na comunicação paralela são enviados vários

bits de uma vez ao longo de um meio de transmissão composto de vários canais, um para cada *bit*. Levando em consideração a transmissão de um caractere (7, 8 ou 9 bits) por exemplo, fica claro que o custo se de alocar um canal para cada *bit* torna-se extremamente alto à medida que a distância cresce (ALBUQUERQUE, 2009).

Considerando o custo pela implementação de longos cabos de várias vias por vários metros, foi criado uma solução mais eficaz: o sistema de comunicação serial (ALBUQUERQUE, 2009).

A comunicação serial, na verdade, é uma variação da comunicação em paralelo. Ao invés de se enviar vários *bits* de uma só vez, cada um em seu canal, a comunicação serial envia os mesmos *bits,* no entanto, um de cada vez pelo mesmo canal (ALBUQUERQUE, 2009).

A transmissão paralela, em comparação com a serial, pode ser definida como:

- Transmissão de dados mais custosa e complexa;
- Necessita de mais de um canal de comunicação;
- Apresenta maiores velocidades durante a transmissão de dados;
- Custo elevado;
- Baixa imunidade a ruídos;
- Utilizados em curtas distâncias.

Na transmissão em paralelo, os *bits* compondo uma informação são conduzidos ao longo de um conjunto de vias, como é mostrado na figura 4.

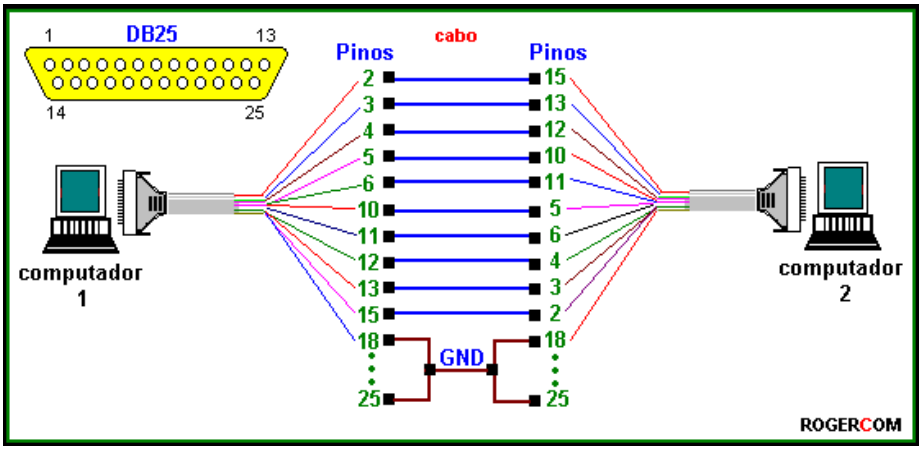

Figura 4 Transmissão paralela.

Fonte: Rogercom *

Na comunicação serial, o número de linhas à transmissão é bastante reduzido, convertendo os dados a serem transmitidos numa sequência serial de bits. A transmissão serial pode ser caracterizada como:

- Transmissão de dados menos complexa;
- Necessita apenas um canal de comunicação (par trançado, por exemplo);
- Apresenta menores velocidades à transmissão de dados;
- Menor custo;
- Maior imunidade a ruídos.

A comunicação serial pode ser definida em dois modos de comunicação que são:

A. Modo Síncrono: Necessita de um sincronismo entre os dois sistemas de comunicação. Na maioria dos casos, um dos sistemas conectados deve gerar um sinal de *clock* que deve ser seguido pelos demais sistemas. Neste modelo

* **Figura 4, Disponível em:** http://www.rogercom.com/pparalela/MicroMicro.htm

de comunicação, os sistemas devem transmitir e receber dados como verdadeiros registradores de deslocamento (*shift-registersl*), dispositivos em que a entrada é paralela e a saída serial. É importante ressaltar que o sincronismo entre os sinais de dados e o *clock* é um fator crítico para este modo. A figura 5 ilustra o modelo síncrono de comunicação.

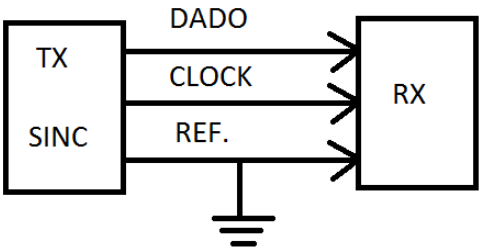

Figura 5 Modo Síncrono de comunicação.

Fonte: Arquivo pessoal.

B. Modo Assíncrono: No modo de comunicação assíncrono, não existe a necessidade do *clock* como no modo síncrono. O controle de tempo entre dois *bytes* consecutivos não é importante, mas o tempo de sequência de *bits* que compõe um *byte* é crítico. Os dois devem ter geradores de *clock* internos programados para a mesma taxa de transmissão de dados, denominada *baud* rate. A figura 6 mostra o modelo de comunicação assíncrona.

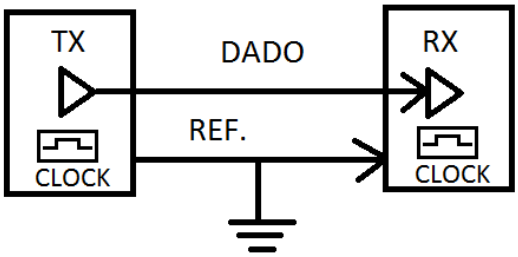

Figura 6 Modelo assíncrono de comunicação.

Fonte: Arquivo pessoal.

A seguir serão resumidos os principais padrões de interface serial avaliados.

3.1.4 PRINCIPAIS PADRÕES DE INTERFACE SERIAL

Os padrões para interface serial especificam as características elétricas, mecânicas e funcionais dos circuitos entre dois equipamentos. Devem determinar nomes, números e fios necessários para se estabelecer a comunicação. Estes padrões são estabelecidos pela TIA (Associação Internacional de Telecomunicações) e EIA (Associação Internacional de Eletrônica) (ALBUQUERQUE, 2009).

Os padrões mais conhecidos e utilizados são o RS-232, RS-422 e o RS-485 e serão resumidos, respectivamente, a partir deste ponto.

4.1.4.1 RS-232

Surgido em 1969, foi desenvolvido originalmente para especificar as conexões entre terminais e modens. Emprega a transmissão de sinais desbalanceados e os fios básicos para a transmissão são o TXd (*Transmitted data), o* Rxd (*Received data)* e o SG (*Signal Ground)* (ALBUQUERQUE, 2009).

Para evitar conflitos de dados, os equipamentos são divididos em dois tipos: DTE (*Data Terminal Equipament),* geralmente microcomputadores e DCE (*Data Communication Equipment)*, geralmente MODEM. A diferença entre os dois na prática se resume aos pino do conector da porta serial no qual se tem Txd e Rxd e nas linhas usadas para controle de fluxo (ALBUQUERQUE, 2009).

Como a comunicação é digital, deve-se definir o que significa um bit 0 e o que significa um bit 1. No padrão RS-232, o bit 0 é uma tensão positiva entre o intervalo de +5V e +15V, para saída e entre +3V e +15V para saída. Já para o bit 1 são os mesmo valores em módulo, porém negativos. Todos esses níveis de tensão são em

relação ao SG (Sinal *Ground*) (ALBUQUERQUE, 2009). A figura 7 ilustra como é feito a comunicação RS-232.

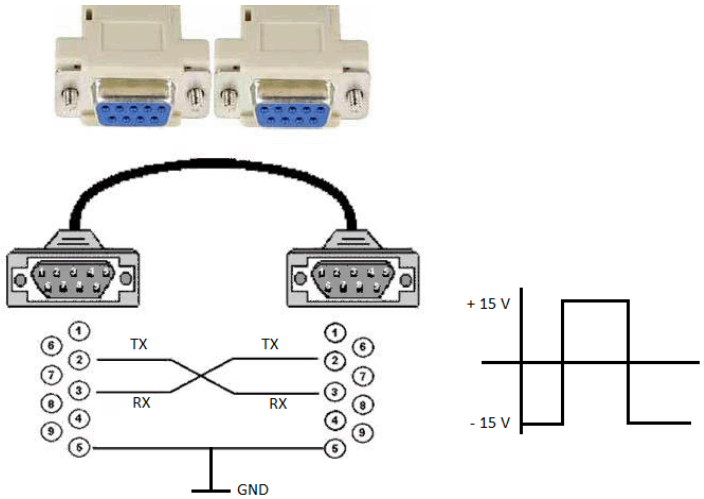

Figura 7 Comunicação RS-232

Fonte: (Baú da Eletrônica, 2011)* (modificada)

O alcance máximo é 15 metros, porém a distância efetiva está diretamente relacionada com a taxa de transmissão, o cabo utilizado e as condições de ruído do ambiente (ALBUQUERQUE, 2009).

4.1.4.2 RS-422

Este padrão é destinado para transmissões à longa distância (1200m – por norma) e que exigem altas velocidades; são necessários dois pares de fios para transmissão *duplex (full duplex)* e é capaz de interligar um dispositivo transmissor à até 10 receptores (ALBUQUERQUE, 2009; KRON, 2013).

Neste padrão, o nível 1 é representado fisicamente por uma tensão positiva do pino de sinal (+) em relação ao pino de referência(-), podendo variar de 2V a 12V para

* Figura 7, Disponível em: http://baudaeletronica.blogspot.com.br/2011/09/pinagem-da-rs232c.html, Acesso em Março,2015.

saída e de 0,2V a 12V para entrada. Já o nível zero deve ter uma tensão negativa do pino de sinal (+) em referência ao pino de referência (-) (ALBUQUERQUE, 2009).

4.1.4.3 RS-485

No protocolo RS-485 existe apenas um par de fios para transmissão e recepção que deve ser compartilhado. Esta estratégia possui algumas vantagens e desvantagens. A grande vantagem é a possibilidade de interligar vários equipamentos que podem se comunicar entre si através do mesmo cabo. Normalmente, um dispositivo transmissor/receptor corresponde a uma “unidade de carga”, o que faz com que seja possível comunicar com até 32 dispositivos. Entretanto, existe dispositivos que consomem frações de unidade de carga, o que aumenta o número máximo de dispositivos a serem interligados. O meio físico mais utilizado é um par trançado, onde através deste único par de fios, cada dispositivo transmite e recebem dados. O alcance da transmissão do RS-485 é compatível ao padrão RS-*422* (ALBUQUERQUE, 2009; KRON, 2013).

A grande desvantagem é que em determinado instante de tempo, somente um dispositivo pode transmitir o que caracteriza esta rede como *half-duplex.* Cada dispositivo aciona o seu transmissor apenas no instante que necessita transmitir, mantendo-o desligado no resto do tempo de modo a permitir que outros dispositivos transmitam dados. O *software* de comunicação deve gerenciar a habilitação de transmissão e recepção para evitar uma confusão de dados que pode ocorrer quando dois equipamentos tentam transmitir ao mesmo tempo. Neste caso, é importante permitir que cada equipamento só inicie uma transmissão quando verificar que a linha está livre (ALBUQUERQUE, 2009; KRON, 2013).

Outra solução para o problema de sincronismo, é ter um gerenciador de rede, que é o único a poder começar uma transmissão e é quem autoriza os demais componentes da rede a iniciar uma comunicação. Desse modo, evita-se todo conflito, porém há a desvantagem de que componentes da rede não podem comunicar entre si livremente, tendo que aguardar pelo gerenciador para desencadear o processo. Algoritmos mais elaborados devem contornar esse problema (ALBUQUERQUE, 2009; KRON, 2013).

É importante mencionar que a inserção de resistores nas duas extremidades do cabeamento da rede RS-485 permite um melhor casamento de impedâncias no cabo, evitando reflexões do sinal ao ponto de deteriorar a comunicação, conforme apresentado na figura 8 (ALBUQUERQUE, 2009).

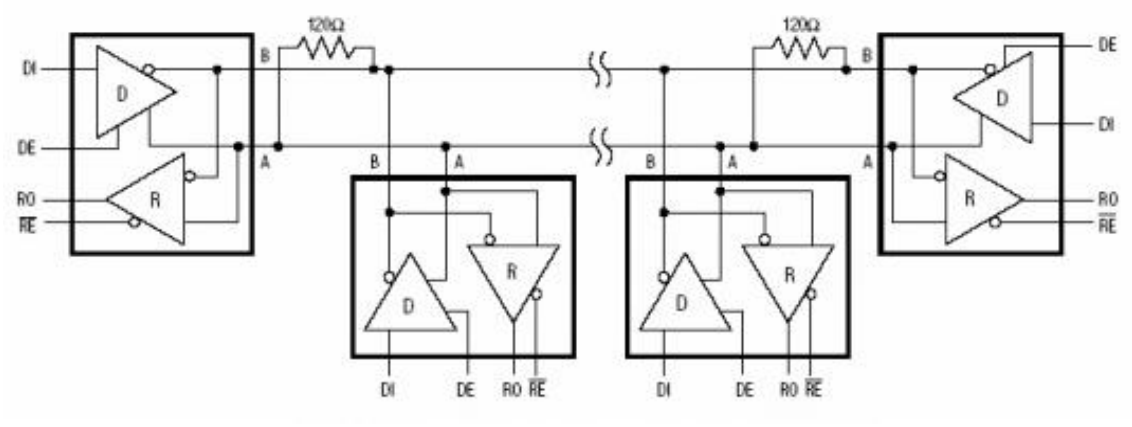

Figura 8 Sistema Rs-485 Com comunicação half-duplex

Fonte: **(Desconhecida)***

Os padrões RS-422 e RS-485 especificam um comprimento máximo de 1200 metros para os cabos de comunicação. A velocidade máxima de comunicação (em

* Figura 8: Fonte da imagem desconhecida; Disponível em: http://labdegaragem.com/forum/topics/scadabr?commentId=6223006%3AComment%3A135308

bits por segundo – bps) irá depender das características dos equipamentos instalados, da capacitância dos cabos de comunicação e dos resistores de terminação instalados.

Apesar de poder alcançar grandes distâncias, quanto maior for o comprimento do cabo, menor deve ser a velocidade de comunicação (KRON, 2013). A figura 9 ilustra a perda da velocidade da comunicação pelo comprimento do cabo.

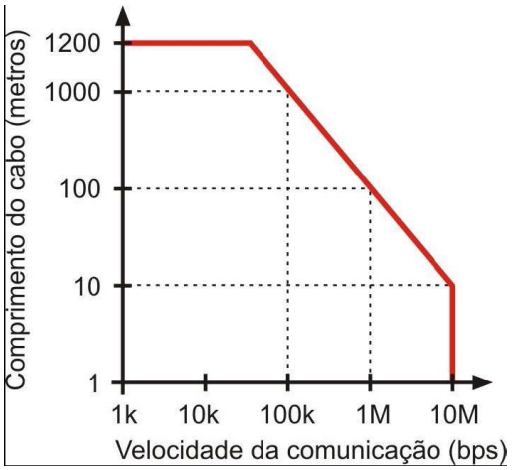

Figura 9 Velocidade da Comunicação X Comprimento do cabo

***Fonte:* (KRON, 2013)**

A tabela 2 apresenta uma comparação entre os padrões já citados:

Tabela 1 Comparação entre Padrões RS-232, RS-422 e RS-485.

Características	***RS-232***	***RS-422***	***RS-485***
Referência	Desbalanceada	Balanceada	Balanceada
Quantidade de Dispositivos	2	10	32
Distância Máx.	15 m	1200 m	1200 m
Taxa de Transmissão	20 kbps	10Mbps	10Mbps
Sensibilidade de Entrada	+/- 3V	+/- 0,2V	+/- 0,3V
Resistência de Entrada	3 a 7 kOhm	>4 kOhm	>12 kOhm

A seguir será feito um breve estudo sobre microcontroladores, com maior ênfase dada a plataforma Arduino, empregada neste trabalho.

3.2 MICROCONTROLADORES

Nos primórdios da tecnologia, todos os equipamentos eletrônicos eram constituídos por uma densa quantidade de componentes analógicos que faziam com que os dispositivos fossem maiores, mais pesados e de maior complexidade comparados aos que hoje são constituídos por microcontroladores (BRAGA, 2015; PEREIRA, 2003).

Existem diversas plataformas para microcontroladores. Em especial este trabalho dará ênfase a plataforma *Arduino* ®, mais especificamente ao *Arduino Uno.*

3.2.1 ARDUINO UNO

Segundo o próprio site do fabricante, o *Arduino* ® é uma plataforma de prototipagem eletrônica *open-source* que se baseia em *hardware* e *software* flexíveis e fácil de se usar (ARDUINO, 2015).

Deve se destacar as vantagens de se utilizar este equipamento que são:

- Baixo Custo: as placas de Arduinos são relativamente mais baratas em comparação com outras plataformas de microcontroladores;
- Multi-plataformas: o software Arduino pode ser executado em Windows, Macintosh OSX, Linux e outros sistemas operacionais. A maioria dos sistemas de microcontroladores são limitados ao Windows.
- Ambiente de programação simples;
- *Oppen Source:* o *software* Arduino é publicado como ferramentas de código aberto, disponível para extensão por programadores experientes.

Destaca-se neste trabalho o modelo *Arduino Uno.* Baseado no chip Atmega328, que dispõe de quatorze pinos digitais de entrada / saída, dos quais seis

podem ser usados como saída PWM (Pulse Width Modulation), seis entradas analógicas, um cristal de 16MHz *(Mega Hertz*), conexão USB, entrada para alimentação e botão de *reset*. Sua grande vantagem em relação aos outros microcontroladores, é que ele contém o necessário para apoiar o microcontrolador, basta conectá-lo a um computador com um cabo USB ou 32ili-lo com um adaptador AC-DC (*Alternate Corrent – Direct Corrent)* ou bateria para começar (ARDUINO, 2015; SOUZA, 2013). A figura 10 ilustra o Arduino Uno.

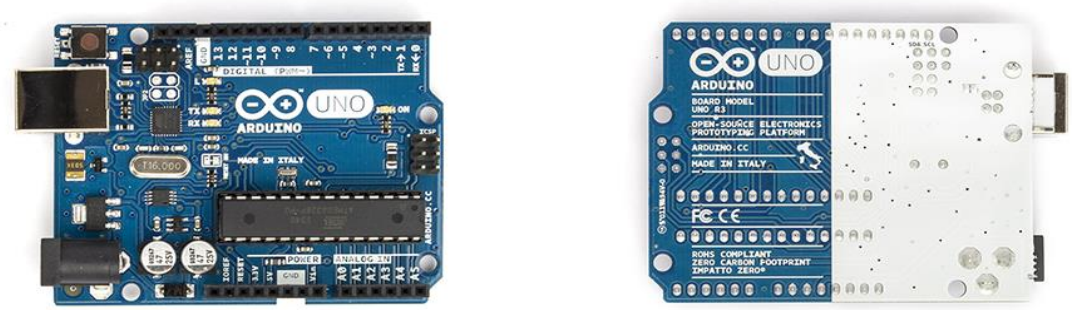

Figura 10 Arduino Uno.

Fonte: (ARDUINO, 2015)

Conhecido o *Arduino uno* deve-se identificar suas entradas e saídas, e também como 32iliampe-lo.

4.2.1.1 ENTRADAS E SAÍDAS

A placa Arduino Uno possui entradas e saídas analógicas e digitais. Ela possui quatorze pinos que podem ser utilizados como entradas ou saídas digitais (pinos 0 ao 13). Estes Pinos operam em 5 V (Volts), onde cada pino pode fornecer ou receber uma corrente máxima de 40 mA (32iliampere).

Cada um dos pinos digitais possuem resistores de *pull-up* interno que pode ser habilitado por software. Alguns desse pinos possuem funções especiais:

- PWM: os pinos 3,5,6,9,10 e 11 podem ser usados como saídas PWM de 8 bits;

- Comunicação serial: os pinos 0 e 1 podem ser utilizados para comunicação serial. Deve-se observar que estes pinos são ligados ao microcontrolador responsável pela comunicação USB com o PC;
- Interrupção externa**:** os pinos 2 e 3 podem ser configurados para gerar uma interrupção externa.

O *Arduino* uno possui ainda seis portas analógicas, onde cada uma possui uma resolução de 10 bits. Como padrão, a referência do conversor A/D (analógico /digital) está ligado internamente a 5V. O que significa que quando o estado analógico está em estado máximo, o valor digital convertido será 1023 (SOUZA, 2013). A figura 11 mostra a disposição das portas citadas presentes no Arduino.

Figura 11 Portas Arduino Uno.

Fonte: (SOUZA, 2013)

4.2.1.2 PROGRAMAÇÃO

O software para programação do *Arduino* é uma *Integrated Development Environment* (IDE), ou ambiente integrado de desenvolvimento (figura 12), que permite a criação de *sketches* para a placa *Arduino*. Ele apresenta um alto grau de abstração, possibilitando o uso de um microcontrolador sem que o usuário conheça o

mesmo, nem como deve ser usado os registradores internos de trabalho (SOUZA, 2013).

A IDE do Arduino possui uma linguagem de programação própria baseada na linguagem C e C++. A partir do momento que é feito o *upload* da programação feita pelo usuário, o Arduino não precisa mais estar conectado ao computador. O Arduino executará o *sketch* criado, desde que seja ligado a uma fonte de energia (SOUZA, 2013).

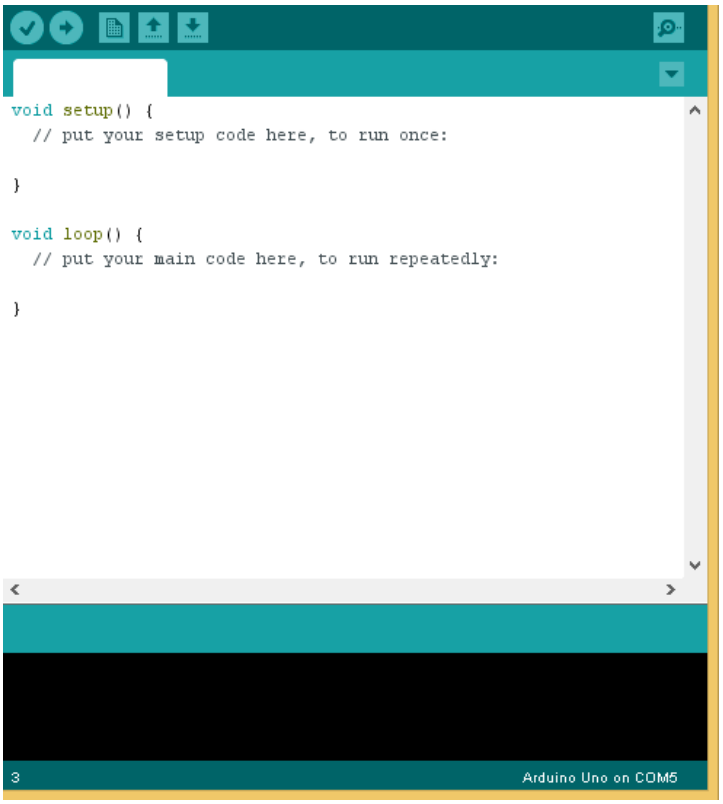

Figura 12 IDE ARDUINO.

Fonte: Arquivo Pessoal

Conforme já citado, foi definida a metodologia de controle baseada no modelo mestre-escravo sendo o microcontrolador empregado o Arduino Uno. Para construção do algoritmo de controle deste trabalho, foi feito uso do método de programação

sequencial por máquinas de estados finitos, e um breve referencial bibliográfico será descrito a seguir.

3.3 MÁQUINA DE ESTADOS FINITOS

Pode-se definir uma máquina de estados finitos (MEF) como um modelo de comportamento de um determinado processo composta por estados, transições e saídas (SILVA, 2011; YANO, 2012).

Segundo (SILVA, 2011), estados, transição e saída são definidos por:

- Estados: Comporta-se como uma memória, ou seja, armazena as informações sobre as saídas em determinado momento;
- Transição: é a condição para que ocorra a mudança de um estado para outro;
- Saída: descreve a atividade que deve ser realizada em um determinado estado.

A cada instante, a máquina pode estar em somente um de seus estados. Em resposta a um evento de entrada, a máquina gera um evento de saída e muda de estado. Tanto o evento de saída gerado quanto o novo estado são definidos unicamente em função do estado atual e do evento de entrada (YANO, 2012).

Para (TATAI, 2002), formalmente, uma máquina de estados finitos é definida como sendo a seguinte tupla:

$$(E, X, f, x0, F)$$

Na qual:

- E é um conjunto finito de eventos;
- X é um conjunto de estados finito;
- f é uma função de transição de estado, $f : X \times E \rightarrow X$;
- x0 é um estado inicial, $x0 \in X$
- F é um conjunto de estados finais, $F \subseteq E$

No entanto, é conveniente representar de forma gráfica uma máquina de estados finitos, o que é feito através de um diagrama de transição de estados. Esse diagrama nada mais é que um grafo direcionado no qual os nós (representados por círculos) representam os estados e os arcos representam eventos. Assim sendo um arco rotulado como '*e'* que conecta dois nós saindo de 'X' e indo para X0 representa uma transição do estado 'X' para o estado X0 quando da ocorrência do evento '*e'* (TATAI, 2002).

A figura 13 mostra um exemplo de máquina de estados desenvolvida para um tipo de jogo físico.

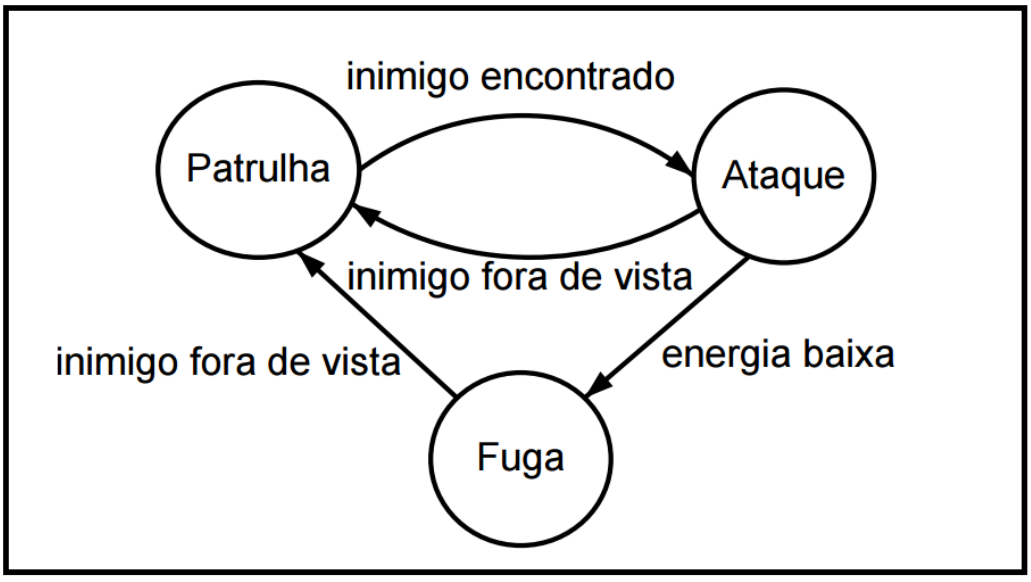

Figura 13 Exemplo de uma máquina de estados.

Fonte: (TATAI, 2002)

Após o estudo realizado neste capítulo, foi determinada toda a metodologia do trabalho e os materiais que foram necessários. Essas informações são descritas no próximo capítulo.

4 MATERIAIS E MÉTODOS

A seguir será feito um estudo da planta solar do IFMG - Formiga, indicando seus problemas e pontos de melhora.

4.1 SISTEMA SOLAR IFMG-FORMIGA

O início deste trabalho consistiu em estudar e identificar os problemas da planta solar construída no Instituto Federal de Minas Gerais (IFMG) – Campus Formiga. Sistema este construído à partir do Edital 007/2012 do programa institucional de fomento à pesquisa aplicada, de título: “Estudo Sobre a Energia Solar e Suas Aplicações à inclusão Social da População de Baixa Renda e ao Programa ‘Luz Para Todos’”.

A planta solar do IFMG - Formiga, tem o objetivo de aumentar a eficiência do aquecedor de água quente comercial através da abertura e fechamento de uma válvula solenoide e também reaproveitar a água fria retida no encanamento do reservatório de água quente, que anteriormente era desperdiçada, utilizando um reservatório e uma bomba d’água, como mostra a figura 14.

Figura 14 Reservatório, Bomba d'água e Válvula Solenoides.

Fonte: Arquivo Pessoal.

Ilustrada pela figura 15, a planta é composta por:

- Um aquecedor de baixo custo: Construído através de material reciclado, e destinado a população de baixa renda;
- Um aquecedor Comercial: Construído a partir de equipamentos industrializados e adaptado para as aplicações de controle;
- Um sistema fotovoltaico *off-grid*, ou seja, não conectado à rede elétrica, que tem a função de fornecer energia aos sensores, controladores e atuadores da planta.

Figura 15 Planta Solar do IFMG.

Fonte: Arquivo Pessoal

4.1.1 ARQUITETURA DA PLANTA SOLAR

A planta do IFMG-Formiga, é composta por aquecedor de baixa renda e por um aquecedor comercial, este trabalho utilizou apenas o segundo. A seguir será mostrado a arquitetura pela figura 16 e descrita a lógica de controle da planta solar.

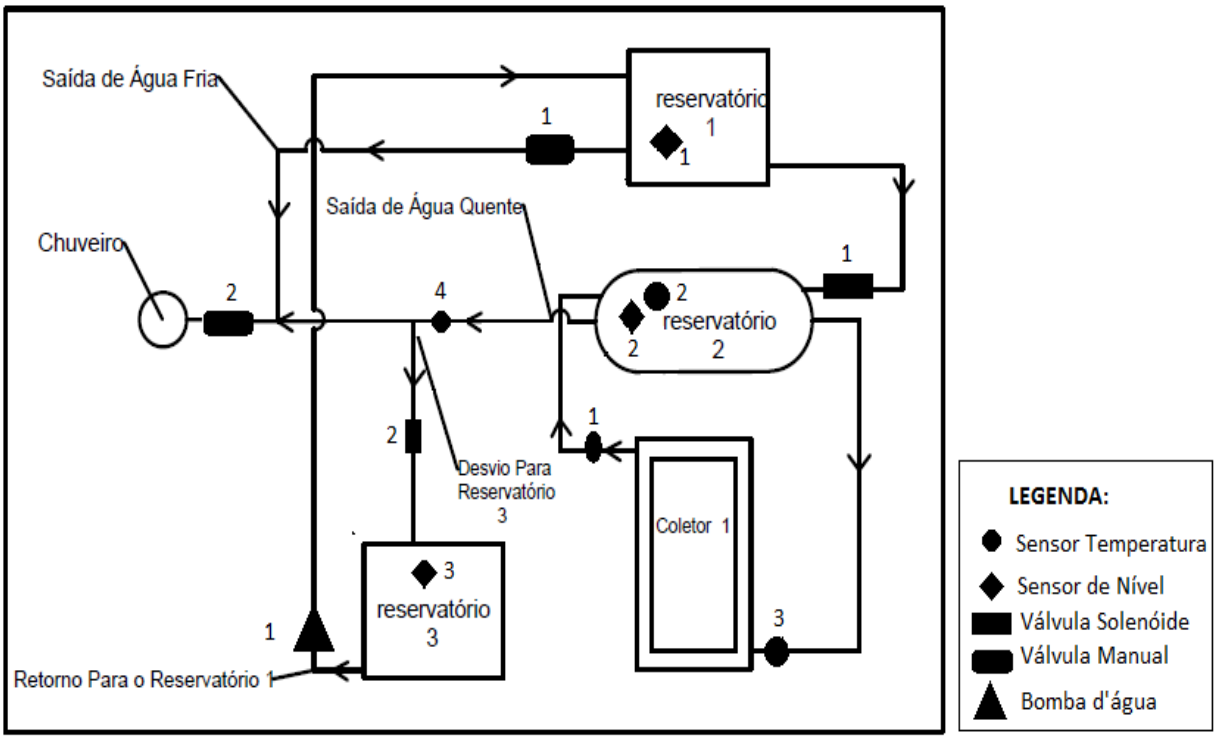

Figura 16 Arquitetura da Planta solar.

Fonte: Arquivo Pessoal

A arquitetura da planta possui quatro sensores de temperatura, três sensores de nível, um reservatório de água quente, um reservatório de água fria principal e outro para armazenamento, duas válvulas solenóides, uma bomba d'água, duas válvulas manuais e um controlador responsável por coordenar todos os elementos da planta.

A lógica de controle possui duas vertentes: fazer com que aumente a eficiência do aquecedor solar, e reaproveitar a água fria dos encanamentos de água quente. A maneira para efetuar estes procedimentos será descrita, mas antes, deve-se definir o significado de nível baixo, alto e médio adotados. O nível corresponde à altura de coluna d'água medida pelo sensor de nível. A tabela 1 mostra a definição dos níveis.

Tabela 2 Definição de Níveis Alto, Médio e Baixo.

Status	Escala
Nível Alto	H > 80%
Nível Médio	H entre 40% e 80%
Nível Baixo	H < 40%

Onde H é a altura de coluna d’água do reservatório.

Para garantir que seja aproveitado todo volume de água quente, sem que haja perda de temperatura, é controlado a abertura e fechamento da válvula solenóide 1. Para isso, o controlador deve garantir as seguintes condições:

- Caso a temperatura da água de saída do Coletor 1, medida pelo sensor de temperatura 1, for maior do que a temperatura de entrada do mesmo, medida pelo sensor de temperatura 3;
- O volume medido do nível no reservatório 2 não for Alto: a válvula solenoide 1 é aberta; caso qualquer uma destas condições não for atendida, a válvula permanecerá fechada.

A comparação entre as temperaturas de entrada e saída do coletor, garante ao controlador que a água fria vinda do reservatório 1 para o reservatório 2 será aquecida. Conhecer o nível do *boiler* é necessário para evitar o acionamento da válvula 1 sem necessidade, evitando assim, desperdício de energia. Já para o reaproveitamento de água, a lógica de controle é a seguinte:

- Ao ligar o chuveiro, a válvula solenóide 2 é aberta, escoando assim toda água fria retida na tubulação de água quente para o reservatório 3. Quando o sensor de temperatura 4 indicar ao controlador que a temperatura no encanamento for igual a temperatura do reservatório 2, medida pelo sensor de temperatura 2, a válvula é fechada, assim a água quente é encaminhada para o uso.
- A água desviada para o reservatório 3 pode ser reaproveitada para uso geral ou pode ser direcionada para o reservatório 1. Para isso, o controlador deverá receber a informação de que o volume medido pelo sensor de nível 1 está “médio” ou “baixo” e o volume medido pelo sensor de nível 3 indica volume “alto” ou “médio”. Quando isso acontecer, o controlador deverá acionar a bomba d’água 1, ligada ao reservatório 3.

É importante ressaltar que, apesar da arquitetura estar preparada para a instalação do chuveiro e também para receber a lógica do primeiro tópico do controle de reaproveitamento de água, a planta solar do IFMG-Formiga não possui estes equipamentos. No entanto, ela pode ser observada em (PEREIRA, 2014). O referido trabalho se baseia no misturador de água automático para banhos e completa a lógica de reaproveitamento d’água e do controle de água quente.

Por não possuir o chuveiro e o encanamento de saída para água quente, o reservatório 3 está sempre em nível alto para poder redirecionar a água deste para o reservatório 1 quando necessário.

Embora o projeto de pesquisa tenha conseguido atingir seus objetivos, alguns problemas puderam ser identificados e são listados na sequência.

4.1.2 PROBLEMAS ENCONTRADOS

O sistema de controle da planta solar, consiste em um único controlador *Arduino Mega* responsável por receber todas as informações de sensores, realizar o processamento dos dados e acionar os atuadores, caso necessário.

O sistema centralizado com apenas um único controlador torna o sistema com baixa confiabilidade. Como todo processo depende dele, e caso haja algum problema em seu funcionamento, toda a rotina será comprometida. Outro problema é a limitação do número de sensores e atuadores que podem fazer parte do sistema. Este número é limitado pelas entradas e saídas presentes no *Arduino Mega.*

O fato de se ter apenas um controlador, invariavelmente, acarreta em grandes distâncias entre ele o os pontos de medição. Este problema reflete em duas consequências graves. A primeira é de que cabos longos podem gerar ruídos nas medições. Isso ocasiona incertezas e imprecisão nos valores medidos, o que pode provocar erros no controle do sistema. A figura 17 mostra o resultado da aferição do sensor de temperatura LM35 utilizando um cabo de apenas 30 metros, identifica evidente o ruído do sinal.

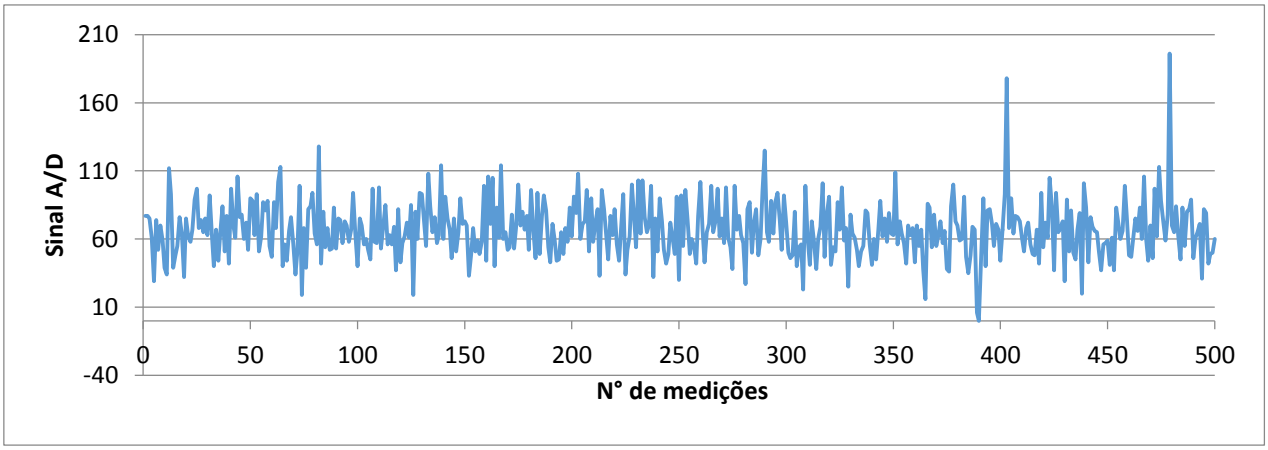

Figura 17 Ruídos ocasionados pela distância entre o LM35 e o controlador.

Fonte: Arquivo Pessoal

O segundo problema gerado pelos longos e excessivos cabeamentos, ilustrado pela figura 18, é de que eles geram gastos desnessários, ao utilizar outros métodos, como o proposto neste trabalho, isso pode ser evitado.

Figura 18 Sistema Centralizado de Controle da Planta solar.
Fonte: Arquivo Pessoal

Este capítulo também trata, em detalhes, sobre os equipamentos utilizados no controle da planta, a metodologia do trabalho e o método de programação.

4.2 SENSORES

Os sensores empregados nesse trabalho foram o sensor de nível MPX 5010, e o sensor de temperatura LM35. Ambos descritos a seguir.

4.2.1 MPX 5010

O sensor de nível que indica aos microcontroladores o volume de água dos reservatórios de água quente e fria é o MPX 5010 (Figura 19). Este sensor varia o seu sinal de saída a partir da variação da altura de coluna d'água comparando a pressão de entrada com a pressão atmosférica (MOTOROLA, 2002).

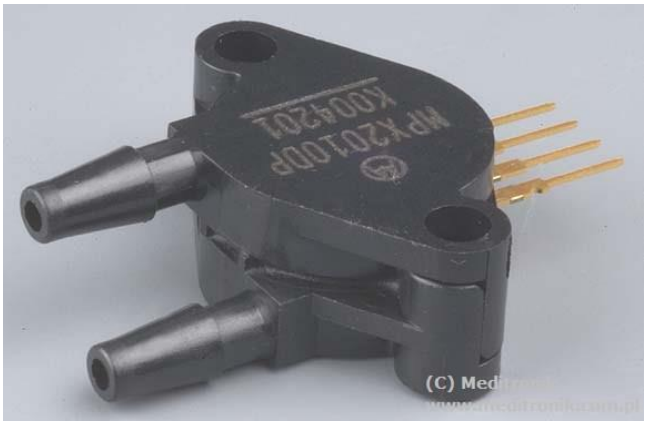

Figura 19 Sensor de Nível MPX 5010.

Fonte: (Meditronik, 2010)*

Para definir o modelo matemático do sinal de saída deste sensor, importante para conhecer o nível de água dos reservatórios, foram feitas medições que consistiram na variação da altura de coluna de água, refletindo na variação do sinal digital recebido pelo Arduino. O resultado deste procedimento pode ser observado na figura 20.

* Figura 19, Disponível em: http://www.meditronik.com.pl/doc/mini/mpx2010dp.jpg, Acesso em Abril,2015.

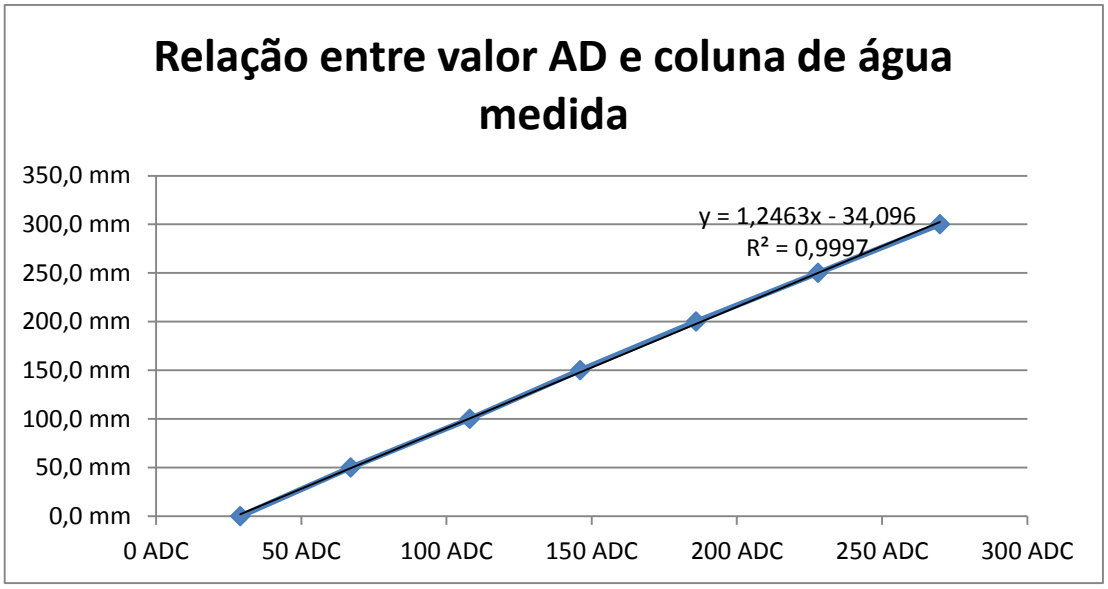

Figura 20 Perfil do sensor de nível MPX 5010.

Fonte: Arquivo pessoal.

Pode-se observar na figura 20 que o coeficiente R^2 é de 99,97%, este coeficiente indica quanto o modelo foi capaz de explicar os dados coletados, quanto mais próximo de 1, melhor é o modelo. Isso significa que a variação da altura de coluna d'água é proporcional a variação do sinal de saída do sensor e que a equação que define a reta é:

$$H = 1{,}2436S - 34{,}096 \quad (1)$$

Onde H é a altura de coluna d'água e S é o sinal (A/D) de saída do sensor.

4.2.2 SENSOR DE TEMPERATURA LM 35

As séries LM35 são sensores precisos de temperatura, na qual a saída de tensão é linearmente proporcional a escala Celsius (°C) de temperatura. Ele não requer nenhum tipo de calibração externa, atua em uma faixa de -55°C e 150 °C (SEMICONDUTOR,1994). Suas características essenciais para este trabalho são:

- Escala Linear, sendo +10 mV/°C;

- Precisão de 0.5°C à 25°C;

4.3 METODOLOGIA DO TRABALHO

Como já informado este trabalho não tem o objetivo de modificar a dinâmica de controle já citada, mas sim propor um meio de comunicação baseado no método Mestre-Escravo, a fim de reduzir o número de cabos, reduzir a distância entre os sensores e o controlador além de aumentar a confiabilidade do sistema.

A proposta sugerida é: a partir do uso de dois microcontroladores escravos, instalados em pontos estratégicos da planta, é feita a coleta das informações dos sensores para que seja feita toda a tomada de decisão do controlador Mestre. Foram utilizados para isso três Arduinos Unos.

Os Arduinos possuem comunicação serial e podem se comunicar ponto a ponto sem que haja necessidade de um padrão específico de comunicação. No entanto se um terceiro componente necessitasse se comunicar com os outros dois não seria possível. Para suprimir esta necessidade, foram estudados padrões de comunicação serial e paralelo. O padrão escolhido foi o de comunicação serial RS-485 e os motivos serão apresentados a seguir.

4.3.1 PADRÃO DE COMUNICAÇÃO SERIAL RS-485

Após estudar os padrões de comunicação seriais descritos no referencial teórico deste trabalho, para a construção da rede de comunicação mestre-escravo decidiu-se então utilizar o padrão RS-485. Os motivos, seguem enumerados:

1. Os três padrões (RS-232, RS-422 e RS-485) são capazes de diminuir a quantidade de cabos do sistema e a distância entre os sensores e o

controlador. No entanto, o Padrão RS-232 limita o número de controladores em dois e ainda, só garante uma comunicação de no máximo 15 metros entre eles.

2. Com isso, restam os padrões RS-422 e RS-485. Ambos permitem uma comunicação a longas distâncias, até 1200 metros, e permitem o número necessário de controladores que a planta solar do IFMG-Formiga necessita (três). No entanto, a comunicação feita pelo padrão RS-422 utiliza o método *full-duplex.* Este método necessita de dois pares de fios trançados e garante até 10 controladores ligados no barramento.
3. Como um dos principais objetivos deste trabalho é a redução do número de cabos do projeto, o padrão RS-485 foi escolhido justamente por necessitar de apenas de um par de fio trançado, já que sua comunicação se dá pelo método *half-duplex.* Outro fato decisivo foi o número de componentes que podem ser ligados ao barramento de comunicação: 32. Isso garante margem para que o projeto seja ampliado sem que um novo método de comunicação e controle se faça necessário.

É importante ressaltar que o padrão de comunicação paralelo foi descartado por dois motivos: o primeiro é que o controlador escolhido para este trabalho, *Arduino Uno,* trabalha em comunicação serial. O segundo é que um dos objetivos deste trabalho é exatamente a redução no número de cabos para comunicação e assim como do ruído provocado por eles, e estes problemas são característicos deste padrão.

Definido o padrão de comunicação que será utilizado pelo mestre e escravos, deve-se construir o meio físico, e para tanto, utiliza-se o circuito integrado (CI) MAX 485 destinado especificamente para comunicações seriais RS-422 e RS-485.

4.3.2 BARRAMENTO DE COMUNICAÇÃO RS-485

Um transmissor / receptor (transceptor) RS-485 traduz um sinal lógico TTL (0 a 5 Volts) em outros dois sinais, denominados de A e B, que podem ser também chamados de Data+ e Data-. O sinal A possui a mesma lógica do sinal TTL, enquanto que o sinal B é complementar. A informação do sinal de entrada está codificada no sinal A-B, ou seja, na diferença entre os sinais A e B. Caso a diferença entre eles seja superior a 200mV, tem-se nível lógico 1 (SILVA, 2013), como mostra a figura 21.

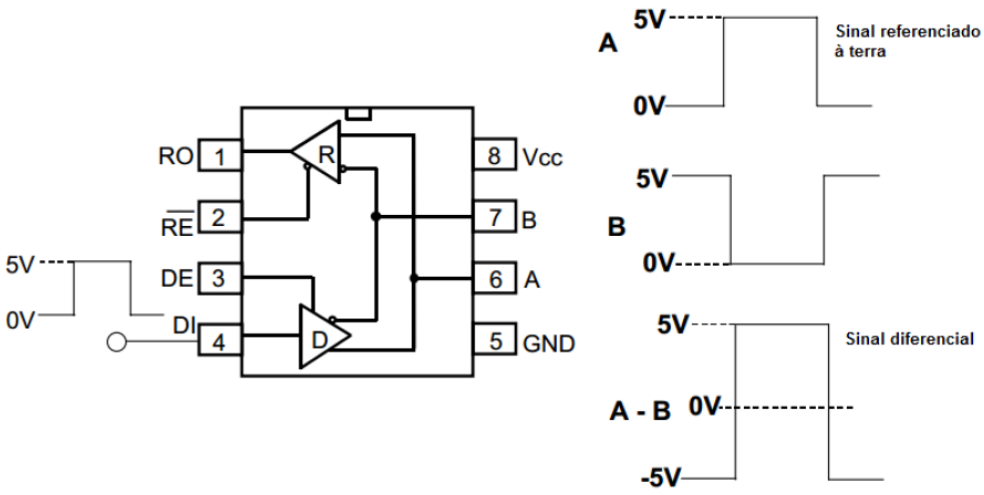

Figura 21 CI MAX485 e barramento e sinais diferenciais.
Fonte: (SILVA, 2013)

Em casos em que a diferença for inferior a -200mV, então considera-se nível lógico 0. No intervalo de -200mV a 200mV o nível lógico é indefinido, servindo também como meio de detecção de cabo solto, para alguns transceptores comerciais (SILVA, 2013).

Uma das vantagens da transmissão diferencial é sua robustez à interferência eletromagnética. A conexão entre dispositivos RS-485 é feita por cabos de par trançado, com resistores de terminação para balanceamento, ilustrado pela figura 22.

Dessa forma, se um ruído é introduzido na linha, ele é induzido nos dois fios de modo que a diferença entre A e B é quase nula.

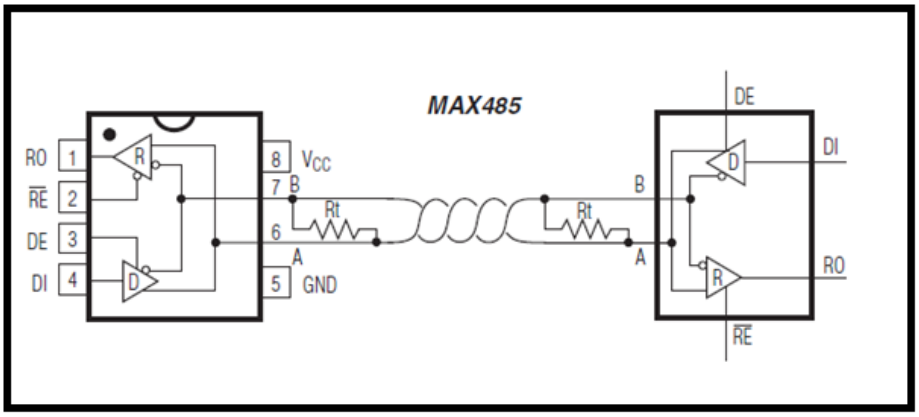

Figura 22 Par de fios trançados ligando dois componentes.
Fontes: (MAXIM, 2003) (Modificado)

Pode ser adicionado ao par de fios trançados, um terceiro: o terra. Ele pode ser dispensado, mas seu uso garante que a tensão de modo comum no receptor fique dentro de uma faixa segura e isso aumenta a estabilidade e imunidade a ruídos da interface (AUTOMALABS, 2013; SILVA, 2012).

Em resumo pode-se definir as entradas do CI MAX485 da figura 22 como: O pino 2 (**RE**) é responsável por habilitar o driver de recepção **R**, sendo ativo no nível 0; O pino 3 (**DE**) habilita o driver de transmissão **D** (ativo em 1). Os pinos **RO** e **DI** representam saída da recepção e driver de entrada respectivamente, e trabalham com níveis lógicos TTL (0 a 5Volts). Já as saídas **A** e **B** para o barramento operam com tensão diferencial entre seus terminais. O transceptor normalmente é alimentado com 5 Volts Corrente continua. Como o padrão RS485 é um *half-duplex*, é comum conectar o pino 2 com o pino 3 transformando-os em um único pino. Por exemplo: quando for aplicado nível lógico 1 a linha (te), o transmissor (pino 3) habilita a linha TX, e ao mesmo tempo o receptor (pino 2) desabilita a linha RX (SILVA, 2013).

Baseado pelo CI MAX485 e todo o estudo feito neste trabalho sobre padrão de comunicação serial RS-485, foi possível realizar a comunicação entre os *Arduinos* escravos (slave) e o *Arduino* mestre (master), como representada na figura 23.

Como mostra a figura 23, não foram utilizados os resistores entre os canais A e B nas extremidades do barramento de comunicação. Isso se deve ao fato de que os *shields* foram construídos para serem genéricos, ou seja, todos podem se comportar como mestre ou escravo e ser anexado em qualquer ponto do barramento.

Para redução dos ruídos, que podem ser causados pelas tensões e correntes de *off-set* (características dos Amplificadores Operacionais reais), foram utilizados dois resistores conectando o canal A ao Vcc (R1) (tensão de alimentação 5 volts) e o canal B a referência terra (R2) (BOYLESTAD, 2004). Então a nova equação do sinal diferencial pode ser dada por:

$$\boldsymbol{Vs = \frac{R1}{R2}(A - B)} \qquad (\mathbf{2})$$

Para que os resistores não causem interferência no sinal, eles devem ter o mesmo valor. O resultado final pode ser observado na figura 23, e os componentes utilizados foram:

- 6 Resistores 680Ω;
- 3 Resistores 1 kΩ;
- 3 CIs MAX 485;
- 3 Arduinos Uno e
- 1 par de fios trançados compartilhado pelos controladores.

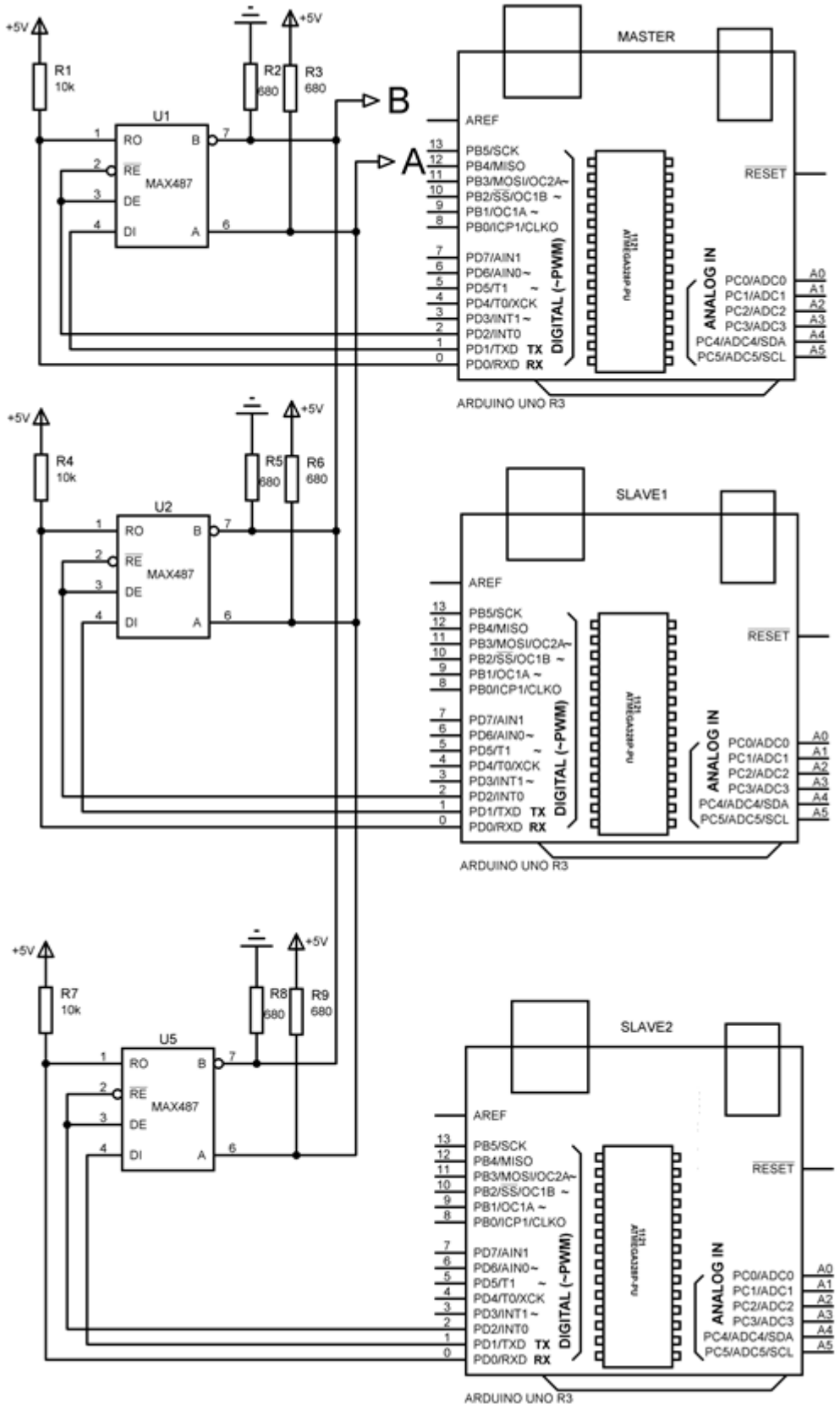

Figura 23 Ligação entre os Mestres e Escravos pelo padrão serial RS-485.

Fonte: Arquivo Pessoal

A seguir será descrita como foram realizadas as programações do mestre e dos escravos do sistema.

4.3.3 PROGRAMAÇÃO

A estrutura do algoritmo baseia-se em máquinas de estado finitos, que pode ser definida como: um circuito sequencial genérico que consiste de uma seção feita a partir de lógica combinacional como mostram as figuras 24 e 25.

A máquina de estado do Arduino mestre possui seis etapas. O ciclo é repetitivo e foi programado para ocorrer a cada 30 (trinta) segundos. Os estados são definidos por:

A. Enviar Endereço: Neste estado, o mestre envia um byte ao barramento, correspondente ao endereço do escravo com quem ele deseja se comunicar;

B. Enviar Requerimento: Nesta etapa, o Arduino mestre envia um byte que indica ao escravo a ação que o mestre deseja que ele execute (medição de um sensor ou atuação de uma válvula ou bomba d'água);

C. Caso seja requisitado uma informação, o mestre permanecerá no estado "Aguardar dados do sensor" até que o escravo lhe envie um byte com a informação desejada, em seguida o mestre retorna ao estado "Enviar Requerimento". Essa ação é repetida até que o mestre tenha todas as informações dos sensores;

D. Processar: Após receber todas as informações desejadas e requisitadas, o *Arduino* mestre processará seu algoritmo de controle e poderá tomar as suas decisões. Depois de processar os dados, o mestre pode se direcionar para dois estados diferentes;

E. O primeiro é o estado “Mudar Endereço”. Caso o controlador mestre, após processar todas as informações recebidas, identifique que a planta não deva sofrer nenhuma alteração, ele informa ao escravo que ele pode voltar ao estado inicial e inicia a comunicação com outro escravo de endereço diferente;

F. O segundo é o estado “Mandar Atuar”. Este estado é responsável por solicitar ao escravo que seja acionado algum de seus atuadores. Logo após, o mestre é direcionado ao estado “Mudar de Endereço”.

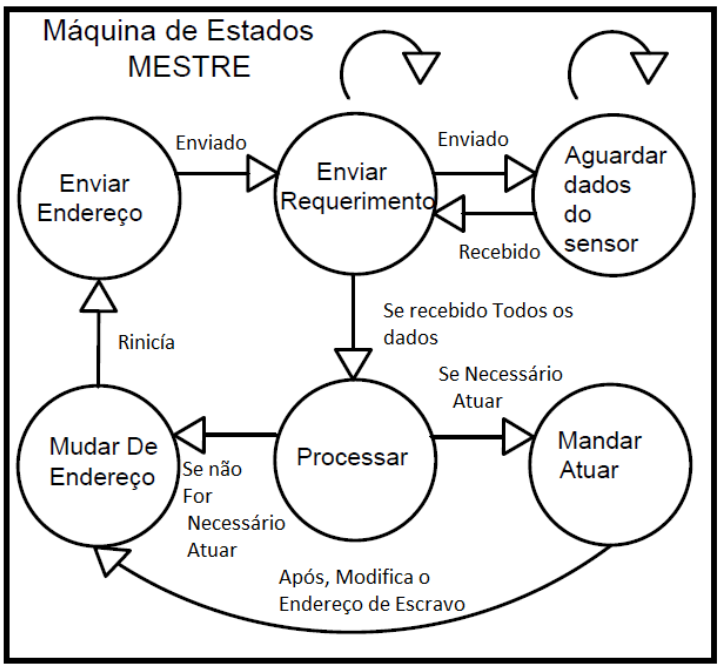

Figura 24 Máquina de Estados Arduino Mestre.

Fonte: Arquivo Pessoal

Os escravos seguem uma máquina de estados que completa a do mestre. Seus estados são:

A. Reconhecer Endereço: Neste estado, o *Arduino* aguarda o endereço enviado pelo mestre no barramento serial RS485. Quando isso acontece, ele altera seu estado para “Aguardar Requisição”;

B. Aguardar Requisição: Neste estado, o escravo pode seguir por duas linhas de atuação. Pelo endereço recebido, ele Seguirá para o estado “Enviar dados” ou para o estado “Atuar”;

C. Enviar dados: Neste estado é feito o envio dos valores medidos pelo *Arduino* escravo. Em seguida, ele volta ao estado “Aguardar Requisição” esse processo se repete até que o mestre indique ao escravo que as medições acabaram;

D. Atuar: Este estado recebe a solicitação do mestre e aciona algum dos atuadores. O escravo conhece o limite superior dos níveis, assim ele só irá parar de atuar quando o nível desejado for atingido. Depois disso, o escravo volta ao estado “Reconhecer Endereço”.

É importante ressaltar que, para evitar o controlador não fique preso a qualquer estado, se utiliza o temporizador contador de cinco segundos. Caso o mestre ou o escravo não recebam a informação neste tempo, o *Arduino* faz com que o componente volte ao primeiro estado.

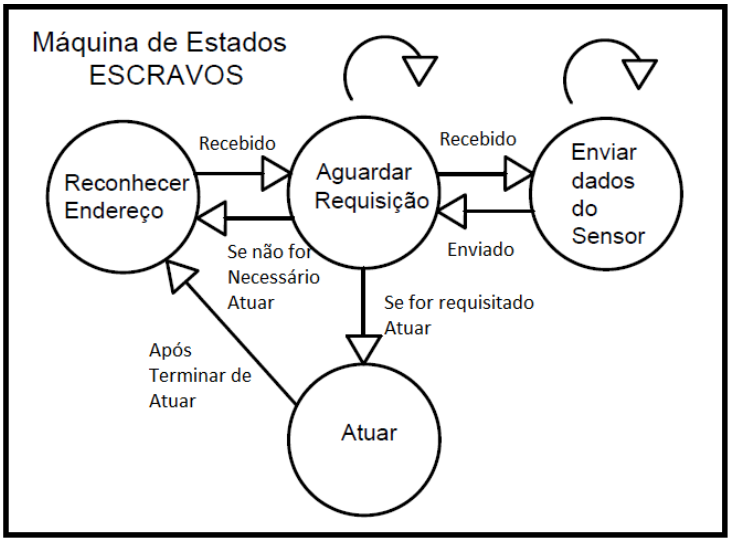

Figura 25 Máquina de Estados Arduinos Escravos.

Fonte: Arquivo Pessoal

Conhecido todos os estados do mestre e de seus escravos é preciso definir os endereços de cada componente. Para isso foi construída uma tabela presente no Anexo C que indica todos os endereços do sistema.

O pacote de dados enviados pelo mestre e pelo escravo podem ser observados nas figuras 26 e 27 respectivamente.

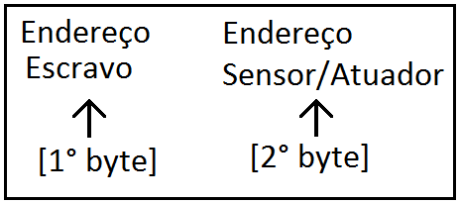

Figura 26 Pacote de dados enviado pelo Controlador Mestres.
Fonte: Arquivo pessoal.

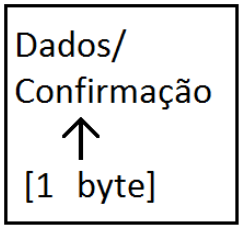

Figura 27 Pacote de dados enviado pelos Escravos ao mestre.
Fonte: Arquivo pessoal.

Nota-se que o mestre precisa enviar dois bytes para comunicação com determinado escravo. O primeiro é responsável por identificar com qual escravo ele deseja se comunicar, e o segundo define a ação que deve ser executada.

O escravo por sua vez envia apenas um byte, informando ao mestre o valor do sensor desejado ou se a ação desejada (acionar um dos atuadores, por exemplo) foi recebida. No anexo C, são encontrados os endereços de cada componente e nos anexos A e B estão os algoritmos construídos para o controlo dos componentes da planta.

5 RESULTADOS E DISCUSSÃO

Neste capítulo, serão discutidos os resultados alcançados com este trabalho. O primeiro deles, que será descrito a seguir, foi a construção de uma placa de circuito impresso que auxilia o microcontrolador Arduino para que o padrão RS-485 possa ser executado. Estas placas se denominam *shields.*

5.1 *SHIELD* RS-485 PARA ARDUINO UNO

Durante o trabalho, foram desenvolvidos três *shields* para Arduino uno. Estes se baseiam no Circuito Integrado (CI) MAX-485 que é exclusivamente dedicado para a construção dos barramentos de comunicação serial utilizados pelos padrões RS-422 e RS-485, como explicado na seção 4.3 deste trabalho.

Cada *Shield* possui:

- 10 entradas/saídas digitais;
- 4 pinos analógicos;
- 1 canal de entrada do barramento;
- 1 canal de saída;

O Arduino uno tem capacidade para utilizar treze canais digitais e seis canais analógicos, porém estes *shields* utilizam três canais digitais para a comunicação serial, disponibilizando para livre uso ao usuário dez canais cada. São utilizados apenas quatro canais analógicos devido a necessidade do projeto, mas as outras duas portas poderiam incorporar ao equipamento. A figura 28 mostra os *shields* em seus estados finais.

Como descrito no referencial teórico, o padrão de comunicação, RS-485, necessita apenas de um par de fios trançados. Apesar de não necessitar de que todos estejam conectados à mesma fonte de energia, foi utilizado um terceiro fio, conectando o potencial de referência das três placas. A figura 29 mostra os três controladores conectados.

Figura 28 *Shield* Desenvolvido para padrão serial RS-485 para Arduino uno.
Fonte: Arquivo pessoal.

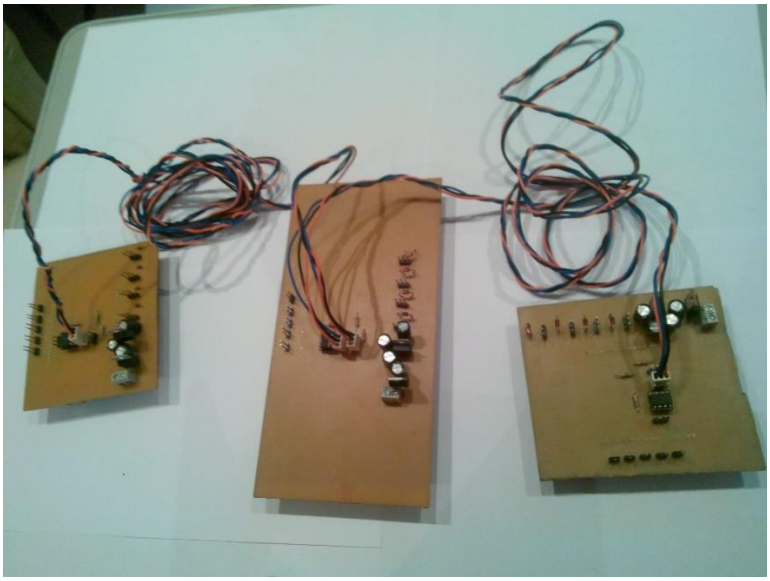

Figura 29 Controlador Mestre (centro) e Escravos conectados.
Fonte: Arquivo Pessoal.

A alimentação do *shield* pode variar de 12 a 20 volts, e os sensores podem ser alimentados pelo próprio prototipo. Ele possui dois reguladores de tensão, um de 9 volts para alimentar o Arduino, e outro de 5 volts para fornecer energia aos sensores. Em paralelo a alimentação dos sensores existe um diodo Zenner de 5,1volts, protegendo assim os mesmos e as entradas do controlador.

5.2 ESCRAVOS

Um dos objetivos deste projeto é diminuir a distância entre os sensores e atuadores do controlador. Para isso os escravos foram instalados em pontos estratégicos da planta.

O escravo 1 é responsável por:

- Fazer a leitura dos sensores de temperatura da entrada e saída do coletor solar;
- Fazer a leitura do nível de água do reservatório de água quente (boiler);
- Realizar, se necessário, o acionamento da válvula solenoide responsável pelo fluxo de água fria do reservatório principal para o boiler.

Utilizando a mesma estratégia de aferição do sensor de temperatura, citada no tópico 4.1.2, onde se utilizou um cabo de trinta metros de comprimento, foi aferido o mesmo sensor mas agora com um cabo de apenas dois metros, (distância essa um pouco maior do que a distância entre o escravo 1 e os sensores) chegou-se ao resultado da figura 30.

É possível notar na figura 30, comparada com a figura 5, uma maior confiabilidade dos dados recebidos, uma vez que o sinal de saída do sensor pouco variou.

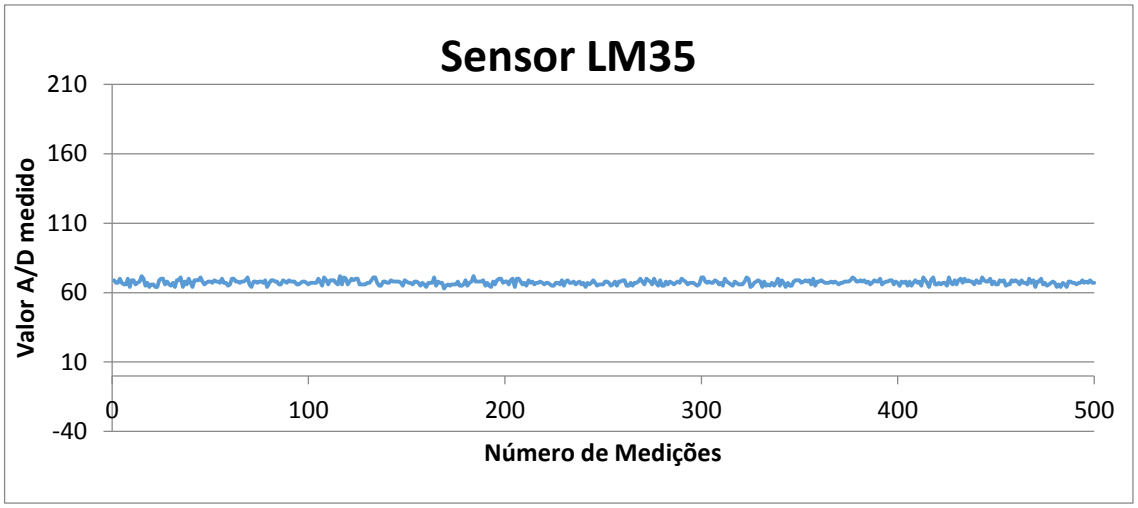

Figura 30 Aferição do sensor LM 35 para um cabo de 2 metros de comprimento.

Fonte: Arquivo Pessoal.

A figura 31 mostra como foi instalado o escravo 1.

Figura 31 *Shields* do escravo 1 instalados na planta solar do IFMG.

Fonte: Arquivo Pessoal.

Já o segundo escravo, é responsável por:

- Medir o nível do volume de água do reservatório principal e do reservatório para reaproveitamento de água;

- Controlar a bomba d’agua que liga esses dois reservatórios.

A figura 32 mostra como o escravo 2 foi instalado.

Figura 32 Escravo 2 instalado.
Fonte: Arquivo pessoal.

O fato de ter dois controladores, responsáveis cada um por uma parte da rotina de controle, torna o sistema mais confiável do que o originalmente instalado. Por exemplo: caso o controlador escravo 2 deixe de trabalhar por qualquer motivo que seja, apenas o controle do reaproveitamento de água será prejudicado, e o controle do aquecedor continuará a funcionar.

A programação de ambos os escravos é praticamente a mesma, diferindo apenas dos endereços de cada um e dos instrumentos ligados a ele. A seguir será descrito como é o comportamento destes escravos.

5.2.1 COMPORTAMENTO DE CONTROLE DOS ESCRAVOS

Os escravos precisam aguardar até identificar que seu endereço tenha sido enviado ao barramento de comunicação para executar alguma de suas ações pré-programadas. Enquanto o mestre não enviar o endereço pré-determinado, o escravo deve permanecer em seu estado de “Aguardo”, como mostra a figura 33.

Ao receber o seu endereço o Arduino espera o segundo comando do mestre. Caso o comando seja a leitura de algum sensor, o valor é lido e informado ao mestre por meio de escalas (Nível alto, Nível médio, Nível baixo, T1 maior, T2 maior). O algoritmo de programação para os escravos encontra-se ao anexo B deste trabalho.

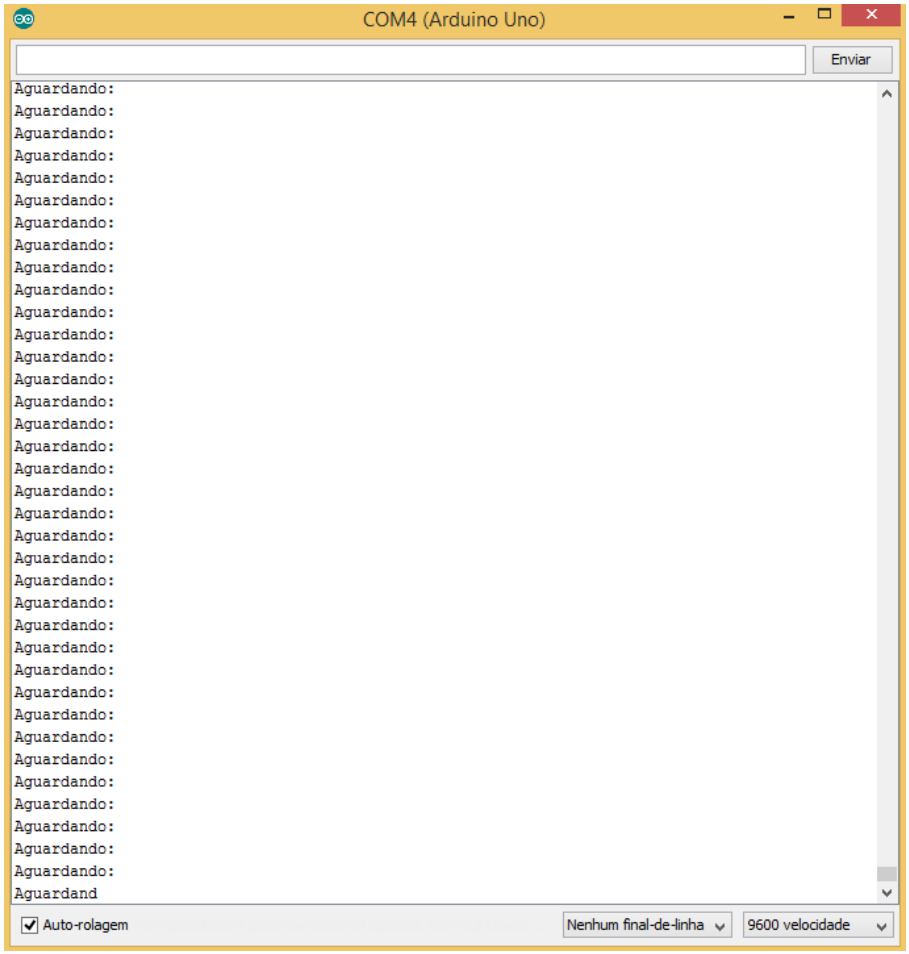

Figura 33 Arduino escravo aguardando o Endereço.

Fonte: Arquivo Pessoal.

5.3 MESTRE

Diminuir os cabos que chegavam ao controlador principal era uma prioridade do trabalho, pois isso acarreta em menor erro de medição e na redução do custo do projeto. A figura 34 mostra o resultado final do novo sistema em comparação ao anterior.

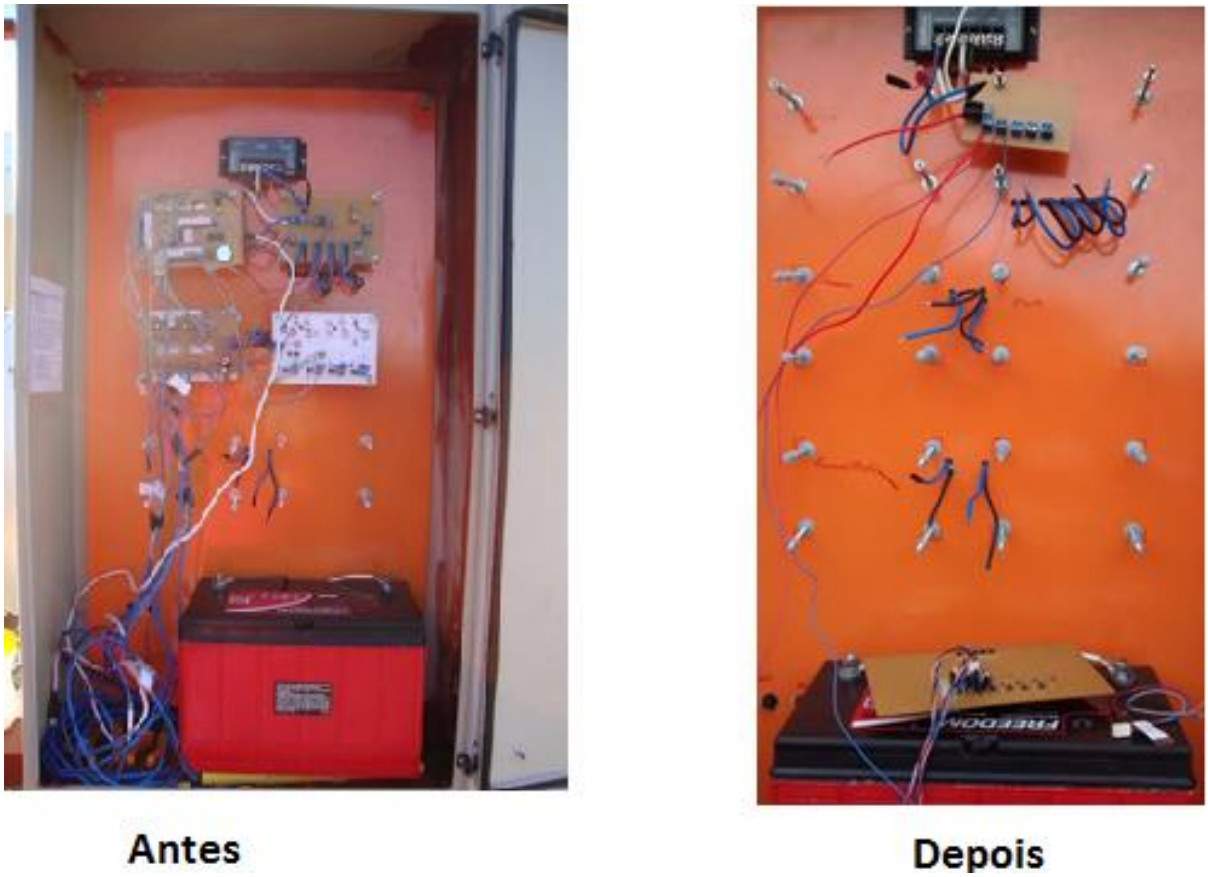

Figura 34 Comparação entre o sistema Anterior e o Proposto.
Fonte: Arquivo Pessoal.

Comparado ao sistema anterior, percebe-se uma grande redução no número de cabos que chegam ao controlador (reduzido para apenas três) e uma menor complexidade do sistema de controle.

O controlador mestre, ilustrado pela figura 35, tem as mesmas características físicas dos controladores escravos. Sua área maior de construção se dá apenas para que seja possível a fixação da mesma no painel da planta solar.

A função do mestre é coletar as informações medidas pelos escravos, processar as mensagens e em seguida efetuar a tomada de decisões, seja ela acionar uma bomba d'água, abrir ou fechar uma válvula solenoide ou até mesmo fazer nada. Sua rotina de comando será descrita a seguir.

Figura 35 Controlador mestre.

Fonte: Arquivo Pessoal.

5.3.1 COMPORTAMENTO DE CONTROLE DO MESTRE

O controlador mestre define com qual escravo deve ser feita a comunicação e para isso, deve enviar o determinado endereço ao barramento de comunicação. Para tanto, ele segue uma série pré-determinada para receber a leitura dos sensores feita pelos escravos, e logo após decidir se deve tomar alguma atitude (acionando algum

dos atuadores). A figura 36 mostra o passo a passo da sequência de leituras que o mestre requisita do escravo 1.

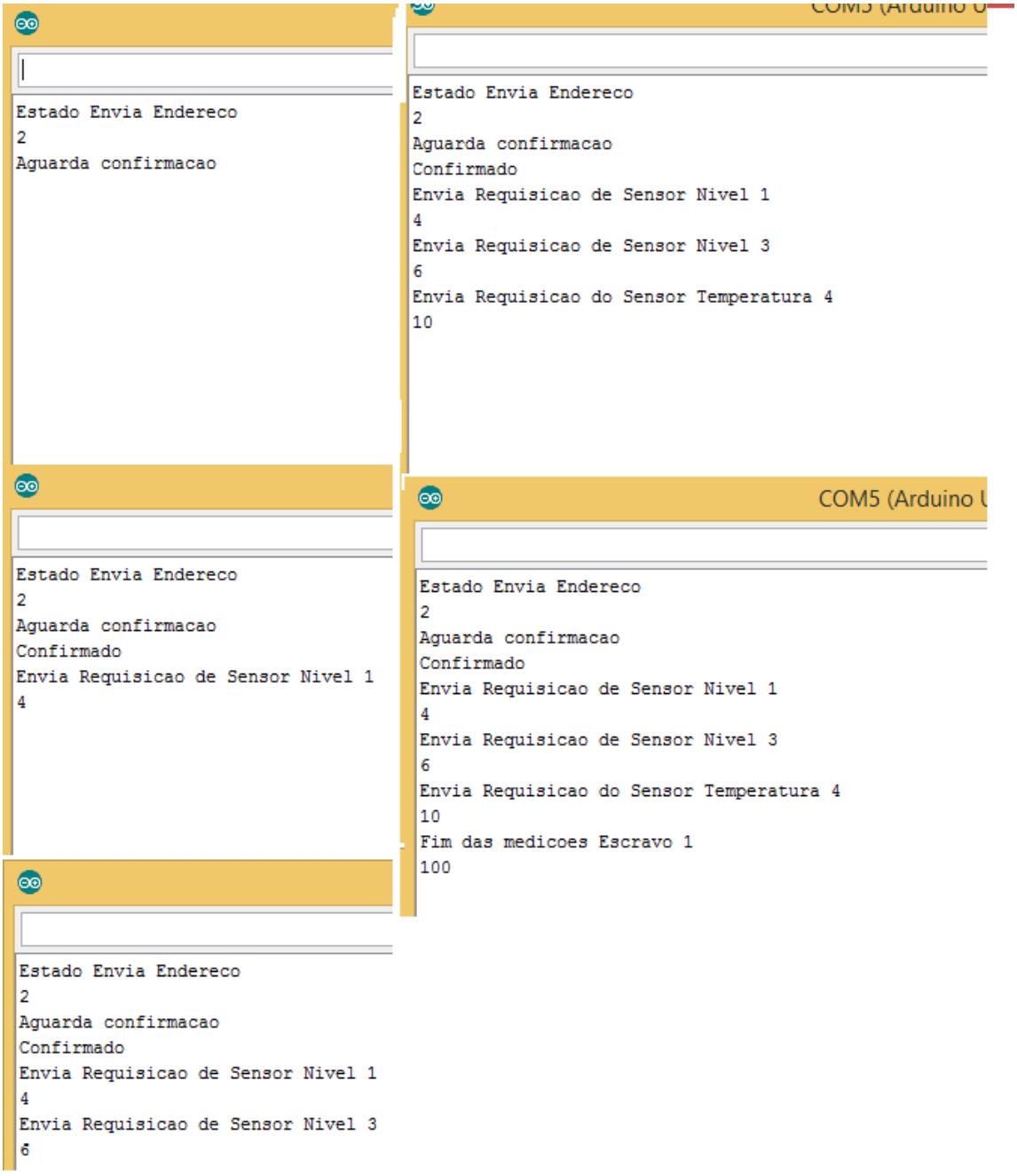

Figura 36 Leituras requisitadas ao escravo 1.

Fonte: Arquivo Pessoal.

Nota-se na figura 36 um número após o nome de cada ação. Este número é o endereço do escravo ou sensor selecionado pelo mestre. É importante não repetir os

endereços para que nenhum outro dispositivo identifique e cause interferência no barramento.

Ao final das leituras, o controlador mestre processa os dados e informa o status da planta solar como mostra a figura 37. Essas condições irão determinar se será preciso enviar uma solicitação de acionamento dos atuadores aos Arduinos escravos.

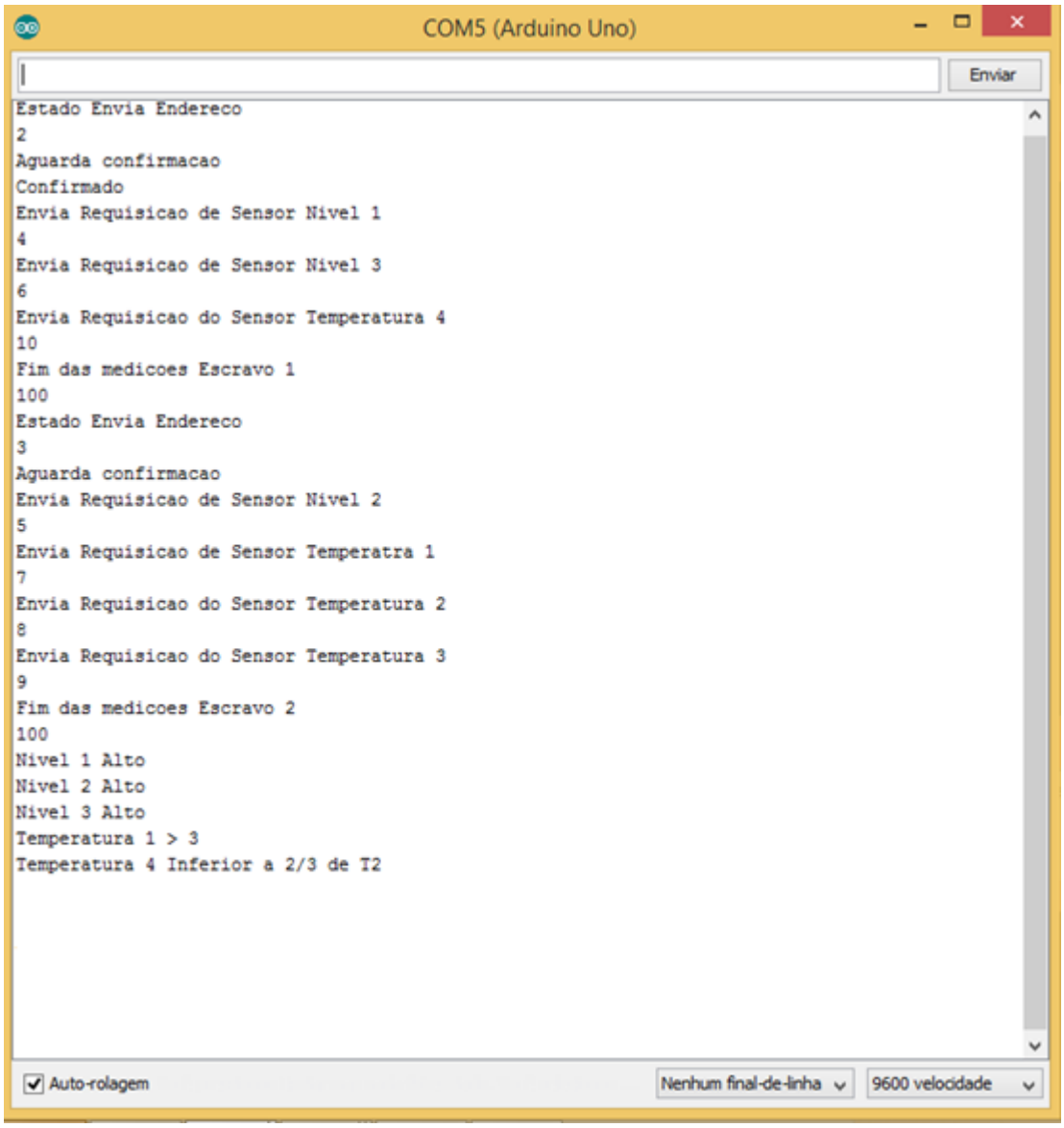

Figura 37 Leituras de todos os dados e Considerações.

Fonte: Arquivo Pessoal.

Ao fim do ciclo, os controladores escravos devem retornar ao estado “Aguardar” da figura 33 até que seu endereço seja solicitado novamente. Já o Arduino mestre aguarda trinta segundos para a próxima medição.

Em virtude de verificar o funcionamento dos atuadores, foi esvaziado o reservatório 1 até o nível médio (um pouco menos de 80%), com isso a bomba 1 foi acionada, enviando água do reservatório 3 (que estava cheio) para o reservatório 1 como mostra a figura 38.

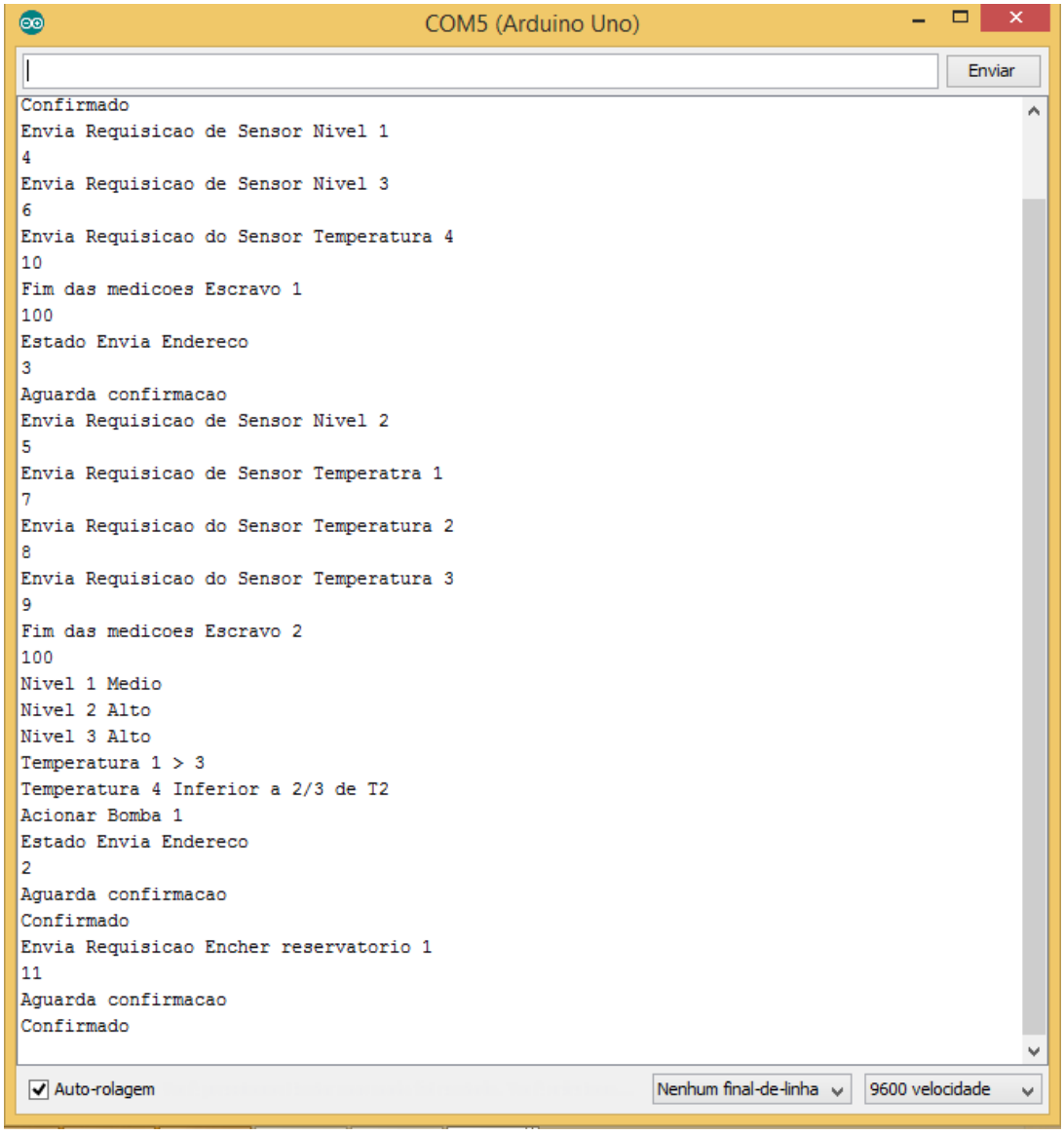

Figura 38 Acionamento da Bomba 1, após esvaziar o reservatório 1.

Fonte: Arquivo Pessoal.

Após um minuto, o sistema apresentou o mesmo resultado da figura 37. Vale ressaltar que o sistema não possui estrutura para saída do chuveiro, portanto não existe o sensor 4 nem tanto a válvula 2 da figura 4. O valor de temperatura do sensor 4 é o medido da temperatura ambiente. Porém a lógica de controle está preparada para encorporar o chuviero ao sistema e mais uma vez, cita-se aqui o trabalho de (PEREIRA, 2014) que trata da inclusão de um misturar de água para banhos, que aliado a este trabalho, melhora ainda mais o controle de água quente.

Para que as figuras 34,35 e 36 pudessem mostrar o passo-a-passo da rotina de programação do mestre, foi preciso fazer uma pequena alteração no programa original, que está em anexo neste trabalho. A mudança feita foi criar um *delay* no algoritmo para que fosse possível mostrar as menssagens sem que houvesse interferência no barramentos de comunicação. O algorítmo original mostra apenas o status da planta e se algum dos atuadores foi acionado. Todo o algorítmo foi baseado em máquinas de estados finitos e está presente no anexo A.

6 CONCLUSÕES

Em tempos de crise energética, como a que foi causada pelo baixo nível dos reservatórios de água das principais hidrelétricas nacionais a partir do ano de 2012, se faz necessário pensar em novas fontes de energia elétrica, e porque não, criar novas tecnologias para otimizar as já existentes. É neste intuito que o trabalho de pesquisa aplicada (OLIVEIRA et al, 2014) e este trabalho foram desenvolvidos.

A planta solar desenvolvida por (OLIVEIRA et al, 2014) conseguiu cumprir com seus objetivos, criando um sistema autônomo de controle de água quente e reaproveitamento de água. Porém, alguns problemas na metodologia de controle poderiam vir a diminuir sua eficiência ou, até mesmo comprometer o seu funcionamento.

Este trabalho identificou alguns problemas e pontos de melhoria da planta solar e propõe um novo modelo, baseado em controle mestre-escravo, padrão de comunicação serial RS-485 e algoritmo sequência em máquinas de estados finitos.

O conjunto proposto neste trabalho não só assegura uma maior confiabilidade para o sistema de controle, como também elimina os problemas gerados pelos excessivos e longos cabos utilizados para a comunicação entre o controlador e os sensores e atuadores.

Outra vantagem é que o método proposto assegura que mais componentes possam integrar ao sistema, podendo, assim, expandir o universo de controle sem que seja necessário modificar a estrutura já existente, e sem grandes alterações no algoritmo de controle.

Ao término do trabalho, pode-se identificar novos trabalhos para ampliar e/ou melhorar o que já existente, e serão brevemente descritos a seguir.

7 TRABALHOS FUTUROS

Seguindo a linha de desenvolvimento do trabalho, alguns trabalhos podem ser desenvolvidos. Por exemplo:

- Desenvolver uma interface homem – máquina (IHM) a partir do sistema supervisório SCADA, para que o processo possa ser controlado e monitorado de maneira remota via computadores, celulares, etc;

- Pode ser desenvolvido um sistema que tenha maior confiabilidade do que este, pois apesar de ter diminuído a fragilidade do sistema atual, o controle da planta continua vulnerável a falhas do controlador mestre. Uma sugestão seria criar uma rede hierárquica onde os controladores escravos identifiquem a ausência do mestre e um novo controlador assume as funções do mestre;

- O sistema misturador de água foi estudado por (PEREIRA, 2014), no entanto a planta solar não possui este sistema. Um novo trabalho seria acoplar a planta um chuveiro e inserir a lógica de controle do misturador a deste trabalho;

- Como sugestão para aumentar a confiabilidade da planta solar, seria o desenvolvimento de um sistema de redundância analógica. Este sistema manteria o funcionamento da planta, mesmo que o mestre fosse desconectado e comprometesse o controle.

8 REFERÊNCIAS BIBLIOGRÁFICAS

ALBUQUERQUE, Pedro Urbano Braga de; Auzuir Ripardo de Alexandria. **Redes Industriais - Aplicações em sistemas digitais de controle distribuído.** 2ª edição. Ed. Ensiono Profissional. São Paulo –SP. 2009.

AUTOMALABS, **Módulo adaptador RS-485 – TTL baseado em MAX485.** Março 2013. Disponível em: < http://www.automalabs.com.br/modulo-adaptador-rs485-ttl-baseado-em-max485/> Acesso em junho, 2015.

ARDUINO. **WHAT IS ARDUINO?** Disponível em: http://www.arduino.cc/en/Guide/Introduction. Acesso em: Maio,2015.

ARDUINO. **Arduino Uno: Overview** Disponível em: http://www.arduino.cc/en/Main/arduinoBoardUno. Acesso em: Maio,2015

BRAGA**,** Newton C. CONHEÇA OS MICROCONTROLADORES PARTE 1 (MIC 001). **Instituto Newton C. Braga, NCB.** Disponível em: <http://www.newtoncbraga.com.br/index.php/microcontroladores/103-microchip->

BOYLESTAD, Robert L. NASHELSKY, Louis. **Dispositivos Eletrônicos e Teoria de Circuitos.** 8ª edição. Editora: Pearson Prentice Hall. São Paulo – SP, 2004.

KRON, Medidores. **Conceitos Básicos de RS-485 e RS-422.** Junho, 2003. Disponível em: < http://pdf.datasheetcataog.com/datasheet/maxim/MAX1487-MAX491.pdf> Acesso em: abril, 2015.

MAXIM. **LOW-POWER, SLEW-RATE-LIMITED RS-485/RS-422 Trasnceivers.**

MOTOROLA, Semicondutor Technical Data. **Integrated Silicion Pressure Sensor On-chip Signal Conditioned, Temperature Compensated and Calibrated.** Motorola, Freescale Semicondutor, Inc. ano: 2002. Disponível em: < http://pdf.datasheetcatalog.com/datasheet2/a/0aiqj65ai5zrwcfcadgglgpsu23y.pdf>. Acesso em Maio, 2015.

OLIVEIRA, I. R. RODRIGUES, M.A.L. **ESTUDO SOBRE A ENERGIA SOLAR E SUAS APLICAÇÕES Á INCLUSÃO SOCIAL DA POPULAÇÃO DE BAIXA RENDA E AO PROGRAMA "LUZ PARA TODOS".** Programa Institucional de Bolsas de Iniciação Científica. Instituto Federal de Ciência Tecnologia e Informação de Minas Gerais IFMG. Formiga, Agosto, 2014.

ONS, Operador Nacional do Sistema Elétrico. **Volume Útil dos Principais Reservatórios**. Disponível em: <http://www.ons.org.br/historico/percentual_volume_util_out.aspx#. Acesso em: Maio,2015>.

PEREIRA, F**. Microcontroladores PIC – Programação em C.** Ed: Ética, 2ª ed. São Paulo, 2003.

PEREIRA, MARCO A. S. **PROTÓTIPO DE UM SISTEMA DE AQUECIMENTO DE ÁGUA PARA POPULAÇÃO DE BAIXA RENDA USANDO ENERGIA SOLAR E ELÉTRICA, COM REAPROVEITAMENTO DE CALOR.** Trabalho de conclusão de curso. Instituto Federal de Minas gerais, campus Formiga. Abril,2014.

SILVA, Clodoaldo. **Técnicas de Programação (Máquinas de Estado).** Clube da Eletrônica, Automação e Controle. 11 Junho de 2011. Disponível em <http://www.clubedaeletronica.com.br/Automacao/PDF/Capitulo%20006%20-%20Logica%20ladder%20-%20utilizando%20maquinas.pdf>, Acesso em Junho,2015.

SEMICONDUTOR, National. **LM35/ LM35A/ LM35C/ LM35CA/ LM35D Precision Centigrade Temperature Sensor,** Dezembro, 1994. Disponível em: <http://pdf.datasheetcatalog.com/datasheet/nationalsemiconductor/DS005516.PDF>. Acesso em: Maio, 2015.

SILVA, B.S. da; PEREIRA, M.de C.; BUCHMANN, R.M. **Padrões de Comunicação Serial Clássicos: RS-232, RS-422 e RS-485.** Universidade Federal do Rio de Janeiro – Escola Politécnica. Departamento de Engenharia Eletrônica e de Computação. 2013.

SOUZA, Fábio. **Arduino Uno.** Embarcados. Novembro,2013. Disponível em: <http://www.embarcados.com.br/arduino-uno/. Acesso em: Maio,2015>.

SOUZA, Fábio. **Arduino-Primeiros Passos.** Embarcados. Novembro,2013. Disponível em:< http://www.embarcados.com.br/arduino-primeiros-passos/>. Acesso em: Maio,2015.

TATAI, Victor Kazuo. **Técnicas de Sistemas Inteligentes Aplicadas ao Desenvolvimento de Jogos de Computador.** Dissertação de mestrado. UNICAMP, Faculdade de Engenharia Elétrica e de Computação. Dezembro, 2002.

TENENBAUM, Andrew S. **REDES DE COMPUTADORES.** Tradução da 4ª ed. Editora: ELSEVIER. Rio de Janeiro, 2003 – 13ª reimpressão.

YANO, Thaise. **Uma Abordagem Evolutiva Multiobjetivo Para Geração Automática de Casos de Teste a Partir de Máquinas de Estado.** Instituto de Computação – UNICAMP. Departamento de Doutorado em Ciência da Computação. Campinas, 15 de Setembro de 2011.

Anexo A.

Algoritmo de programação do controlador MESTRE.

```
// mestre
#include <SoftwareSerial.h>
#define EST_ENVIA      0
#define EST_SENSOR1    1
#define EST_RECEBE     2
#define ENVIA          3
#define EST_SOLENOIDE  4
#define EST_BOMBA      5
#define EST_SENSOR2    6
#define EST_SENSOR3    7
int sensor=0;
int escravo=1;
int estado = EST_ENVIA;
int chegou;
int volume[3];
int volume1=0;
int volume2=0;
int volume3=0;
int Temperatura1=0;
int Temperatura2=0;
long previousMillis = 0;        // Variável de controle do tempo
long Intervalo = 1000;     // Tempo em ms do intervalo a ser executado
int tempo=0;
void setup(){
int SENSOR_1;
  Serial.begin(9600);
```

```
  pinMode(2,OUTPUT);
}

void escravo1(){
  if(escravo=1){
  switch(estado){
    case EST_ENVIA:     // Em caso de estado Inicial ENvia
    sensor=0;
      digitalWrite(2, HIGH); // Habilita para escrever no barramento
      Serial.write(0x02);  // Escreve o Endereço de escravo 2
      estado = EST_SENSOR1; // Altera para o Estado Leitura de Sensor
      delay(100);
      break;
    case EST_SENSOR1: // Em caso de Estado Leitura de sensor
      digitalWrite(2, HIGH); // Habilita para escrever no barramento
      Serial.write(0x04);  // Escreve o Endereço do Sensor a ser lido
      estado = EST_RECEBE; //  Altera para o Estado Receber dados
      break;
    case EST_SENSOR2:
      digitalWrite(2, HIGH); // Habilita para escrever no barramento
      Serial.write(0x05);  // Escreve o Endereço do Sensor a ser lido
      estado = EST_RECEBE; //  Altera para o Estado Receber dados

      case EST_SENSOR3:
      digitalWrite(2, HIGH); // Habilita para escrever no barramento
      Serial.write(0x06);  // Escreve o Endereço do Sensor a ser lido
      estado = EST_RECEBE; //  Altera para o Estado Receber dados
```

```
    case EST_RECEBE:   // Em caso de estado Receber dados

    sensor++;

    digitalWrite(2,LOW); // Habilita o barramento para receber informação

    if(Serial.available()){ // em caso de que algo seja escrito na serial

      volume[sensor] = Serial.read(); // É atribuído o valor recebido a variável volume posição sensor;

        if(volume[sensor]=10){  // Se o valor da serial recebido for igual a 9, o  nível está baixo para os sensores de nível, e T1<T2 para posição2 ;

        digitalWrite(2,HIGH);

        Serial.println("Nivel baixo");

        estado = EST_ENVIA;

      }else if(volume[sensor]=11){

        digitalWrite(2,HIGH); // Se o valor da serial recebido for igual a 10, o  nível está médio para os sensores de nível, e T1>T2 para posição2 ;

        Serial.println("Nivel medio");

        estado = EST_ENVIA;

      }else if(volume[sensor]=12){

        digitalWrite(2,HIGH); // Se o valor da serial recebido for igual a 11, o  nível está alto;

        Serial.println("Nivel alto");

        estado = EST_ENVIA;

      }

    }

    if (sensor = 1){

    estado= EST_SENSOR2;

    } else if( sensor = 2){

      estado = EST_SENSOR3;

    } else if (sensor=3){

      estado = EST_ENVIA;

      escravo=2;

    }
```

```
    break;
  }
  }
}

void escravo2(){
  if(escravo=2){
  switch(estado){
    case EST_ENVIA:     // Em caso de estado Inicial Envia
    sensor=0;
    tempo=0;
      digitalWrite(2, HIGH); // Habilita para escrever no barramento
      Serial.write(0x03);  // Escreve o Endereço de escravo 3
      estado = EST_SENSOR1; // Altera para o Estado Leitura de Sensor
      delay(100);
      break;
    case EST_SENSOR1: // Em caso de Estado Leitura de sensor
    tempo=0;
      digitalWrite(2, HIGH); // Habilita para escrever no barramento
      Serial.write(0x07);  // Escreve o Endereço do Sensor a ser lido
      estado = EST_RECEBE; //  Altera para o Estado Receber dados
      break;
    case EST_SENSOR2:
    tempo=0;
      digitalWrite(2, HIGH); // Habilita para escrever no barramento
      Serial.write(0x08);  // Escreve o Endereço do Sensor a ser lido
      estado = EST_RECEBE; //  Altera para o Estado Receber dados
```

```
    case EST_RECEBE:   // Em caso de estado Receber dados

    sensor++;

    tempo=0;

    digitalWrite(2,LOW); // Habilita o barramento para receber informação

    if(Serial.available()){ // em caso de que algo seja escrito na serial

      volume[sensor] = Serial.read(); // É atribuído o valor recebido a variável volume posição
sensor;

        if(volume[sensor]=10){  // Se o valor da serial recebido for igual a 9, o  nível está baixo
para os sensores de nível, e T1<T3 para posição2 ;

        digitalWrite(2,HIGH);

        Serial.println("Nivel baixo");

        estado = EST_ENVIA;

      }else if(volume[sensor]=11){

        digitalWrite(2,HIGH); // Se o valor da serial recebido for igual a 10, o  nível está médio
para os sensores de nível, e T1>T3 para posição2 ;

        Serial.println("Nivel medio");

        estado = EST_ENVIA;

      }else if(volume[sensor]=12){

        digitalWrite(2,HIGH); // Se o valor da serial recebido for igual a 11, o  nível está alto;

        Serial.println("Nivel alto");

        estado = EST_ENVIA;

      }

    }

    if (sensor = 1){

    estado= EST_SENSOR2;

    } else if( sensor = 2){

      sensor=0;

      escravo=1;

    }
```

```
     break;
  }
  }
}

void bomba(){
  if(escravo=1){
  switch(estado){
    case EST_ENVIA:     // Em caso de estado Inicial ENvia
    sensor=0;
    tempo=0;
      digitalWrite(2, HIGH); // Habilita para escrever no barramento
      Serial.write(0x02);  // Escreve o Endereço de escravo 3
      estado = EST_BOMBA; // Altera para o Estado Leitura de Sensor
      delay(100);
      break;
    case EST_BOMBA: // Em caso de Estado Leitura de sensor
    tempo=0;
      digitalWrite(2, HIGH); // Habilita para escrever no barramento
      Serial.write(0x13);  // Escreve o Endereço do Sensor a ser lido
      estado = EST_ENVIA; //  Altera para o Estado Receber dados
     break;
  }
  }
}

void solenoide(){
  if(escravo=2){
```

```
  switch(estado){
    case EST_ENVIA:     // Em caso de estado Inicial ENvia
    sensor=0;
    tempo=0;
      digitalWrite(2, HIGH); // Habilita para escrever no barramento
      Serial.write(0x03);  // Escreve o Endereço de escravo 3
      estado = EST_SOLENOIDE; // Altera para o Estado Leitura de Sensor
      delay(100);
      break;
    case EST_SOLENOIDE: // Em caso de Estado Leitura de sensor
    tempo=0;
      digitalWrite(2, HIGH); // Habilita para escrever no barramento
      Serial.write(0x14);  // Escreve o Endereço do Sensor a ser lido
      estado = EST_ENVIA; //  Altera para o Estado Receber dados
     break;
  }
  }
}
void loop(){
  unsigned long currentMillis = millis();    //Tempo atual em ms

                              //Lógica de verificação do tempo
                              if (currentMillis - previousMillis > Intervalo) {
                                    previousMillis = currentMillis;    // Salva o tempo atual
              tempo++;
}
  if (tempo==5){
   estado=EST_ENVIA;
```

```
 tempo=0;
}

if (escravo=1){
  escravo1();
  volume1=volume[1]; // atribui o valor do reservatório 1
  volume3=volume[3]; // atribui o valor do reservatório 3
  Temperatura1=volume[2]; // atribui o valor da tempertura4

}else if(escravo=2){
  escravo2();
  volume2=volume[1]; // atribui o valor do reservatório 2
  Temperatura2 = volume[2]; // atribui a comparação entre as temperaturas 1 e 3
}
if ((volume3==12 || volume3==11) && (volume1==11 || volume1==10)){
   escravo=1;
   bomba();
    if(volume1=11){
    Serial.println("Nível 1 Medio");
   }else{
     Serial.println("Nível 1 Baixo");
   }
   if(volume3=12){
   Serial.println("Nivel 3 Alto");
   }else {
   Serial.println("Nivel 3 Medio");
   }
```

```
  }
  if (volume1=12){
    Serial.println("Nivel 1 Alto");
  }
  if (volume3 =10){
    Serial.println("Nivel 3 Baixo");
  }
   if ((Temperatura2==11) && (volume2==11 || volume2==10)){
     escravo=2;
     solenoide();
     Serial.println("Temperatura 3 > 1");
     if(volume2=11){
     Serial.println("Nível 2 Medio");
     }else {
     Serial.println("Nível 2 Baixo");
     }
   }
   if (Temperatura2=10){
     Serial.println ("Temperatura 1 > 3");
   }
   if (volume2 =12){
     Serial.println("Nível 2 Alto");
   }
  delay(30000);
}
```

Anexo B.

Algoritmo de programação do Escravo 1

```
// escravo
#include <SoftwareSerial.h>
#define EST_AG_RX    0
#define EST_SENSOR   1
#define EST_RECEBE   2
#define EST_ATUADOR  3
#define EST_SENSOR_1 4
#define EST_SENSOR_2 5
#define EST_SENSOR_3 6
int estado =  EST_AG_RX;
int chegou;
int SENSOR;
int pin =2;
int valor1=0;
int valor=1;
int tempo=0;
long previousMillis = 0;        // Variável de controle do tempo
long Intervalo = 1000;     // Tempo em ms do intervalo a ser executado

void setup(){
  Serial.begin(9600);
  pinMode(pin, OUTPUT);
  pinMode(6, OUTPUT);
}

 void leitura_sensores(){
```

```
switch(estado){
 case EST_AG_RX:
 tempo=0;
 digitalWrite(2,HIGH);
 Serial.println("Aguardando: ");
 delay(10);
 digitalWrite(pin,LOW);
 delay(10);
 if (Serial.available()){
   chegou =Serial.read();
   if (chegou=0x02){
       estado=EST_SENSOR;
   }
 }
   break;
 case EST_SENSOR:
  tempo=0;
 digitalWrite(2,LOW);
 if (Serial.available()){
   SENSOR = Serial.read();
   if (SENSOR=0){
     estado= EST_AG_RX;
   }else if (SENSOR=0X04){
     estado = EST_SENSOR_1;
   }else if(SENSOR =0X05){
     estado = EST_SENSOR_2;
   }else if(SENSOR =0X06){
     estado = EST_SENSOR_3;
```

```
   }else if(SENSOR=0X13){
     estado = EST_ATUADOR;
   }
  }
  break;
  case EST_SENSOR_1:
  tempo=0;
  digitalWrite(pin,HIGH);
  valor = analogRead(0);
  if (valor<171){  // menor que 40% do nível
    Serial.write(10);
    estado =EST_AG_RX;

  }else if (valor>=171 & valor < 343){ // entre 40% e 80%
  Serial.write(11);
  estado = EST_AG_RX;

  }else {
  Serial.write(12);
 estado = EST_AG_RX;
  }
  break;

case EST_SENSOR_2:
tempo=0;
  digitalWrite(pin,HIGH);
  valor = analogRead(1);
  if (valor<171){ // menor do que 40% do nivel
```

```
    Serial.write(10);
    estado = EST_AG_RX;

  }else if (valor>=171 & valor < 343){ // entre 40 e 80% do nível
  Serial.write(11);
 estado = EST_AG_RX;

  }else {
  Serial.write(12);
 estado = EST_AG_RX;
  }
  break;
case EST_SENSOR_3:
tempo=0;
  digitalWrite(pin,HIGH);
  valor = analogRead(3);
  valor1=analogRead(4);
  if (valor<valor1){
    Serial.write(10);
    estado = EST_AG_RX;

  }else if (valor>valor1){
  Serial.write(11);
 estado = EST_AG_RX;

  }
  break;
  case EST_ATUADOR:
```

```
    tempo=0;
    digitalWrite(pin,HIGH);
    Serial.write("0");
    digitalWrite(pin,LOW);
    valor=analogRead(0);
    while (valor < 400){
    valor=analogRead(0);
    digitalWrite(6,HIGH);
   }
    digitalWrite(6,LOW);
    estado=EST_AG_RX;
   }
 }
void loop(){
  unsigned long currentMillis = millis();    //Tempo atual em ms

                                //Lógica de verificação do tempo
if (currentMillis - previousMillis > Intervalo) {
previousMillis = currentMillis;    // Salva o tempo atual
tempo++;
}
  if (tempo==5){
   estado=EST_AG_RX;
   tempo=0;
  }
  leitura_sensores();
  estado =  EST_AG_RX;
}
```

Anexo C.

Endereço dos Componentes da planta.

Componente	Endereço (Decimal)	N° de bytes
Controlador Mestre	1	1
Componentes Escravo 1		
Controlador Escravo 1	2	1
Sensor Volume 1	4	1
Sensor volume 3	5	1
Comparador de Temperatura 1 (T1 eT3)	6	1
Componentes Escravo 2		
Controlador Escravo 1	3	1
Comparador de Temperatura 1 (T2 eT4)	7	1
Válvula	13	1
Bomba	14	1