UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ – UTFPR
DEPARTAMENTO ACADÊMICO DE ELETROTÉCNICA – DAELT
CURSO SUPERIOR DE TECNOLOGIA EM AUTOMAÇÃO INDUSTRIAL

ARTHUR GUILHERME LANG
AUGUSTO FELLIPE ALVES DUTRA
PAULO HENRIQUE WAROMBY

DESENVOLVIMENTO DE UM IRRIGADOR MICROCONTROLADO INTEGRADO A UM SISTEMA DE SUPERVISÃO E CONTROLE

PROJETO DE PESQUISA

CURITIBA
2013

ARTHUR GUILHERME LANG
AUGUSTO FELLIPE ALVES DUTRA
PAULO HENRIQUE WAROMBY

DESENVOLVIMENTO DE UM IRRIGADOR MICROCONTROLADO INTEGRADO A UM SISTEMA DE SUPERVISÃO E CONTROLE

Projeto apresentado como pré-requisito para o Trabalho de Conclusão de Curso de graduação, do Curso Superior de Tecnologia em Automação Industrial do Departamento Acadêmico de Eletrotécnica – DAELT – da Universidade Tecnológica Federal do Paraná – UTFPR.

Orientadora: Professora Rosângela Winter, M.Sc.

CURITIBA
2013

ARTHUR GUILHERME LANG
AUGUSTO FELLIPE ALVES DUTRA
PAULO HENRIQUE WAROMBY

DESENVOLVIMENTO DE UM IRRIGADOR MICROCONTROLADO INTEGRADO A UM SISTEMA DE SUPERVISÃO E CONTROLE

Este Trabalho de Diplomação foi julgado e aprovado como requisito parcial para a obtenção do Título de **Tecnólogo em Automação Industrial**, do **Curso Superior de Tecnologia em Automação Industrial,** da **Universidade Tecnológica Federal do Paraná**.

Curitiba, 10 de dezembro de 2013

_____
Prof. José da Silva Maia, M.Eng.
Coordenador de Curso
Departamento Acadêmico de Eletrotécnica

_____
Prof. Rafael Fontes Souto, M.Sc.
Responsável pelo Trabalho de Diplomação da Tecnologia
Departamento Acadêmico de Eletrotécnica

BANCA EXAMINADORA

_____
Prof. Juvenal Akita, Esp.
Universidade Tecnológica Federal do Paraná

_____
Profª. Rosangela Winter, M.Sc.
Universidade Tecnológica Federal do Paraná
Orientadora

_____
Prof. Marcio Aparecido Batista, M.Sc.
Universidade Tecnológica Federal do Paraná

_____
Profª Rosana Mayer, M.Sc.
Universidade Tecnológica Federal do Paraná

A Folha de Aprovação assinada encontra-se na Coordenação do Curso.

RESUMO

LANG, Arthur Guilherme; DUTRA, Augusto Fellipe Alves; WAROMBY, Paulo Henrique. **Desenvolvimento de um irrigador microcontrolado integrado a um sistema de supervisão e controle.** 76f. – Tecnologia em Automação Industrial, UTFPR – Universidade Tecnológica Federal do Paraná.

Em tempos de expansão tecnológica no setor da agricultura, existe a necessidade do desenvolvimento de sistemas automatizados voltados a pequenos agricultores. Esses sistemas devem ter como princípio a garantia da qualidade do produto cultivado, de maneira mais simples e eficiente, além de apresentar um baixo custo de implementação, mantendo a competitividade de mercado do produtor. O presente projeto tem como objetivo apresentar uma solução automatizada para o controle de um sistema de irrigação em pequenas lavouras. O resultado do projeto é o desenvolvimento de um irrigador microcontrolado, operado através de um sistema de supervisão e controle que pode ser acessado a partir de dispositivos dotados de tecnologia *wireless*. O sistema leva em conta a temperatura e umidade relativa do ar para calcular o volume a ser irrigado, além de realizar o controle do nível do reservatório de forma automática.

Palavras-chave: irrigação, sistema supervisório, microcontrolador, automação.

ABSTRACT

LANG, Arthur Guilherme; DUTRA, Augusto Fellipe Alves; WAROMBY, Paulo Henrique. **Development of a micro-controlled irrigator linked to a system of supervision and control.** 76p. - Industrial Automation Technology, UTFPR – Federal University of Technology – Paraná, Brazil.

At a time in which the agricultural sector is witnessing technological expansion, automated systems directed towards small-scale producers need to be developed. These systems should aim at guaranteeing the quality of the crop in the simplest and most efficient manner, as well as maintaining low-cost implementation, in order to maintain the producer's competitivity on the market. This project aims to present an automated solution for the control of an irrigation system on small farms. The result of the project will be the development of a micro controlled sprinkler operated through a system of supervision and control that can be accessed from devices equipped with wireless technology. The system takes into account the temperature and humidity to calculate the volume to be irrigated, as well as controlling the reservoir level automatically.

Key words: irrigation, system of supervision, microcontroller, automation.

LISTA DE FIGURAS

Figura 01 Modelo de sistema de irrigação por gotejamento..... 17
Figura 02 Plataforma Arduíno UNO..... 19
Figura 03 Esquema elétrico Arduíno UNO..... 21
Figura 04 Quadro de mensagens Modbus..... 27
Figura 05 Diagrama de blocos do protótipo..... 30
Figura 06 Bancada do protótipo..... 31
Figura 07 Reservatório de água..... 31
Figura 08 Bomba acionada através de corrente alternada..... 32
Figura 09 Válvula solenoide..... 33
Figura 10 Sensor de vazão..... 33
Figura 11 Sensor ultrassônico..... 34
Figura 12 Sensor de temperatura e umidade relativa do ar..... 35
Figura 13 Placa de *interface*..... 36
Figura 14 Plataforma Arduíno UNO instalada..... 37
Figura 15 Aspersor..... 37
Figura 16 Protótipo finalizado..... 38
Figura 17 Esquema elétrico..... 39
Figura 18 Fluxograma do código de programação..... 40
Figura 19 *Interface* humano-máquina..... 43

LISTA DE ABREVIATURAS, SIGLAS E ACRÔNIMOS

A – *Ampère* (unidade de medida de intensidade de corrente elétrica).

ASCII – *American Standart Code for Information Interchange* (em português, Código Padrão Americano para o Intercâmbio de informação).

BIT – *Binary Digit* (em português, Dígito Binário).

CLP – Controlador Lógico Programável.

COM – *Communication* (em português, Comunicação. Refere-se as portas de comunicação serial COM1 e COM2 do computador).

HTTP – *HyperText Transfer Protocol* (em português, Protocolo de Transferência de Hipertexto).

Hz – *Hertz* (unidade de medida de frequência).

IDE – *Integrated Development Environment* (em português, Ambiente de Desenvolvimento Integrado).

IHM – Interface Humano-Máquina.

IP – *Internet Protocol* (em português, Protocolo de *Internet*), padrão de endereçamento, pelo qual um computador é indentificado na *internet*.

JDK – Java *Development Kit* (em português, Kit de Desenvolvimento Java).

JRE – Java *Runtime Environment* (em português, Ambiente de Tempo de Execução Java).

kB – *Kilobyte* (conjunto de 1024 *bytes*).

LED – *Light Emitting Diode* (em português, Diodo Emissor de Luz).

l – Litros.

l/min – Litros por minuto.

mA – Miliampère (unidade de medida de corrente elétrica).

MAC – Macintosh.

MHz – *Megahertz* (unidade de medida de frequência).

m – Metro, unidade de medida de comprimento.

ml – Mililitro, milésima parte do litro.

mm – Milímetro, milésima parte do metro.

ms – Milissegundo, unidade de tempo equivalente à milésima parte do segundo.

RTU – *Remote Terminal Unit* (em português, Unidade Terminal Remota).

SCADA – *Supervisory Control And Data Acquisition* (em português, Controle Supervisório e Aquisição de Dados).

USB – *Universal Serial Bus* (em português, Barramento Serial Universal).

TCC – Trabalho de Conclusão de Curso.

V – *Volt* (unidade de medida de tensão elétrica).

W – *Watt* (unidade de medida de potência).

SUMÁRIO

1 INTRODUÇÃO..... 8
1.1 TEMA..... 9
1.1.1 Delimitação do tema..... 10
1.2 PROBLEMAS E PREMISSAS..... 10
1.3 OBJETIVOS..... 11
1.3.1 Objetivo geral..... 11
1.3.2 Objetivos específicos..... 11
1.4 JUSTIFICATIVA..... 12
1.5 PROCEDIMENTOS METODOLÓGICOS..... 13
2 REFERENCIAL TEÓRICO 14
2.1 IRRIGAÇÃO..... 14
2.2 ARDUÍNO..... 18
2.2.1 Especificações técnicas..... 20
2.2.2 IDE do Arduíno..... 23
2.3 SCADABR..... 24
2.4 MODBUS..... 25
2.4.1 Modbus RTU..... 26
2.4.1.1 Enquadramento de mensagens Modbus RTU..... 26
3 ESTRUTURAÇÃO DO PROTÓTIPO..... 28
3.1 DESCRIÇÃO DO PROTÓTIPO..... 28
3.2 DESENVOLVIMENTO DO PROTÓTIPO..... 29
3.3 ESQUEMA ELÉTRICO..... 39
3.4 CÓDIGO DE PROGRAMAÇÃO..... 40
3.5 SISTEMA DE SUPERVISÃO E CONTROLE..... 41
3.5.1 Operação do sistema de supervisão e controle..... 43
3.6 TESTES E RESULTADOS..... 44
4 CONSIDERAÇÕES FINAIS..... 46
5 CONCLUSÃO..... 49
REFERÊNCIAS..... 51
GLOSSÁRIO..... 54
APÊNDICES..... 56
ANEXOS..... 73

1 INTRODUÇÃO

Pode-se definir irrigação como a aplicação artificial de água ao solo, com a finalidade de complementar as chuvas, resultando em melhor produtividade e qualidade do produto cultivado.

Conforme relatam Lima, Ferreira e Christofidis (2013), o aumento do custo da terra, aliado ao considerável capital necessário à exploração agrícola, não permite mais que a produção final dependa da ocorrência ou não de um regime de precipitação adequado. Assim sendo, a nova tendência do meio empresarial agrícola tem sido a de aumento do interesse pela prática da irrigação, que, além de reduzir riscos, proporciona outras vantagens significativas ao produtor irrigante.

Com o passar dos anos, percebeu-se a necessidade da utilização de tecnologias nos sistemas de irrigação artificiais, acompanhando as novas técnicas de cultivo e a busca por melhores resultados.

“A automação se faz necessária não somente pela possibilidade de diminuição dos custos com mão de obra, mas principalmente por necessidades operacionais.” (SUZUKI; HERNANDEZ, 2012).

No Brasil, a irrigação automatizada segue um constante crescimento, pois se observa excelentes resultados com relação ao aumento da produtividade e redução de desperdício de água.

> Sistemas automáticos de controle de irrigação se tornaram uma ferramenta essencial para a aplicação de água na quantidade necessária e no devido tempo, contribuindo para a manutenção da produção agrícola e, também, para a utilização eficiente de recursos hídricos (TESTEZLAF(a), 2011).

Segundo Andrade (2001), o processo de seleção do sistema de irrigação requer a análise detalhada das condições apresentadas, em função das exigências de cada sistema, de forma a permitir a identificação das melhores alternativas.

O incentivo à pratica da irrigação automatizada tem sido adotado pelos governos federal, estadual e municipal de todo o país, porém este incentivo ainda mostra-se tímido diante do atual cenário agrícola brasileiro.

> Há muitos anos o agricultor sofre com problemas de estiagem, que trazem prejuízos em alguns períodos. Mas nos últimos dias o agricultor teve um bom motivo para comemorar, pois foi contemplado com um projeto que busca minimizar este problema. O agricultor é o primeiro beneficiado do Projeto de Incentivo à Implantação de Sistemas Irrigados (Proisi) da Secretaria de Obras, Públicas, Irrigação e Desenvolvimento Urbano (SOP). O empreendimento, escolhido para implantar o equipamento - Luiz Roberto Muntini e Cia Ltda. -, foi selecionado por pregão na semana passada. O valor do investimento é de quase R$ 9 mil (AGRODIÁRIO, 2013).

1.1 TEMA

Entre os pequenos agricultores existe um paradigma de que a irrigação é algo para grandes áreas de cultivo, com alto custo e fora de suas realidades. Baseado nisso, encontra-se uma necessidade de preencher essa lacuna, levando soluções de irrigação viáveis para pequenas propriedades de cultivo em geral.

> Agricultores familiares de Salto do Lontra, região Sudoeste do estado do Paraná, estão investindo na irrigação como alternativa para melhorar os resultados técnicos e econômicos de seus empreendimentos e diminuir os riscos de perdas na produção em consequência das estiagens. Hoje a tecnologia está presente em 117 propriedades rurais, cerca de 10% do total de unidades produtivas familiares existentes no município (AGROSOFT BRASIL, 2012).

A irrigação controlada em pequenas lavouras, além de trazer a diminuição nas perdas, traz também a possibilidade da diversificação dos produtos cultivados, aumentando a expectativa de renda para esses produtores que, muitas vezes, têm a família toda trabalhando na plantação, sendo essa a única fonte de renda familiar.

> Abre para as famílias, também, a possibilidade de investir em alternativas de produção, como a fruticultura ou a olericultura. Temos pessoas que nunca tinham plantado melancia, melão e, atualmente, exploram essas culturas como um novo negócio dentro da propriedade graças a irrigação, gerando renda e mais qualidade de vida (AGROSOFT BRASIL, 2012).

A irrigação pode auxiliar os pequenos produtores a aumentar a produtividade de suas plantações, podendo mantê-los menos vulneráveis às variações climáticas. Traz também uma segurança para investimentos em novas culturas.

“Projetos de irrigação de pequena escala podem gerar diversos benefícios, particularmente em termos de eficiência, baixos custos de participação e mais influência sobre a gestão dos recursos hídricos.” (DILLON, 2011).

1.1.1 Delimitação do Tema

O projeto visa apresentar um protótipo de pequeno porte de um sistema de irrigação microcontrolado, comandado por meio de dispositivos móveis que possuam a tecnologia *wireless*, como *notebooks, tablets* e *smartphones*, utilizando um sistema de supervisão e controle, focando em uma solução confiável, flexível e acessível aos pequenos produtores.

O escopo do projeto é apresentar informações técnicas relacionadas a confecção do protótipo, apresentando as tecnologias utilizadas e integradas.

Não faz parte do objetivo a implementação em campo do sistema de irrigação microcontrolado, bem como o estudo dos resultados oriundos da irrigação fornecida pelo protótipo. O foco é demonstrar que integrando tecnologias é possível obter soluções eficientes, confiáveis e de baixo custo.

1.2 PROBLEMAS E PREMISSAS

Atualmente, os processos de irrigação de pequeno porte utilizam sistemas com pouca ou nenhuma automação. Sistemas basicamente manuais geram grande risco de perdas na produção, devido ao baixo nível de controle e monitoramento.

> Inserir os pequenos produtores dentro de um projeto de agricultura sustentável é um dos desafios da Empresa Brasileira de Pesquisa Agropecuária (Embrapa). Na avaliação do presidente da Embrapa, Pedro Arraes, a combinação entre cooperativismo e ação mista de agentes públicos e privados na disseminação de tecnologias pode viabilizar o projeto de agricultura sustentável nas pequenas propriedades (GONÇALVES, 2012).

Pequenos produtores não possuem grandes áreas de plantio que lhes proporcionem a possibilidade de investir em sistemas automáticos de irrigação, que possuam interfaces e periféricos mais elaborados. Desta forma, é imprescindível que o desenvolvimento de sistemas voltados a este público leve em consideração o baixo custo de implementação.

Diante deste cenário, surge o seguinte questionamento: **como reduzir as perdas na irrigação e produção de pequenos produtores que praticam agricultura sustentável, utilizando sistemas de irrigação automatizados e de baixo custo?**

Utilizando a integração de soluções tecnológicas é possível agregar flexibilidade, facilidade de operação e informatização aos sistemas de irrigação, características geralmente encontradas em sistemas inacessíveis a este público.

1.3 OBJETIVOS

1.3.1 Objetivo Geral

Elaborar um protótipo de um irrigador microcontrolado integrado a um sistema de supervisão e controle, operado através de dispositivos móveis dotados de tecnologia *wireless.*

1.3.2 Objetivos Específicos

- Pesquisar as tecnologias que foram utilizadas no protótipo;
- Estudar e definir qual plataforma de prototipagem eletrônica será utilizada;
- Programar e simular o *firmware*, bem como os recursos de comunicação do *hardware*;
- Estudar os sistemas supervisórios “*softwares* livres” disponíveis, utilizando a ferramenta que melhor se adapte aos princípios do projeto;
- Definir uma interface de comunicação entre e o controlador e o sistema de supervisão e controle;
- Definir os dispositivos móveis para comandar o sistema;
- Integrar as diversas partes do sistema;
- Simular os circuitos;
- Construir o protótipo e realizar os testes.

1.4 JUSTIFICATIVA

O investimento de grandes produtores agrícolas em tecnologia vem apresentando um ótimo resultado na diminuição de perdas, melhoria na qualidade dos produtos colhidos, além de possibilitar a colheita da entressafra. Como consequência, percebe-se um aumento relativo de lucros por área plantada. No entanto, alguns produtores ainda se deparam com fatores que operam contra a implementação de um sistema de irrigação. Um problema muito frequente, especialmente para aqueles que possuem propriedades de cultivo com uma área menor, é a exigência de um investimento proporcionalmente muito alto em relação a sua receita, somando-se a falta de mão de obra que se torna um agravante deste problema.

Almeja-se, portanto, propor uma solução que viabiliza a difusão de um sistema de automação nesse nicho de mercado, disponibilizando a inserção tecnológica no campo e auxiliando esses pequenos produtores.

Trazer novas tecnologias, até então não comumente utilizadas, permite que a automação auxilie no aumento da produtividade e renda de pequenos produtores.

> A necessidade da busca da otimização dos recursos produtivos, da competividade no mercado produtivo, da necessidade de aumento de produtividade e redução de custos, levam a uma tendência de adoção de tecnologias capazes de tornar a exploração cada vez mais competitiva e rentável. A automação se faz necessária não somente pela possibilidade de diminuição dos custos com mão de obra, mas principalmente por necessidades operacionais, tais como irrigação de grandes área no período noturno (SUZUKI; HERNANDEZ, 2012).

Para tanto, a proposta atingirá concretude através de um protótipo que demonstre a viabilidade financeira de um sistema de irrigação automatizado, de pequeno porte, aliado a procedimentos simples de operação que gerem empatia entre o operador e o sistema em questão.

1.5 PROCEDIMENTOS METODOLÓGICOS

Para a construção do protótipo foi necessário o estudo específico das tecnologias a serem integradas no projeto.

A pesquisa iniciou-se com a busca por sensores de custo acessível que atendam as especificações do projeto. Posteriormente, terá início a busca por informações referentes à utilização de plataformas de prototipagem eletrônicas microcontroladas, com a sua programação utilizando linguagem C. Dentro deste estudo, estão também envolvidos os tópicos a seguir:

- Protocolo de comunicação *Modbus*, através de conexão USB;
- Interface dos periféricos com a plataforma;
- Integração dos dispositivos móveis;
- Integração com sistema de supervisão e controle;
- Acionamento de máquinas de corrente alternada.

Houve ainda a necessidade de realizar a programação do sistema de supervisão e controle, a fim de fazer a leitura e escrita de valores no microcontrolador.

Com as tecnologias testadas independentemente, foi realizada a integração das mesmas com a elaboração efetiva do protótipo. Com esta integração, serão necessários ajustes e testes para que os objetivos venham a ser alcançados com sucesso.

2 REFERENCIAL TEÓRICO

Este capítulo elencará e apresentará as tecnologias envolvidas no projeto. Os conteúdos aqui apresentados facilitarão o entendimento do capítulo seguinte, relativo à estruturação do protótipo.

2.1 IRRIGAÇÃO

A irrigação é um método artificial de utilização da água na agricultura, tendo como principal objetivo, o controle da quantidade de água aplicada na área de cultivo, visando à diminuição de perdas na hora da colheita.

> Na visão inicial, a irrigação era vista somente como aplicação de água e tinha como objetivo principal, a luta contra a seca e, ou, a criação de condições de subsistência para os produtores. No novo conceito, a irrigação evoluiu de simples aplicação de água na agricultura para um importante instrumento no aumento da produção, produtividade e rentabilidade, diminuição dos risos de investimento. (MONTOVANI; BERNARDO; PALARETTI, 2009).

Segundo Silva e Folegatti (2007), a irrigação foi uma das primeiras mudanças que o homem implementou no ambiente, visto que seu manejo era fundamental para o estabelecimento de tribos. Portanto, os sistemas de irrigação tiveram como ponto de partida as antigas civilizações, tais como o povo egípcio que se estabeleceu nas proximidades do rio Nilo, que por sua vez, era uma rica fonte de água, e o povo chinês, em torno do rio Amarelo na China. Neste padrão os povos começaram a se instalar nas proximidades dos rios, utilizando-os como fonte de água para suas plantações, que eram irrigadas por intermédio de canais abertos pelos agricultores, ainda de forma rudimentar. Embora rudimentares algumas técnicas, há lugares que utilizam as mesmas construções até hoje, como o Sri Lanka e o Japão, em relação aos tanques de irrigação e barragens de terras construídas para irrigar arroz.

Para os mencionados autores, já nas Américas, a técnica foi desenvolvida há mais de dois mil anos pelas civilizações Maias, Incas e Astecas. Ainda hoje é possível encontrar vestígios dos sistemas de irrigações deixados por essas tribos no Peru, Chile, Argentina e México. Nos EUA a irrigação já era desenvolvida por indígenas.

Conforme Silva e Folegatti (2007), no Brasil a irrigação passou a ser utilizada inicialmente nas culturas de arroz do Rio Grande do Sul, e também na região central

do país. A análise histórica do desenvolvimento da irrigação junto à história dos povos que a utilizavam é fundamental, visto que a partir da utilização da irrigação conseguiram obter alimentos de modo estável e um consequente aumento populacional.

> O crescimento da população humana tem exigido maiores investimentos para aumentar a produção de alimentos e de outros produtos agrícolas, seja pela incorporação de novas áreas ao sistema produtivo, seja pelo aumento da produtividade. A limitada disponibilidade de novas fronteiras agrícolas e, principalmente, os impactos ambientais provocados pela ação do homem no último século, com mudanças acentuadas no clima do planeta, têm demandado ações concretas e eficazes que visam um aumento da eficiência e da sustentabilidade da agricultura. (PEREIRA, 2011)

Segundo Testezlaf(b), Matsura e Cardoso (2002), com o passar dos anos e com o surgimento de novas civilizações, os sistemas de irrigação sofreram algumas evoluções, e por consequência um aumento progressivo na aplicação dessa técnica, que hoje corresponde a cerca de 18% da área cultivada no mundo, oferecendo 40% de produto como alimentos e fibras, o que comprova a efetividade da utilização deste método.

Para Mantovani, Bernardo e Palaretti (2009), a elaboração de um projeto de irrigação leva em consideração uma série de parâmetros e critérios agronômicos, de engenharia de irrigação e também um cálculo hidráulico. Alguns critérios também relevantes incluem o clima, o solo, topografia, viabilidade econômica, sustentabilidade, distribuição de energia, equipamento escolhido, entre outros.

> Nos projetos de irrigação é imprescindível que sejam realizados, ainda na fase de planejamento, estudos coordenados e concomitantes relacionados aos aspectos ambientais, econômicos e técnicos, para que as soluções e alternativas adotadas efetivamente tenha em si incorporadas medidas de redução dos impactos negativos sobre o meio ambiente (MONTOVANI; BERNARDO; PALARETTI, 2009).

Conforme Testezlaf(c) (2011), os principais sistemas de irrigação, os quais podem ser utilizados com diferentes métodos, são:

.

- IRRIGAÇÃO POR INUNDAÇÃO: a água é aplicada sobre toda a área de cultivo e se acumula na superfície do solo. Nesse caso, além da água se infiltrar durante a sua movimentação na área, ela pode permanecer acumulada ou represada na superfície de forma permanente, no caso da cultura do arroz, ou de forma temporária, no caso de outras culturas;

- IRRIGAÇÃO POR SULCOS: a água é aplicada na área a ser irrigada pela inundação parcial da mesma, acompanhando as linhas da cultura, e escoando por sulcos ou pequenos canais construídos na superfície do solo. Nesse caso, a água se infiltra durante a sua movimentação na área e também no tempo em que permanecer acumulada na superfície do solo após atingir o final do sulco. O melhor exemplo de cultura que utiliza esse sistema no Brasil, principalmente no estado de São Paulo é a do tomate de mesa. Entretanto, culturas anuais e permanentes como algodão e citros, respectivamente, podem também ser irrigadas por esse sistema;

- ASPERSÃO CONVENCIONAL: É um dos que apresentam menor custo, entretanto demanda uma mão de obra maior. Esse sistema é muito utilizado no Brasil nas lavouras de café. Consiste em lançar jatos de água que caem em forma de chuva sobre a cultura, o que gera uma desvantagem em relação ao elevado consumo de água, pois grande parte dessa água é evaporada, visto que a irrigação não é precisa como em outros sistemas. Existem sistemas que são móveis e fixos, e sistemas que são fixos, os automatizados. Fala-se também em “aspersão em malha”, que é uma técnica na qual ficam móveis os aspersores e ficam fixas as linhas principais, de derivação e as laterais;

- GOTEJAMENTO - A água é aplicada com baixa intensidade e grande frequência na região da raiz da planta, sendo levada através de tubos até ser aplicada por emissores. É utilizada em culturas perenes (que são as culturas que ao final de um ciclo produtivo, não há a necessidade de replantio) e em fruticulturas. Possui um elevado custo, mas sua efetividade está na média de 90%. Na Figura 01 se ilustra o modelo de um sistema de irrigação por gotejamento, com a linha principal e suas derivações laterais, destacando o gotejador;

> Este sistema aplica água em apenas parte da área, reduzindo, assim, a superfície do solo que fica molhada, exposta às perdas por evaporação. Com isso, a eficiência de aplicação é bem maior e o consumo de água menor. A irrigação localizada é usada, em geral, sob a forma de sistema fixo, ou seja, o sistema é constituído de tantas linhas laterais quantas forem necessárias para suprir toda a área, isto é, não há movimentação das linhas laterais. Porém, somente determinado número de linhas laterais deve funcionar por vez, a fim de minimizar a capacidade da cabeça de controle (BERNARDO, 2002).

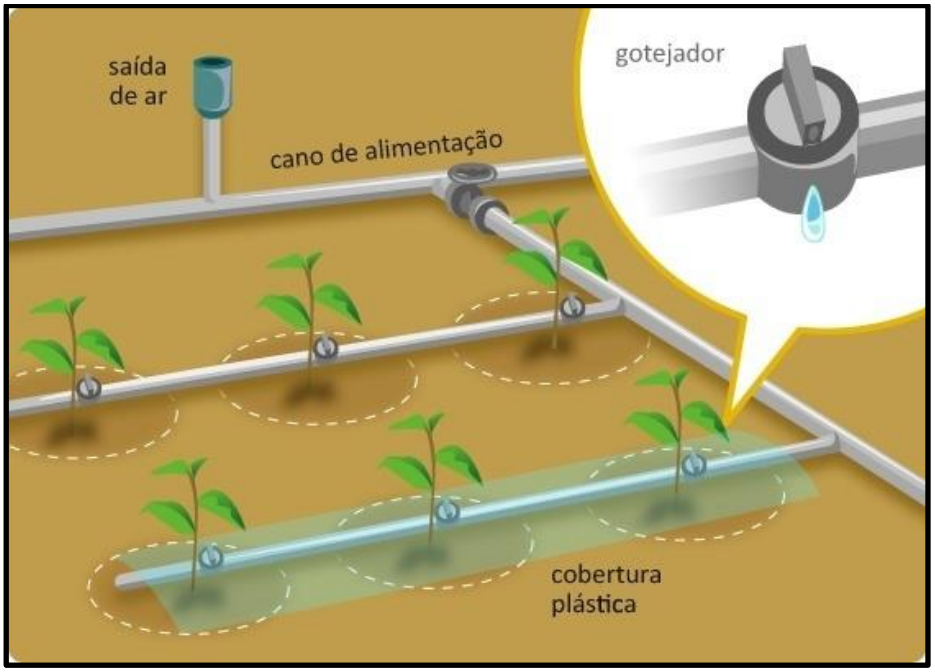

Figura 01: Modelo de um sistema de irrigação por gotejamento.
Fonte: Ciminos (2013).

- MICROASPERSÃO - É considerada uma irrigação localizada, pois a vazão de seus microaspersores é maior que a dos gotejadores. Possui efetividade maior que a aspersão convencional, é muito utilizada em culturas perenes;

- PIVÔ CENTRAL - São instalados aspersores em uma tubulação metálica, que recebe água proveniente de um dispositivo central. Apoia-se em torres metálicas triangulares. Geralmente é utilizado para irrigar áreas de 50 a 130 ha. Além de utilizado para a irrigação, o dispositivo pode ser utilizado para aplicação de fertilizantes;

- CANHÃO HIDRÁULICO - Trata-se de um aspersor de grande porte que é utilizado manualmente, não é indicado para áreas que sofram com ventanias, visto que a eficiência do canhão pode ser prejudicada pelo vento.

Segundo Rodrigues e Irias (2004), apesar do aprimoramento das técnicas, do desenvolvimento de tecnologias e de milhares de pesquisas nesta área, a irrigação traz alguns danos ambientais que ainda não foram controlados. Pode-se citar como principal deles o problema com o consumo de grande volume de água e por

conseguinte a limitação de recursos hídricos em muitas regiões, além da salinização do solo, problemas de saúde pública com o aumento do número de agentes transmissores de doenças, tais como caramujos e mosquitos.

Conforme Mantovani, Bernardo e Palaretti (2009), a irrigação possui uma série de impactos positivos, dentre eles: oferta de alimentos, geração de empregos e consequente fixação do homem no campo, diminuição do êxodo rural e produção controlada de produtos essenciais à vida humana, portanto, a partir de um uso racional e de desenvolvimento de novas técnicas, acredita-se que é possível conciliar os benefícios e malefícios da irrigação.

2.2 ARDUÍNO UNO

Pode-se definir o Arduíno UNO como uma plataforma de prototipagem eletrônica, dotada de um microcontrolador, com suporte a entradas e saídas embutido.

A utilização da plataforma Arduíno UNO foi definida devido a sua facilidade de integração com os demais periféricos, ao número de entradas e saídas correspondentes a necessidade e a possibilidade de integração com um sistema de supervisão e controle.

Para Souza (2009), pode-se definir microcontrolador como um “pequeno” componente eletrônico, dotado de uma “inteligência” programável, utilizado no controle de processos lógicos. Toda a lógica de operação é estruturada em forma de programa e gravada no microcontrolador, sendo executada toda vez que o componente é alimentado.

> Em termos práticos, um Arduíno é um pequeno computador que você pode programar para processar entradas e saídas entre o dispositivo e os componentes externos conectados a ele. O Arduíno é o que chamamos de plataforma de computação física ou embarcada, ou seja, um sistema que pode interagir com seu ambiente por meio de *hardware* e *software* (MCROBERTS, 2011).

De forma simples, pode-se conectar dispositivos de entrada e saída, programando a plataforma de forma a realizar a atividade desejada.

Ensina McRoberts (2011) que a plataforma Arduíno pode ser utilizado para desenvolver objetos interativos independentes, ou pode ser conectado a um computador, a uma rede, ou até mesmo à *internet*, com a finalidade de recuperar e enviar dados, atuando sobre eles. Em outras palavras, ele pode enviar um conjunto

de dados recebidos de alguns sensores para um *site*, dados estes que poderão ser exibidos na forma de gráfico.

A plataforma Arduíno pode ser conectada a diversos tipos de periféricos, como *displays*, botões, sensores, módulos *Ethernet*, entre outros. Qualquer dispositivo que emita dados ou possa ser controlado pode ser utilizado.

A plataforma Arduíno possui diferentes versões, sendo que a versão utilizada no protótipo é denominada Arduíno UNO. A versão é composta de um microcontrolador Amtel AVR de vinte e oito pinos, um cristal oscilador e um regulador linear de 5 V. A placa possui conectores correspondentes às entradas e saídas do microcontrolador, permitindo a conexão dos periféricos. Possui uma porta USB, com um *chip* programado como conversor USB para serial, o que permite conectá-lo a um computador para envio e recebimento de dados.

A figura 02 ilustra a vista superior da plataforma Arduíno UNO, onde se podem observar os conectores de alimentação (USB ou fonte de alimentação externa) no lado esquerdo da figura, o microcontrolador no canto inferior direito, os pinos de conexão na parte inferior e superior da imagem, bem como os demais itens da plataforma.

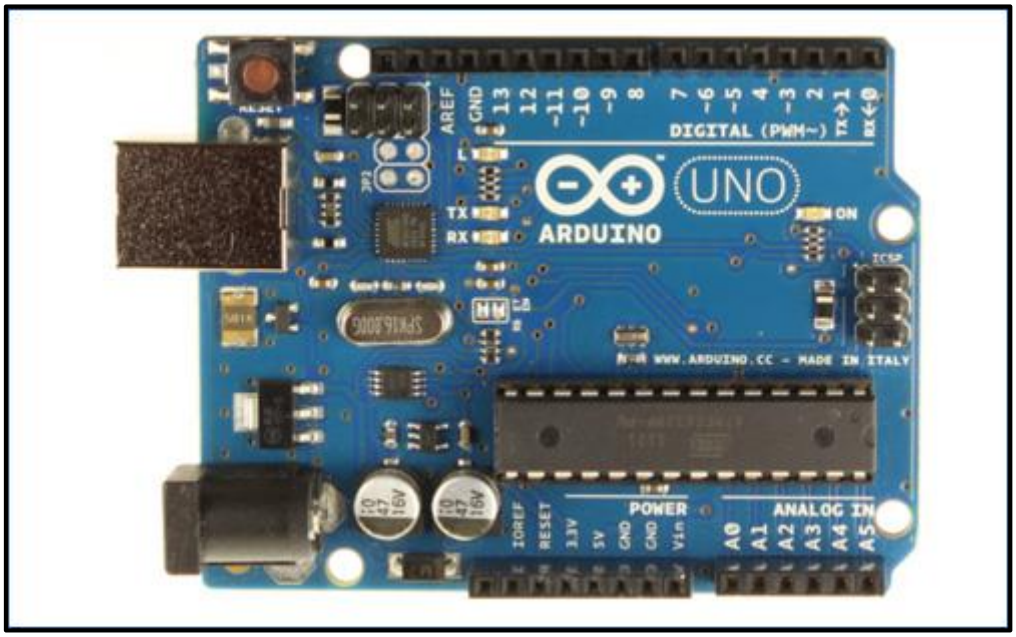

Figura 02: Plataforma Arduíno UNO
Fonte: Arduíno(d) (2013).

Conforme McRoberts (2011), o *firmware* da plataforma é desenvolvido no IDE do Arduíno, um *software* livre onde se realiza a programação das instruções que

serão interpretadas e executadas pelo dispositivo, sendo que o próprio *software* realiza o *upload* do código desenvolvido para a plataforma. A linguagem de programação adotada pelo Arduíno é baseada em C e C++.

Ainda segundo McRoberts (2011), tanto o IDE do Arduíno (*software* para programação e *upload*) quanto o *hardware* da plataforma possuem código aberto, ou seja, qualquer pessoa pode ter acesso aos esquemas eletrônicos e códigos de programação. Com isso, qualquer usuário pode realizar alterações ou montar seu próprio Arduíno, desde que não utilize a reprodução do nome, pois esse possui registro em referência a placa original.

Considerando as características mencionadas acima sobre a plataforma Arduíno, bem como o fato notório de que a mesma possui ampla documentação fornecida pelo fabricante e na *internet*, justifica-se a escolha da mesma.

2.2.1 Especificações técnicas

Na figura 03 se ilustra o esquema elétrico da plataforma Arduíno UNO, onde se observa no lado esquerdo da mesma os itens relacionados à alimentação do equipamento. No canto superior direito se apresenta o microcontrolador e os pinos de E/S. No canto inferior direito é exibido o dispositivo responsável por disponibilizar a comunicação USB entre a plataforma e o computador, que cria uma porta COM virtual para estabelecer a comunicação serial de dados.

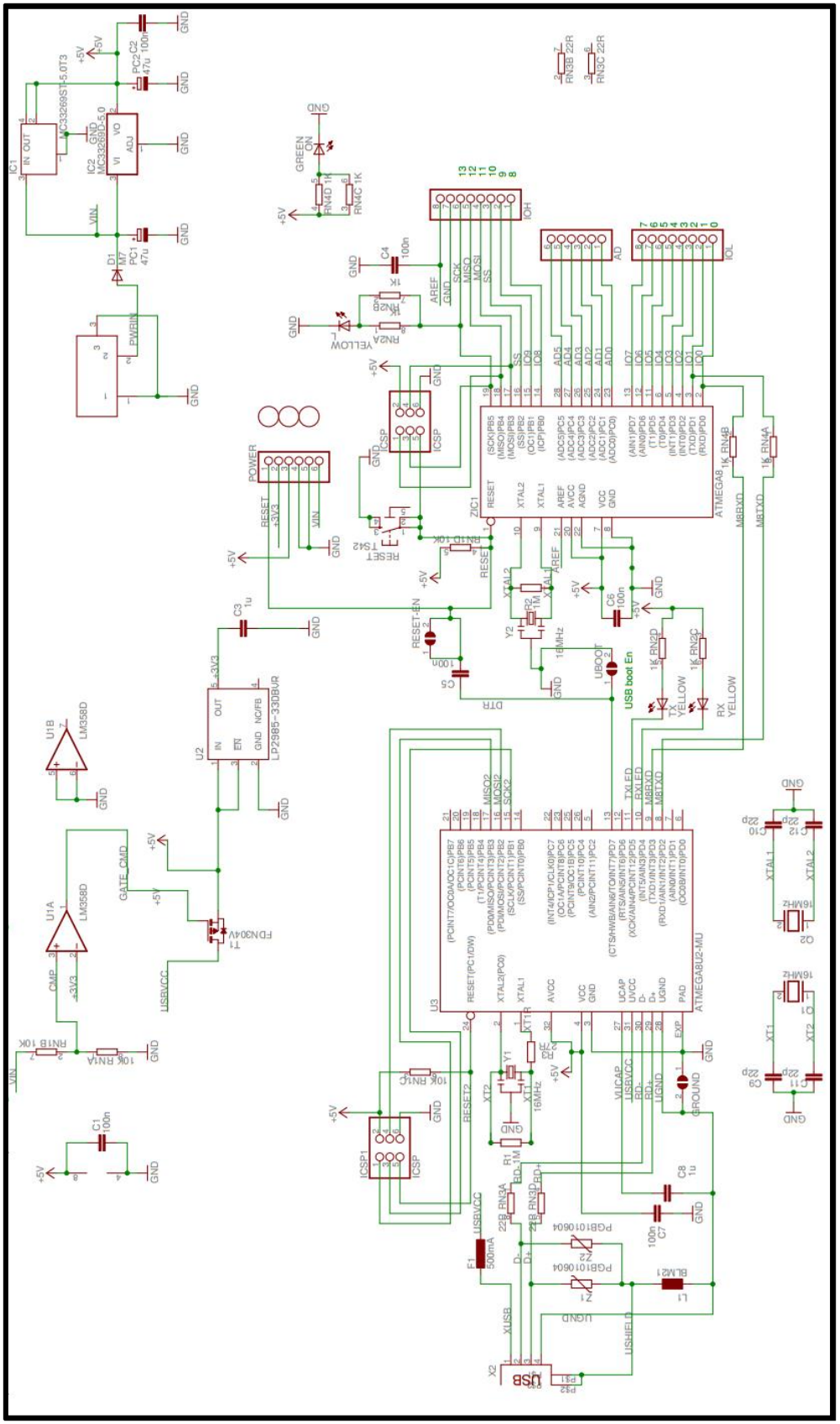

Figura 03: Esquema elétrico Arduíno UNO

Fonte: Arduíno(e) (2013).

Segundo o fabricante da plataforma, Arduíno(d) (2013), a mesma apresenta as seguintes especificações técnicas:

- Microcontrolador: ATmega328;
- Tensão Operacional: 5 V;
- Tensão de entrada (recomendada): 7-12 V;
- Tensão de entrada (limites): 6-20 V;
- Pinos E/S digitais: 14;
- Pinos de entrada analógica: 6;
- Corrente CC por pino E/S: 40 mA;
- Corrente CC para o pino 3,3 V: 50 mA;
- *Flash Memory*: 32 kB (ATmega328).
- SRAM: 2 kB (ATmega328);
- EEPROM: 1 kB (ATmega328);
- Velocidade de *Clock*: 16 MHz.

O Arduíno pode ser alimentado pela conexão USB ou com uma fonte de alimentação externa. A alimentação é selecionada automaticamente. Esta placa pode funcionar com uma fonte de alimentação externa de 6 a 20 V. No entanto se a alimentação for inferior a 7 V, o pino 5 V pode fornecer menos de 5 V, e a placa pode se mostrar instável. Se a alimentação for maior do que 12 V, o regulador de tensão pode superaquecer e danificar a placa. A faixa recomendada é de 7 a 12 V (ARDUÍNO(d) 2013).

Os pinos de alimentação são os seguintes (ARDUÍNO(d) 2013):

- VIN: a entrada de alimentação para a placa Arduíno, quando se está utilizando uma fonte de alimentação externa (em oposição à conexão USB ou outra fonte de alimentação regulada). Caso esteja alimentando através da USB, pode-se acessar a tensão de alimentação neste pino;
- 5 V: a fonte de alimentação regulada usada para o microcontrolador e para outros componentes na placa. Pode vir tanto do VIN através do regulador embarcado ou da conexão USB;
- 3V3: uma fonte de 3,3 V gerada pelo regulador embarcado. A corrente máxima suportada é de 50 mA.

Conforme Arduíno(d) (2013), o Arduíno UNO possui 14 pinos digitais que podem ser utilizados como uma entrada ou uma saída, utilizando-se as funções “pinMode()”, “digitalWrite()”, e “digitalRead()”, sendo que estes operam a 5 V. Cada pino pode fornecer ou receber um máximo de 40 mA, e possuem um resistor *pull-up* interno (desconectado por padrão) de 20-50 kΩ.

Além destes pinos, possui também alguns pinos específicos, tais como:

- Serial: 0 (RX) e 1 (TX): usados para receber (RX) e transmitir (TX) dados seriais TTL. Estes pinos são conectados aos pinos correspondentes do *chip* serial USB;
- LED: pino 13. Há um LED integrado ao pino digital 13. Quando este pino está no valor lógico alto, este LED está aceso;
- AREF: Tensão de referência para as entradas analógicas.
- Reset: reseta o microcontrolador ao receber um pulso de valor lógico zero.

Ainda para o fabricante, a comunicação entre o dispositivo e um computador ocorre de forma simples e fácil. O Atmega328 fornece comunicação serial TTL (5 V), que está disponível nos pinos digitais 0 (RX) e 1 (TX). Um ATmega16U2 na placa canaliza esta comunicação para a USB e aparece como uma porta virtual para o *software* no computador. O *firmware* do ATmega16U2 utiliza os *drivers* padrão USB/COM, não sendo necessário a instalação de *drivers* adicionais.

2.2.2 IDE do Arduíno

O IDE do Arduíno é utilizado para desenvolver o código e realizar o *upload* do mesmo para a plataforma. No mundo Arduíno, é comum o termo *sketch*, que representa o código (ou esboço) que está em desenvolvimento.

O *software* pode ser obtido gratuitamente no portal oficial do Arduíno (ARDUÍNO(b), 2013). Existem versões disponíveis para *Windows*, MAC, e *Linux*. No portal também é possível ter acesso ao código de programação do IDE do Arduíno, se o usuário assim desejar.

No software deve-se definir o modelo de plataforma Arduíno a ser utilizado, bem como a porta de comunicação.

Os menus disponíveis no *software*, bem como a descrição dos mesmos, podem ser vistas em detalhes no manual do fabricante (ARDUÍNO(a) 2013). O

fabricante também disponibiliza ao usuário a descrição e a explicação de como utilizar as funções e instruções necessárias para desenvolver o código, na linguagem utilizada pelo IDE do Arduíno (ARDUÍNO(c), 2013).

2.3 SCADABR

Segundo Filho (1996), a respeito do sistema de supervisão e controle, pode-se defini-lo como uma ferramenta, que integrada a um controlador, oferece a possibilidade de leitura e escritura de dados referentes ao processo.

ScadaBR (2010) afirma que sistemas SCADA são utilizados como interface entre processos e os operadores. São utilizados em diversos tipos de aplicação, desde aplicações mais simples como um pequeno procedimento industrial, até grandes aplicações como monitoramento de distribuição de energia elétrica.

Conforme Garbrecht (2012), os desenvolvimentos de aplicações de supervisão SCADA oferecem até 80% de economia no custo de engenharia.

Para ScadaBR (2010), um sistema SCADA deve fornecer *drivers* de comunicação com o dispositivo que se deseja controlar, registro de dados contínuos e uma *interface* gráfica para o operador, conhecida como IHM, necessária para supervisionar e controlar o sistema, ou somente supervisionar.

Filho (1996) leciona que os sistemas de supervisão e controle de processos produtivos, utilizando interfaces humano-máquina, podem ser vistos como a libertação do homem da enfadonha tarefa de controlar e comandar a máquina.

Segundo ScadaBR (2010), o *software* ScadaBR é baseado na política de *software* livre, possuindo distribuição gratuita, bem como acesso ao código-fonte. Se o usuário desejar, poderá realizar alterações e a redistribuição da versão de sua autoria.

“A interface principal do ScadaBR é de fácil entendimento e utilização, já fornecendo visualização das variáveis, gráficos, configuração de protocolos, alarmes, construção de telas tipo IHM, etc.” (SCADABR, 2013).

Conforme ScadaBR (2010), o ScadaBR é um sistema baseado em Java, que roda em mais de um sistema operacional, podendo ser *Linux* ou *Windows.* O sistema pode ser executado a partir de um servidor de aplicação HTTP, o qual permite que a aplicação seja acessada através de um navegador de *internet*, e por consequência, a partir de outro computador. O servidor de aplicação indicado pelo

desenvolvedor é o Apache Tomcat, que vem integrado ao instalador do aplicativo, facilitando a instalação do sistema.

Além de outras características, a gratuidade do *software*, a facilidade de construção de telas e de configuração de variáveis de comunicação justificaram a utilização desta ferramenta.

2.4 MODBUS

Conforme Fortec (2013), até a década de sessenta, o controle de processos industriais era realizado por comando eletromagnético, baseado na lógica de relés. Devido ao alto custo de manutenção e complexidade na alteração de um processo, anexo ao desenvolvimento da eletrônica, notou-se a necessidade do desenvolvimento de novas soluções.

Ainda segundo o autor, no final da referida década, foram implementados os primeiros Controladores Lógicos Programáveis, os CLP's. A responsável por isso foi a empresa Bedford Associates General Motors, que propôs um sistema que permitia realizar manobras de controle sem a necessidade de religação. O sistema recebeu o nome de Controlador Modular Digital, ou Modicon. Por volta de 1973 surgiu a possibilidade de comunicar dispositivos inteligentes, no padrão cliente-servidor, em um sistema industrial.

> Em 1979 a Modicon lançou o Modbus, um protocolo de comunicação de código aberto, de fácil implementação na área industrial, que permite a fácil troca de informações entre dispositivos. O mesmo também permitiu a programação remota, diagnóstico dos elementos conectados, entre outros. O mesmo estabelece uma estrutura de mensagens padrão, que os controladores reconhecem e executam, independente do tipo de rede que será utilizada para esta comunicação (FORTEC, 2013).

Denardin (2010) relata que o protocolo serial Modbus é um protocolo mestre-escravo. Um sistema operando como mestre-escravo possui um nó mestre, que emite comandos explícitos para um dos nós escravos e processa a sua resposta. Tipicamente os escravos não irão transmitir dados sem uma requisição do nó mestre e não se comunicam com outros escravos.

> O protocolo Modbus operando em linha serial é um protocolo mestre-escravos. Isso significa que somente um mestre é conectado ao barramento ao mesmo tempo. Quanto aos escravos, um ou mais nós (número máximo de 247) podem ser conectados a este mesmo barramento. Uma comunicação Modbus é sempre iniciada pelo mestre. O nó escravo nunca irá transmitir dados sem receber uma requisição do nó mestre. Os nós escravos nunca irão se comunicar entre eles. O nó mestre inicia somente uma transação Modbus por vez. (DENARDIN, 2010).

Para Denardin (2010), Modbus em linhas seriais podem usar diferentes interfaces físicas (RS485, RS232, etc.). A interface de dois fios é a mais comum. No entanto, a interface RS485 também pode ser implementada. A interface serial RS232 só poderá ser utilizada quando uma comunicação ponto a ponto de curta distância for requerida.

Na referência Cefet-RN (2013), algumas características do protocolo Modbus são fixas, como o formato da mensagem, funções disponíveis e tratamento de erros de comunicação. Outras características são selecionáveis, como o meio de transmissão, velocidade, *timeout*, *bits* de parada e paridade, e o modo de transmissão (RTU ou ASCII).

Denardin (2010) descreve que estes modos de transmissão definem o conteúdo de *bits* dos campos das mensagens transmitidas serialmente no barramento. Eles determinam como a informação é empacotada nos campos das mensagens e, posteriormente, como são decodificadas.

Denardin (2010) ainda diz que o modo de transmissão (e o padrão da porta serial) deve ser o mesmo em todos os dispositivos conectados a linha serial.

2.4.1 Modbus RTU

Conforme Denardin (2010), quando utilizado o modo RTU em linhas seriais, cada mensagem de um *byte* contém dois caracteres hexadecimais de quatro *bits*. A principal vantagem deste modo é que sua maior densidade de caracteres permite um melhor processamento de dados do que o modo ASCII para o mesmo *baud rate*. Cada mensagem deve ser transmitida em um fluxo contínuo de caracteres.

Ainda segundo o autor, no modo RTU, cada byte possui codificação binária de oito *bits*, sendo um *bit* de início, oito *bits* de dados (sendo o menos significativo enviado primeiro), um *bit* de paridade e um *bit* de parada. Na forma serial, o *caractere* é enviado na seguinte forma: Início, 1, 2, 3, 4, 5, 6, 7, 8, Paridade, Parada, sendo o primeiro o menos significativo, e o último o mais significativo.

2.4.1.1 Enquadramento de mensagens Modbus RTU

Para Denardin (2010), as mensagens Modbus são colocadas dentro de quadros com início e fins definidos. Isto garante que o dispositivo que está recebendo a mensagem saiba exatamente o início e o fim da mesma. Quando um quadro não é completado, erros devem ser gerados como resultado da leitura parcial.

Os quadros são separados por um intervalo de pelo menos 3.5 tempos de caractere. A figura 04 representa esta transmissão, considerando um intervalo de 3.5t.

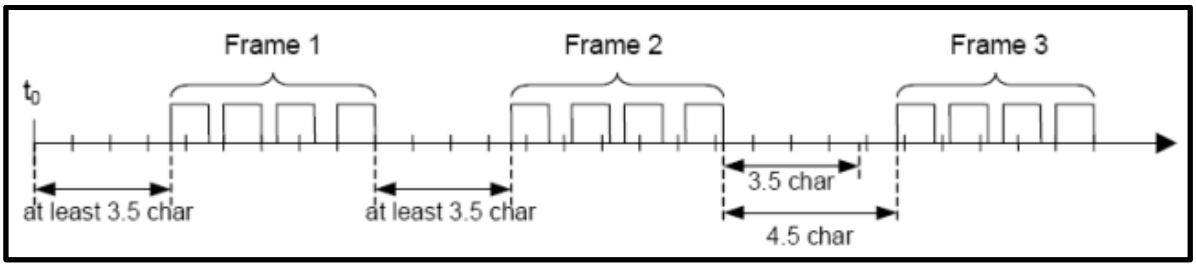

Figura 04: Quadro de mensagens Modbus

Fonte: Denardin (2010).

3 ESTRUTURAÇÃO DO PROTÓTIPO

Este capítulo apresentará a descrição do protótipo, bem como as etapas da sua construção.

3.1 DESCRIÇÃO DO PROTÓTIPO

O irrigador microcontrolado integrado a um sistema de supervisão e controle consiste em um equipamento que realiza de forma automática a irrigação de acordo com a temperatura e umidade relativa do ar.

O equipamento conta com um reservatório de água, que tem seus limites de mínimo e máximo controlados de forma automática. Toda a água utilizada para irrigação vem deste reservatório, sendo que o enchimento do mesmo é controlado por uma válvula solenoide. Conectada por mangueira a uma torneira externa, quando acionada, a válvula permite a entrada de água no reservatório.

O protótipo possui dois conjuntos formados por uma bomba, uma válvula solenoide e um sensor de vazão conectados em série. Estes conjuntos são ligados ao reservatório de água através de mangueiras.

Um dos conjuntos possui a finalidade de esgotamento do reservatório, ou seja, quando acionado retira a água do reservatório através de uma mangueira que pode ser direcionada pelo operador. Esta opção foi criada devido a eventuais necessidades de manutenção do reservatório de água.

O outro conjunto tem a finalidade de realizar a irrigação. Quando acionado, o mesmo envia água do reservatório para um aspersor, que é conectado ao sensor de vazão através de uma mangueira.

O protótipo possui dois modos de funcionamento, automático e manual. Operando de forma manual, o operador pode acionar as saídas do sistema no momento que desejar, bem como, acionar a válvula de enchimento do reservatório, o conjunto de esvaziamento do reservatório e o conjunto de irrigação. No modo manual, o próprio operador deverá fazer o controle visual do nível do reservatório.

Operando em modo automático, o equipamento identifica a temperatura e a umidade do ar por meio de um sensor. Através destes dados, os tempos de irrigação são determinados de acordo com o material de apoio fornecido pela Embrapa, que

pode ser verificado no ANEXO A – EMBRAPA - Cultivo de Tomate para Industrialização. Este material sugere os tempos de irrigação necessários para o cultivo do tomate. Para a apresentação do TCC, os tempos reais foram diminuídos de forma diretamente proporcional.

Ainda em modo automático, um sensor instalado na parte superior do reservatório envia constantemente ao controlador a distância em que a água se encontra do mesmo. Conhecendo a distância e a área do reservatório, o sistema calcula a quantidade de litros de água presente no recipiente. Com um valor mínimo e máximo de litros definido na programação, sempre que o valor mínimo é atingido o enchimento do reservatório é acionado, até atingir o valor máximo.

A operação e monitoramento do sistema ocorre através de um sistema de operação e controle desenvolvido no ScadaBR. O sistema pode ser acessado através de um *notebook*, *tablet* ou *smartphone*. Além de permitir a realização de todos os procedimentos de operação do equipamento, é apresentada ao usuário a quantidade de litros de água presente no reservatório, a temperatura ambiente, a umidade do ar, bem como a situação de todas as saídas do equipamento.

3.2 DESENVOLVIMENTO DO PROTÓTIPO

Após a realização de um estudo comparativo entre as opções de componentes disponíveis no mercado, sendo levado em consideração o baixo custo e a possibilidade de integração, foi dado início ao desenvolvimento do protótipo.

Neste capítulo será detalhada a construção física do protótipo, que tem seu diagrama de blocos ilustrado na figura 05. O diagrama mostra que o protótipo é dotado de um controlador e uma placa de *interface*. Todos os dispositivos elétricos e eletrônicos, de entrada e saída (exceto o sensor de temperatura e umidade do ar), são conectados a placa de *interface*, sendo ela a responsável por fazer a ponte entre o controlador e os dispositivos. O diagrama também apresenta um reservatório de água conectado mecanicamente a uma válvula solenoide de enchimento e as bombas responsáveis pela irrigação e esvaziamento do mesmo. O diagrama mostra os dispositivos necessários para que a irrigação ocorra no fim do processo.

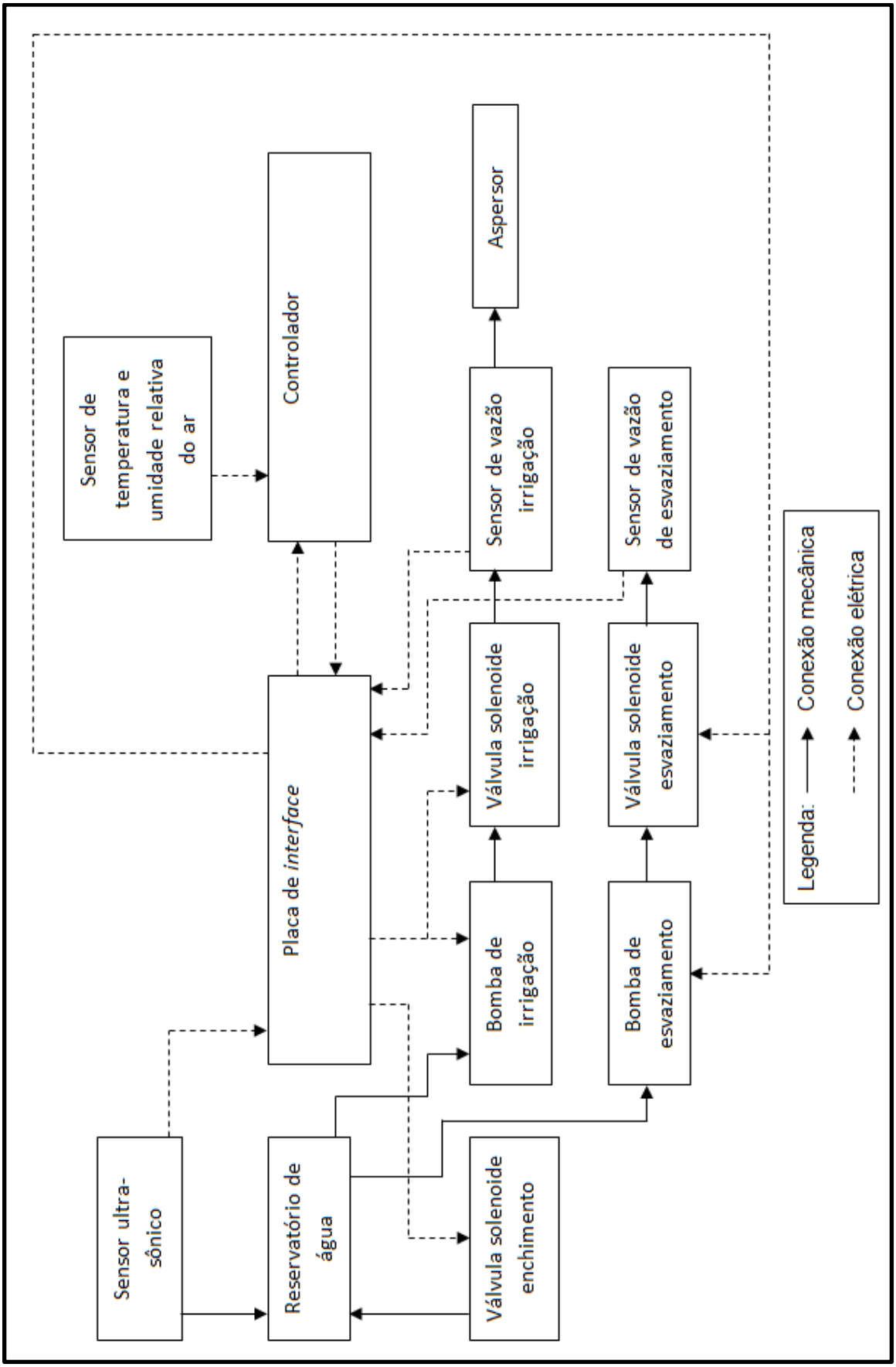

Figura 05: Diagrama de blocos do protótipo

Fonte: Autoria própria.

O primeiro passo do desenvolvimento se deu na adaptação de uma bancada para receber os dispositivos do protótipo. A figura 06 ilustra a bancada após a pintura e as adaptações necessárias para fixação dos dispositivos. Na mesma figura, também pode se observar as bombas e as válvulas solenoide, acionadas através de corrente alternada.

Figura 06: Bancada do protótipo
Fonte: Autoria própria.

Um recipiente plástico foi adaptado para ser usado como reservatório de água. O mesmo é ilustrado na figura 07.

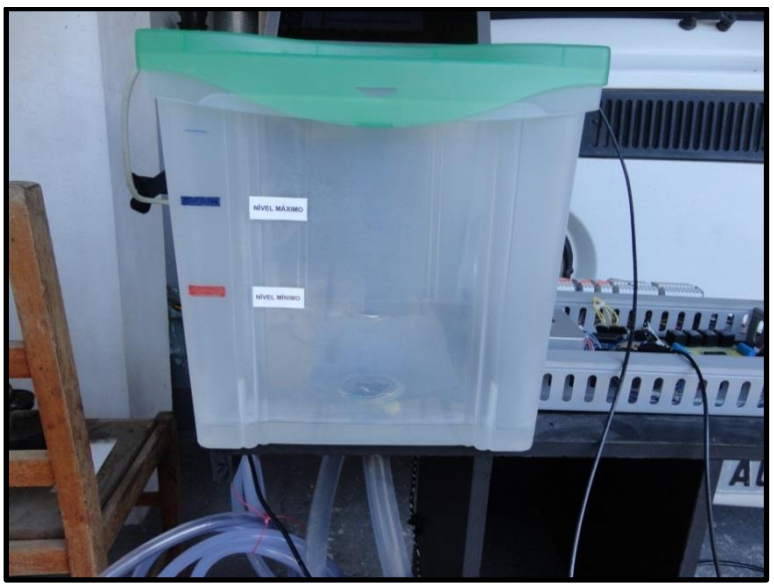

Figura 07: Reservatório de água
Fonte: Autoria própria.

Como mostra a figura 07, no fundo do reservatório foram instaladas as conexões para as mangueiras mostradas na figura 05, as quais possuem diâmetro de 7/8”. Para o enchimento do reservatório, uma válvula solenoide é conectada através de uma mangueira a uma torneira externa. Dessa forma, quando a válvula é acionada, ocorre a entrada de água no reservatório. Na figura também se observa as marcações de máximo e mínimo.

Para retirar a água do reservatório, foram usadas bombas acionadas por meio de uma alimentação de 127 V em corrente alternada, as quais trabalham com 34 W de potência, estes dados foram retirados da identificação do componente. A figura 08 ilustra em detalhes o modelo de bomba utilizado no protótipo.

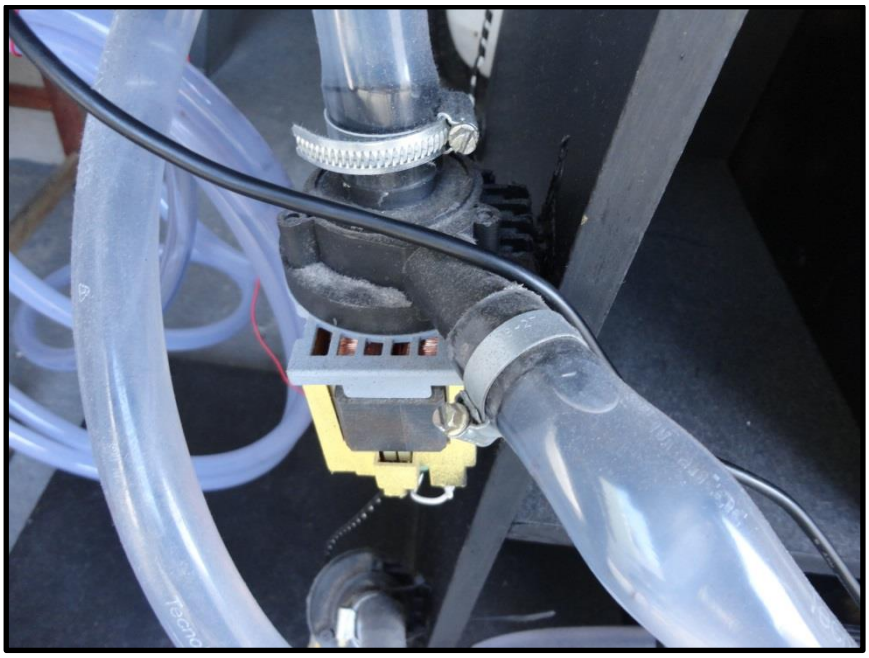

Figura 08: Bomba acionada através de corrente alternada
Fonte: Autoria própria.

Como se observa na figura 08, a admissão da água vinda do reservatório ocorre na parte superior do equipamento, sendo enviada pela saída frontal. O equipamento não retém a passagem da água quando submetida à pressão da gravidade, sendo que houve a necessidade de conectar uma válvula solenoide em série com a saída da bomba, para que não houvesse saída de água do reservatório de forma indesejada. Dessa forma, o conjunto bomba e válvula solenoide sempre será acionado simultaneamente.

A figura 09 ilustra a válvula solenoide, a qual permite a passagem de água quando é acionada com uma alimentação de 127 V em corrente alternada. Na imagem observa-se a admissão e a saída de água. A admissão ocorre pelo lado esquerdo e a saída pelo lado direito. Na parte traseira do equipamento se encontram os conectores de alimentação.

Figura 09: Válvula solenoide
Fonte: Autoria própria.

Para saber a quantidade de água retirada do reservatório, foi instalado um sensor de vazão em série com a válvula solenoide, ilustrado na figura 10.

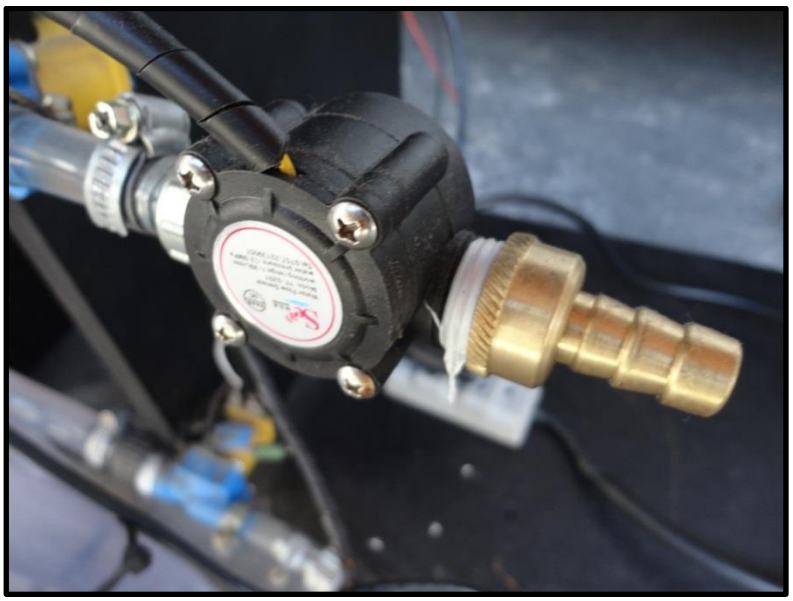

Figura 10: Sensor de vazão
Fonte: Autoria própria.

O sensor ilustrado na figura 10 é alimentado com 5 V em corrente contínua, sendo que o mesmo envia pulsos relacionados à vazão ao controlador. A leitura dos pulsos é atualizada a cada segundo, com o objetivo de definir em Hz a frequência lida. Por exemplo: se em um segundo o sensor enviar 55 pulsos, a frequência é de 55 Hz. Esta frequência é dividida por 4 (taxa de vazão do sensor), resultando na quantidade de l/min que está passando pelo sensor. Seguindo o exemplo acima, teríamos naquele segundo 13,75 l/min ou 229 ml, aproximadamente, sendo retirados do reservatório. A figura 09 ilustra o sensor de vazão utilizado no protótipo.

O protótipo é dotado de dois conjuntos formados por uma bomba, uma válvula solenoide e um sensor de vazão conectados em série, os quais serão chamados de conjunto “A” e conjunto “B”. O conjunto A é responsável pelo envio da água, quando acionado, a um aspersor que realiza a irrigação; o conjunto B tem a finalidade de esvaziar o reservatório em uma situação de manutenção.

Com a finalidade de saber a quantidade de litros de água presente no reservatório, foi instalado um sensor ultrassônico na parte superior do mesmo. O sensor realiza a leitura da distância do mesmo até a face água presente no reservatório. A figura 11 ilustra o sensor descrito acima.

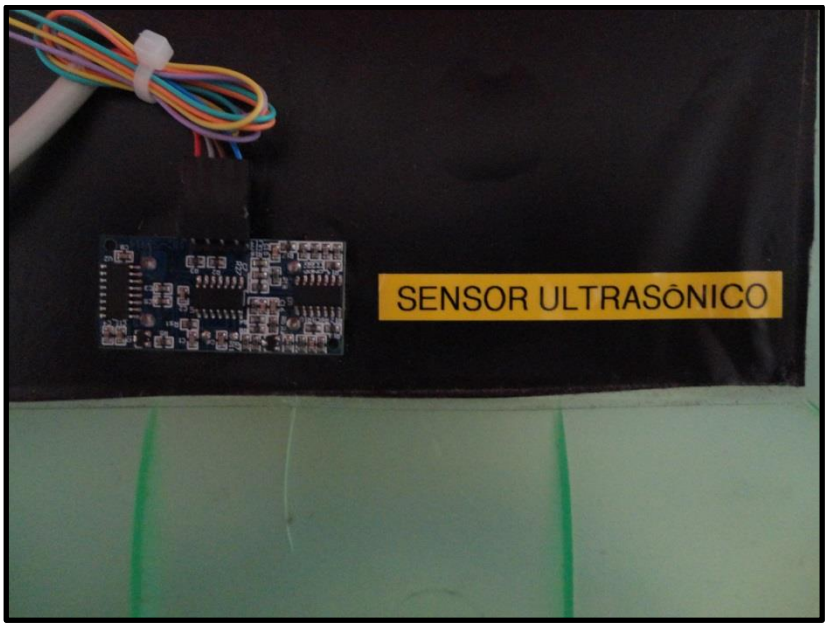

Figura 11: Sensor ultrassônico

Fonte: Autoria própria.

O sensor é alimentado com 5 V em corrente contínua, sendo que possui uma entrada e uma saída digital. Quando recebe o comando de disparo de leitura (vindo do controlador), emite um sinal ultrassônico que reflete na água. Quando a reflexão

é concluída, um pulso é enviado ao controlador. A relação entre os tempos de disparo de leitura e o pulso de retorno resulta na distância entre o sensor e a água em milímetros. Por exemplo, se o tempo for de 1,76 ms, a distância entre o sensor e a água é de 300 mm. A equação utilizada para este cálculo é apresentada no código de programação, no APÊNDICE A, com seus devidos comentários. Os dados técnicos de funcionamento e a forma de ligação do mesmo podem ser observados no manual do fabricante, em Elecfreaks (2013).

O protótipo realiza a leitura da temperatura e umidade relativa do ar. Essa leitura é realizada por um sensor que lê os dois valores e os envia constantemente ao controlador. A figura 12 ilustra o sensor de temperatura e umidade relativa do ar utilizado no protótipo.

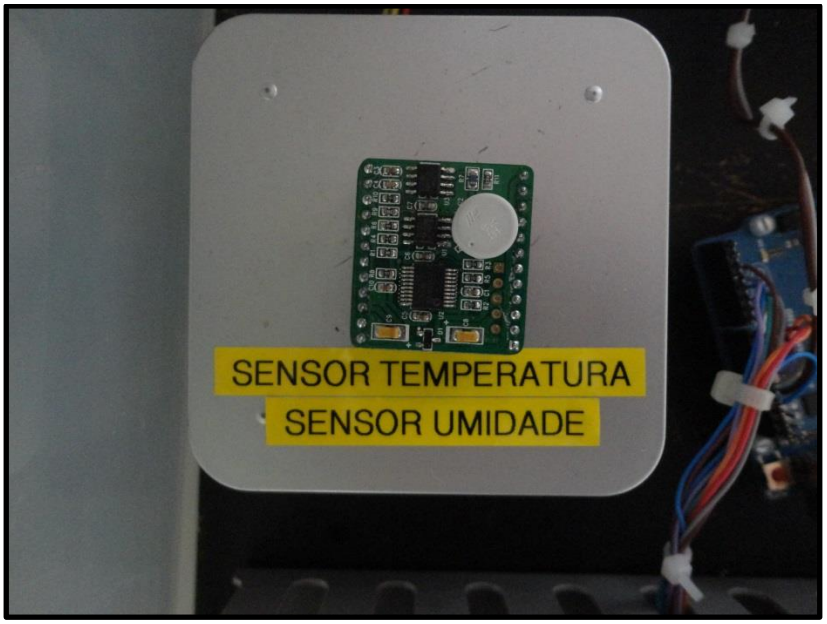

Figura 12: Sensor de temperatura e umidade relativa do ar
Fonte: Autoria própria.

O sensor é alimentado com 5 V em corrente contínua, sendo que envia de forma analógica ao controlador um sinal que varia de 0 a 5 V. O sinal é enviado através de dois pinos, sendo um responsável pela temperatura e o outro pela umidade relativa do ar. Uma escala relaciona a tensão elétrica enviada pelo sensor ao controlador com a temperatura e umidade relativa do ambiente. A escala é fornecida pelo manual do fabricante, em Sureeletronics (2009).

Os sinais emitidos pelos sensores instalados no protótipo são enviados a uma placa de *interface, que* diminui a flutuação dos sinais. A placa de *interface* também é

responsável pelo acionamento das saídas do protótipo, amplificando o sinal de acionamento vindo do controlador. A figura 13 ilustra a placa descrita acima.

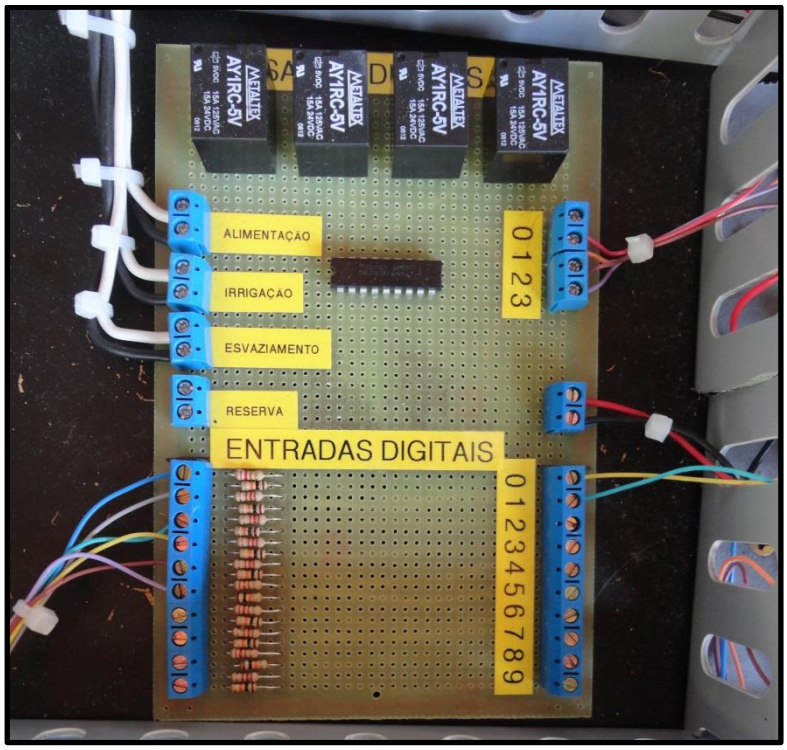

Figura 13: Placa de *interface*
Fonte: Autoria própria.

Na parte superior da placa se observa os relés de acionamento. O primeiro é responsável pelo acionamento da válvula solenoide de enchimento; o segundo pelo conjunto A; o terceiro pelo conjunto B; o quarto é reserva, facilitando a implementação de uma nova funcionalidade no projeto. O esquema elétrico da placa de *interface* está integrado ao esquema elétrico do protótipo, apresentado no item 3.3.

Os sinais emitidos pelos sensores, os comandos do operador e as funções desempenhadas pelo protótipo são processadas em uma plataforma de prototipagem eletrônica Arduíno UNO, na qual um microcontrolador interpreta e executa as funções presentes no código de programação desenvolvido. A figura 14 ilustra a plataforma instalada no protótipo com os dispositivos externos conectados aos pinos de entrada/saída.

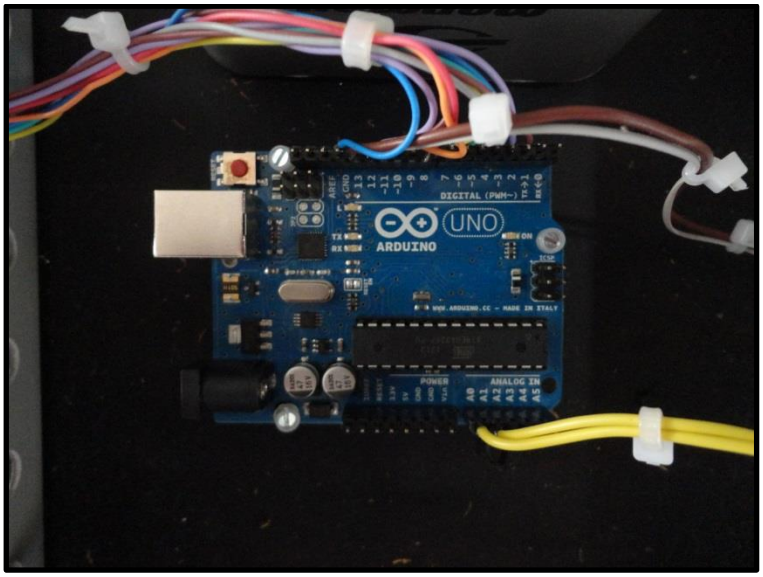

Figura 14: Plataforma Arduíno UNO instalada
Fonte: Autoria própria.

O projeto utiliza o modo de irrigação por aspersão convencional. O conjunto “A” envia água a um aspersor conectado por mangueira, o qual lança a água em forma de chuva sobre a área cultivada. A figura 15 ilustra o aspersor em funcionamento.

Figura 15: Aspersor
Fonte: Autoria própria.

A figura 16 ilustra o protótipo finalizado e as respectivas indicações dos dispositivos instalados.

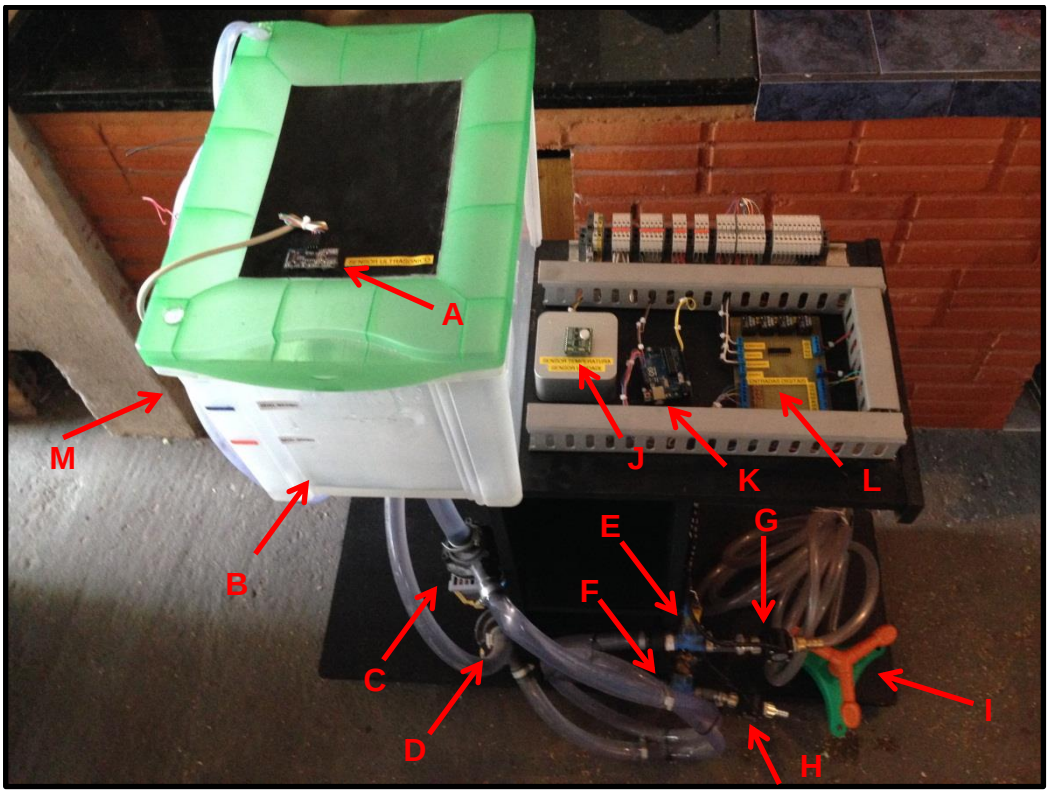

Figura 16: Protótipo finalizado

Fonte: Autoria própria.

Na figura acima, pode-se identificar os componentes descritos no diagrama de blocos do protótipo:

A. Sensor ultrassônico;
B. Reservatório de água;
C. Bomba de irrigação;
D. Bomba de esvaziamento;
E. Válvula solenoide de irrigação;
F. Válvula solenoide de esvaziamento;
G. Sensor de vazão de irrigação;
H. Sensor de vazão de esvaziamento;
I. Aspersor;
J. Sensor de temperatura e umidade relativa do ar;
K. Controlador;
L. Placa de *interface;*
M. Válvula solenoide de enchimento.

3.3 ESQUEMA ELÉTRICO

As conexões elétricas efetuadas no protótipo são ilustradas na figura 17.

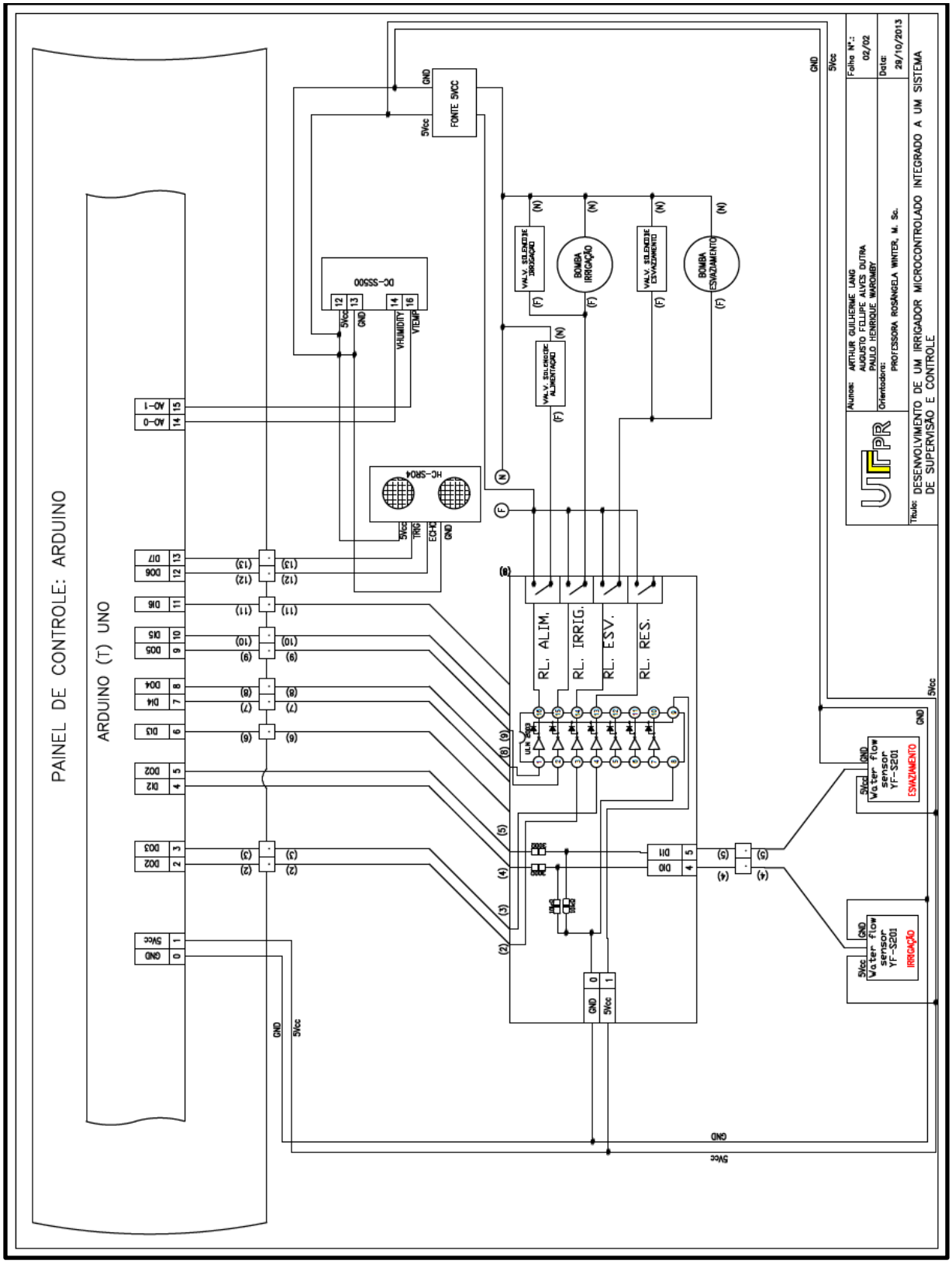

Figura 17: Esquema elétrico

Fonte: Autoria própria.

Na figura 17, observa-se o controlador na parte superior e a placa de interface na parte central. Os sensores são conectados a placa de *interface*, que reduz a flutuação do sinal e o envia ao controlador. Somente o sensor de temperatura e umidade relativa do ar é conectado diretamente ao controlador.

No canto direito central é possível observar que as saídas do protótipo são conectadas à placa de *interface*, que amplifica o sinal de acionamento vindo do controlador para realizar o acionamento dos relés. É possível verificar o acionamento em paralelo dos conjuntos bomba e válvula solenoide de irrigação, bem como do conjunto responsável pelo esvaziamento.

Com relação à alimentação, observa-se a existência de uma fonte 5 V em corrente contínua que alimenta o controlador e os sensores, bem como a alimentação 127 V em corrente alternada que alimenta as saídas do protótipo.

3.4 CÓDIGO DE PROGRAMAÇÃO

O código de programação, também chamado de *firmware,* desenvolvido no IDE do Arduíno é representado pelo fluxograma apresentado na figura 18.

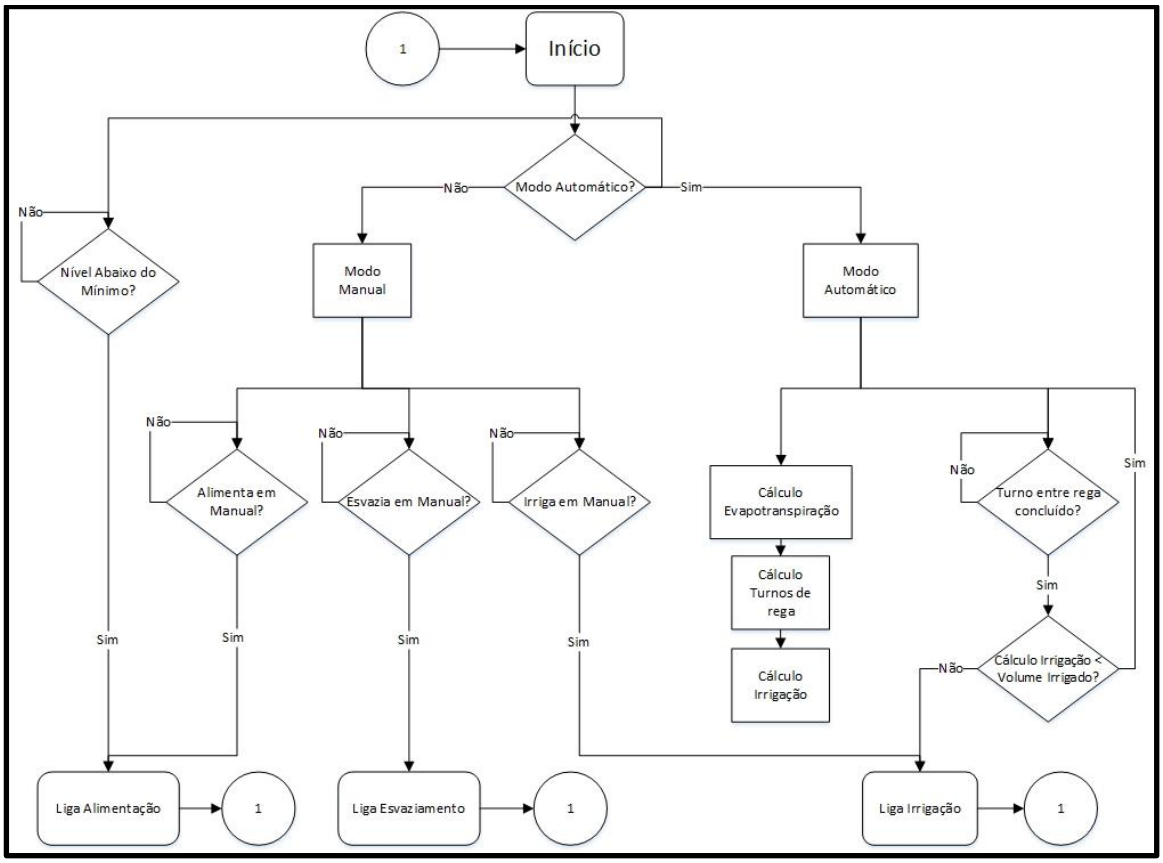

Figura 18: Fluxograma do código de programação

Fonte: Autoria própria.

A figura 18 ilustra a sequência do código de programação do protótipo. A primeira decisão é tomada ao verificar o modo de operação definido pelo usuário: automático ou manual.

Se o modo selecionado for manual, o código aguarda o comando do operador para acionar as saídas individualmente: enchimento do reservatório; esvaziamento do reservatório e/ou irrigação.

Se o modo selecionado for automático, o código realiza três operações simultaneamente:

1. Monitora o nível do reservatório: utilizando os dados do sensor ultrassônico; se o nível mínimo for atingido, aciona o enchimento do reservatório até atingir o nível máximo;
2. Calcula o volume a ser irrigado e o intervalo de tempo de irrigação (turno entre rega), através dos dados lidos pelo sensor de temperatura e umidade do ar, baseado nos dados fornecidos pelo ANEXO A – EMBRAPA – Cultivo de tomate para industrialização;
3. Verifica se o intervalo de tempo da irrigação foi atingido; sendo concluído, verifica se o volume de irrigação calculado é menor que o volume irrigado efetivamente. Enquanto o valor irrigado não atingir o valor calculado, a irrigação se mantém acionada.

O código de programação completo, com as funções programadas devidamente comentadas, se encontra no APÊNDICE A – Código de programação do protótipo.

3.5 SISTEMA DE SUPERVISÃO E CONTROLE

O sistema de supervisão e controle foi desenvolvido no *software* ScadaBR. Através da *interface* humano-máquina, o operador tem o controle de todo o protótipo, podendo realizar os procedimentos de operação e visualizar o *status* das saídas do equipamento.

O sistema de supervisão e controle é executado em um *notebook* que fica conectado ao controlador Arduíno UNO através da porta USB, que foi denominado servidor. A comunicação ocorre de forma serial, sendo a plataforma Arduíno UNO responsável pela conversão dos dados USB/Serial.

Os comandos inseridos pelo operador são registrados em variáveis enviadas ao controlador, que as processa e ativa as saídas de acordo com o código de programação, enviando ao sistema de supervisão e controle o valor atual das saídas, para que possa ser exibido ao operador.

Como o ScadaBR é rodado a partir de um servidor de aplicação, o sistema de supervisão e controle pode ser acessado a partir de qualquer dispositivo (*notebook, smartphone* ou *tablet*) que possua a tecnologia *wireless*.

O *notebook* servidor é conectado a um roteador *wireless*. Dessa forma, os demais dispositivos que queiram acessar o sistema de supervisão e controle devem ser conectados a esse roteador, acessando através de seus navegadores de *internet* o endereço da aplicação, no seguinte formato: http://192.168.1.133:8080/ScadaBR/views.shtm, onde se acessa o endereço IP do servidor.

Para conectar a aplicação, é necessário realizar a autenticação fornecendo os dados do usuário e senha, pré-cadastrados no ScadaBR. O usuário cadastrado no projeto é “admin”, com senha “admin”.

3.5.1 Operação do sistema de supervisão e controle

Ao conectar a aplicação, o operador encontra a *interface* humano-máquina ilustrada na figura 19.

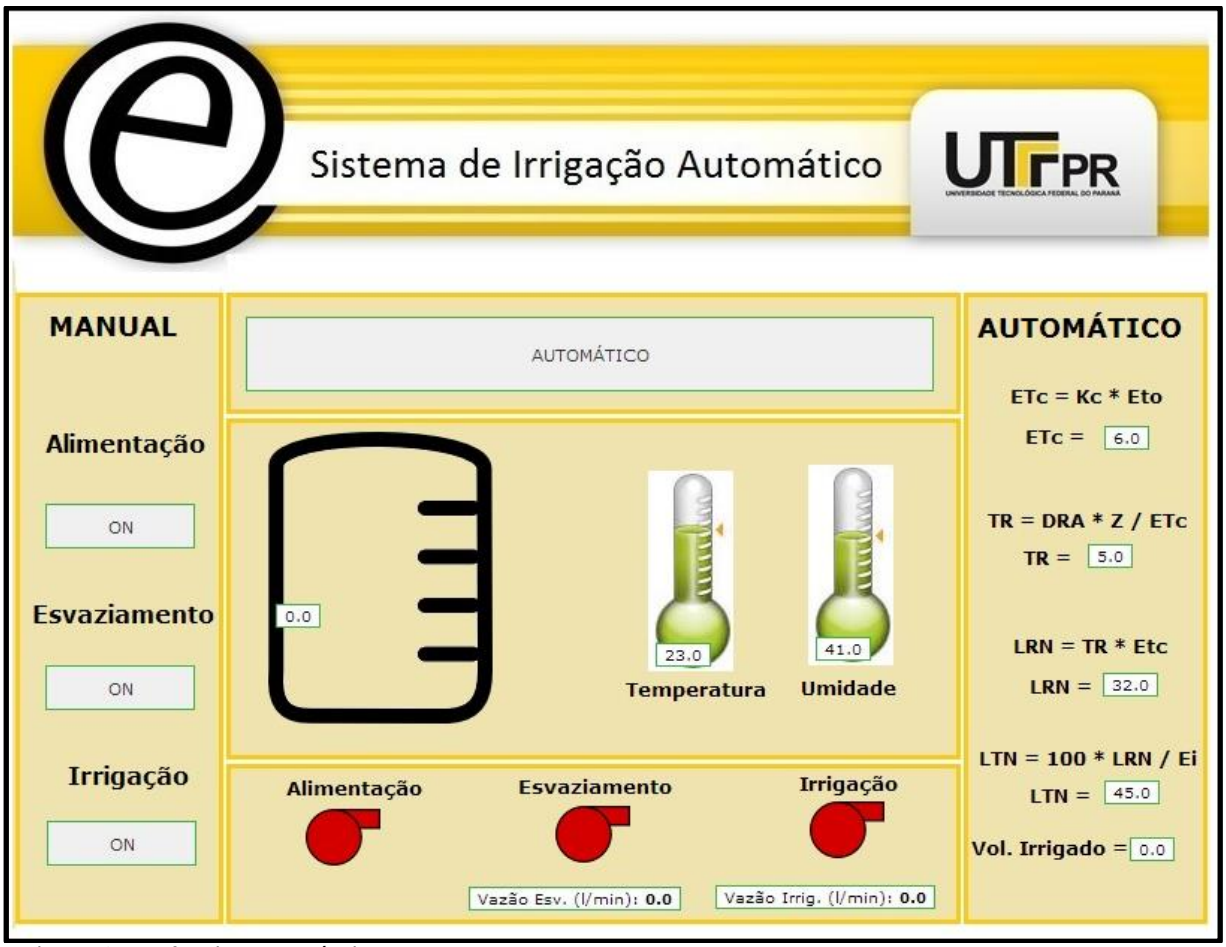

Figura 19: *Interface* humano-máquina

Fonte: Autoria própria.

No centro da tela é apresentada em tempo real a quantidade de litros presente no reservatório de água, bem como a temperatura ambiente e a umidade relativa do ar.

Sobre as indicações descritas acima, existe o botão em que o operador altera o modo de funcionamento entre automático e manual.

Do lado esquerdo da tela, operando em modo manual, clicando no botão “ON” podem ser acionadas as saídas desejadas: alimentação atua o enchimento do reservatório; esvaziamento aciona o conjunto “B”; e irrigação atua o conjunto “A”, que envia água ao aspersor.

Na parte inferior da tela é exibido o *status* das saídas do protótipo, quando a saída está acionada a mesma é exibida na cor verde. O operador visualiza também a vazão em l/min dos conjuntos “A” e “B” quando estão acionados.

Operando em modo automático, o protótipo está programado para efetuar a irrigação simulando o cultivo de tomate, em fase de frutificação. No lado direito da figura 17 são exibidos os valores das variáveis envolvidas no cálculo da irrigação. A descrição detalhada deste cálculo é exibida no ANEXO A – EMBRAPA - Cultivo de Tomate para Industrialização.

3.6 TESTES E RESULTADOS

Antes de iniciar a integração dos componentes, foi necessário realizar seus testes individualmente.

O sensor de temperatura e umidade fornece dois sinais analógicos referentes às medições realizadas dentro da faixa de capacidade de leitura. Para validar os dados lidos pelo sensor, e tomar como base de comparação, foram verificados nos meios de comunicação (jornais e *internet*) as leituras diárias de temperatura e umidade do ar. Como os valores foram compatíveis, tornou-se viável a utilização do sensor no projeto.

O sensor ultrassônico fornece uma leitura da distância até o objeto desejado, que no protótipo é a lâmina de água. Funciona enviando um sinal de ultrassom que reflete na superfície da água e retorna ao sensor. Dessa forma, o tempo que o sinal leva desde a emissão até a recepção, multiplicado pela velocidade conhecida do som (aproximadamente 340 m/s) é obtida a distância até a lâmina de água. A altura total do tanque, subtraído da distância lida pelo sensor ultrassônico, revela a altura de água contida no tanque. A área da base do tanque multiplicada pela altura de água resulta no volume de água contido no tanque.

As distâncias lidas pelo sensor ultrassônico puderam ser validadas medindo lateralmente ao tanque a distância da lâmina de água até a face do sensor. Os dados obtidos foram fiéis aos sinais do sensor, confirmada e admitida a tolerância de erro informada pelo fabricante.

Os sensores de leitura de vazão informam a quantidade de litros por minuto nos sistemas de irrigação e esvaziamento do protótipo. Com esses dados é possível enviar resposta do volume irrigado ao controle de irrigação. Os dados são obtidos

através de um rotor interno no sensor, que gira proporcionalmente à vazão do circuito, enviando pulsos ao controlador. O fabricante informa a taxa de vazão que deve ser relacionada à frequência dos pulsos lidos, tendo como resposta a vazão em litros por minuto.

Os dados obtidos de leitura de vazão foram corroborados alimentando um recipiente com capacidade conhecida e monitorando o tempo até o seu enchimento. Os dados obtidos foram fiéis aos sinais do sensor, confirmada e admitida a tolerância de erro informada pelo fabricante.

Com o protótipo finalizado, foram realizados testes de funcionamento em dias nos quais foram medidos valores diferentes de temperatura e umidade relativa do ar. Verificou-se que os cálculos do volume de água a ser irrigado e o turno entre regas foram bem diversos, conforme os valores de temperatura e umidade relativa do ar lidos pelo sensor naquele momento. Os tempos foram acompanhados através de cronômetro, sendo que atenderam as expectativas do protótipo.

4 CONSIDERAÇÕES FINAIS

O presente projeto teve o propósito de viabilizar o uso da automação aliada a um baixo custo, em sistemas de irrigação de pequeno porte, diminuindo assim possíveis perdas na produção.

Para atingir o propósito do baixo custo, buscaram-se tecnologias baratas e confiáveis, passíveis de integração entre si. A tabela 01 apresenta os custos estimados do desenvolvimento do protótipo.

ESTIMATIVAS DE CUSTOS PARA EXECUÇÃO DO PROJETO				
SISTEMA DE AUTOMAÇÃO				
Produto	Descrição	Valor unitário	Qtde	Total
Arduíno Uno	Controladora	R$ 89,00	1	R$ 89,00
Sensor HC-SR04	Sensor ultrassônico	R$ 8,15	1	R$ 8,15
Sensor DC-SS500	Sensor de umidade e temperatura	R$ 24,95	1	R$ 24,95
Sensor YF-S201	Sensor de vazão de água	R$ 21,38	2	R$ 42,76
Fonte 5V	Fonte de alimentação 5 V 2 A	R$ 35,00	1	R$ 35,00
SUBTOTAL AUTOMAÇÃO				**R$ 199,86**
EQUIPAMENTOS MECÂNICOS, HIDRÁULICOS E ACESSÓRIOS				
Produto	Descrição	Valor unitário	Qtde	Total
Aspersor	Equipamento responsável pela distribuição da água	R$ 21,00	1	R$ 21,00
Tanque	Tanque de 30 l	R$ 40,00	1	R$ 40,00
Bomba d'água	Bomba d'água 127 V/34 W	R$ 35,00	2	R$ 70,00
Válvula solenoide	Válvula solenoide 127 V	R$ 15,00	3	R$ 45,00
Mangueiras	Mangueira de polietileno AD 10m	R$ 30,00	1	R$ 30,00
Estrutura	Antiga estante, adaptada ao TCC	-	1	-
Braçadeiras		R$ 15,00	1	R$ 15,00
SUBTOTAL MECÂNICA, HIDRÁULICA E ACESSÓRIOS				**R$ 221,00**
TOTAL				**R$ 420,86**

Tabela 01: Estimativa de custos para execução do projeto

Fonte: Autoria própria.

Utilizando plataforma de prototipagem eletrônica, substituiu-se a utilização de CLP’s (equipamentos geralmente utilizados neste tipo de automação, os quais possuem um custo elevado).

Ao utilizar sensores de prototipagem que atendessem as necessidades do projeto, notou-se a viabilidade econômica através da importação dos mesmos.

Para comprovar o baixo custo da execução do projeto, elaborou-se a estimativa de custo se o mesmo projeto fosse executado utilizando CLP e sensores industriais. A tabela 02 apresenta o comparativo.

COMPARATIVO DE CUSTOS				
SISTEMA DE AUTOMAÇÃO				
Produto	Descrição	Valor unitário	Qtde	Total
Spyder PUL64385-ILC	CLP Honeywell com 6UI/4DI/3AO/8DO	R$ 2.174,59	1	R$ 2.174,59
PN RL + EP	Relé de Nível e pendulo para relé	R$ 325,52	1	R$ 325,52
Sensor TR21-H	Sensor de umidade e temperatura Honeywell	R$ 617,69	1	R$ 617,69
Sensor YF-S201	Sensor de vazão de água	R$ 21,38	2	R$ 42,76
Quadro de automação	Quadro com alimentação e saídas de bornes para campo	R$ 1.452,45	1	R$ 1.452,45
SUBTOTAL AUTOMAÇÃO				**R$ 4.613,01**
EQUIPAMENTOS MECÂNICOS, HIDRÁULICOS E ACESSÓRIOS				
Produto	Descrição	Valor unitário	Qtde	Total
Aspersor	Equipamento responsável pela distribuição da água	R$ 21,00	1	R$ 21,00
Tanque	Tanque de 30 l	R$ 40,00	1	R$ 40,00
Bomba d'água	Bomba d'água 127 V/34 W	R$ 35,00	2	R$ 70,00
Válvula solenoide	Válvula solenoide 127 V	R$ 15,00	3	R$ 45,00
Mangueiras	Mangueira de polietileno AD 10m	R$ 30,00	1	R$ 30,00
Estrutura	Antiga estante, adaptada ao TCC	-	1	-
Braçadeiras		R$ 15,00	1	R$ 15,00
SUBTOTAL MECÂNICA, HIDRÁULICA E ACESSÓRIOS				**R$ 221,00**
TOTAL				**R$ 4.834,01**

Tabela 02: Comparativo de custos

Fonte: Autoria própria.

No estudo comparativo foram mantidos os equipamentos mecânicos e hidráulicos, pois estes são passíveis de dimensionamento de acordo a necessidade do cliente em uma situação de implementação em campo.

Analisando as tabelas 01 e 02, é notável a redução de custo obtida com a integração das tecnologias utilizadas no projeto, como a utilização de plataforma de prototipagem microcontrolada à utilização de CLP. Os valores dos itens citados são referentes ao ano 2013.

Seguindo a mesma linha de raciocínio, utilizou-se bombas e válvulas solenoides geralmente presentes em eletrodomésticos, garantindo o baixo custo de aquisição e atendendo a necessidade do protótipo,

Com o uso de um sistema de supervisão e controle baseado na política de *software* livre, contempla-se a facilidade de operação e informatização sem geração de custo.

Em uma suposta aplicação real do protótipo em uma lavoura, somente seria necessário alterar as bombas utilizadas, o reservatório de água e o circuito hidráulico, já que o sistema de automação atende todos os requisitos.

Como sugestões de melhorias futuras, pode-se oferecer ao agricultor diversas opções de culturas irrigadas. Por exemplo, tornar selecionável no sistema de supervisão e controle o tipo de produto cultivado, não sendo necessárias alterações no código de programação no caso de alteração do produto cultivado pelo produtor.

Ainda nesta linha de raciocínio é possível realizar a irrigação de várias áreas de cultivo distintas paralelamente. Por exemplo, no sistema de supervisão e controle, gerenciar a irrigação de várias áreas de plantio distintas e independentes entre si simultaneamente.

Também como proposta de melhoria futura, sugere-se a implantação de sensores industriais, que possuem maior precisão e durabilidade, o que aumentaria o grau de confiabilidade do projeto.

Outra sugestão de complementação do projeto é a elaboração dos manuais de instalação, manutenção e operação.

5 CONCLUSÃO

Neste projeto abordou-se a disponibilização de tecnologias de irrigação para pequenos produtores rurais.

Um dos objetivos era pesquisar as tecnologias utilizadas na criação do protótipo. Este objetivo foi contemplado ao realizar o estudo dos sensores disponíveis no mercado, aprendendo o seu modo de funcionamento, suas aplicabilidades e definido os modelos a serem utilizados.

Após a análise das plataformas de prototipagem disponíveis, definiu-se pelo Arduíno UNO devido a sua facilidade de utilização e ampla possibilidade de integração de equipamentos. Foi necessário o aprendizado do seu modo de funcionamento e programação, testando códigos de programação para conhecer individualmente as funções que o equipamento fornece.

Buscou-se uma ferramenta baseada na política de *software* livre, que permitisse a criação de um sistema de supervisão e controle capaz de fornecer toda a *interface* de operação ao usuário e a comunicação com o controlador. Foi definido o ScadaBR, pois o mesmo facilita a criação das telas e a configuração das variáveis enviadas ao controlador.

Definidos o controlador e o sistema de supervisão e controle, o protocolo de comunicação de *hardware* utilizado foi o Modbus RTU, pois garante a integridade dos dados trafegados, garantindo em quadros definidos a mensagem enviada e recebida.

Como o ScadaBR permite a execução do sistema de supervisão e controle a partir de um servidor de aplicação, observou-se a possibilidade de operação de sistemas através de dispositivos móveis conectados a uma mesma rede *wireless*. Deixando um *notebook* servidor conectado ao controlador e rodando a aplicação, é possível acessá-lo a partir de outros dispositivos, como *tablets*, *smartphones* e *notebooks*.

Os dispositivos pertencentes ao protótipo foram testados individualmente para entender seus funcionamentos e a comunicação com o controlador. Para cada item era desenvolvido um código de programação de teste. Quando todos os itens foram testados individualmente, iniciou-se a integração dos equipamentos, com a elaboração do código de programação do protótipo.

Após o protótipo ser finalizado, foram realizados diversos testes e comissionamentos, havendo a necessidade de pequenas alterações de *hardware* e de *firmware*, a fim de conseguir atingir os objetivos propostos.

Uma das principais dificuldades encontradas foi o desenvolvimento do código de programação, pois a linguagem utilizada pelo controlador não era familiar à equipe. A sua realização foi possível com o estudo de materiais de apoio e o auto aprendizado.

Encontrou-se também dificuldade na utilização dos sensores, pois o material fornecido pelos fabricantes não possuía detalhamento de suas programações a contento. A lógica teve que ser criada a partir de testes e observações realizadas pela equipe.

Com a finalização do projeto atingiu-se os objetivos propostos, pois o mesmo foi implementado um baixo custo de execução e facilidade de operação, aproximando-se na medida do possível, o protótipo de uma situação real de implantação.

Com o irrigador microcontrolado integrado a um sistema de supervisão e controle, o pequeno agricultor pode diminuir suas perdas provenientes de problemas de irrigação com um baixo investimento, inserindo a informatização e automação na produção rural.

REFERÊNCIAS

AGRODIÁRIO. **Produtor implanta sistema de irrigação**. Disponível em: <http://www.diariodamanha.com/noticias.asp?id=50600> Acesso em: 24 mai. 2013.

AGROSOFT BRASIL. **Irrigação noturna melhora renda de produtores no Paraná**. Disponível em: <http://www.agrosoft.org.br/agropag/215047.htm> Acesso em: 02 mar. 2013.

ANDRADE, Camilo de Lelis. **Seleção do Sistema de Irrigação**. Disponível em: <http://www.jc.iffarroupilha.edu.br/site/midias/arquivos/20121019102321 34selecao_do_sistema_de_irrigacao.pdf> Acesso em: 24 mai. 2013.

ARDUÍNO(a). **Arduino *Development Environment*.** Disponível em: < http://arduino.cc/en/Guide/Environment> Acesso em 01 ago. 2013.

ARDUÍNO(b). ***Download the* Arduino *Software*.** Disponível em: < http://arduino.cc/en/Main/Software> Acesso em 01 ago. 2013.

ARDUÍNO(c). ***Language Reference.*** Disponível em: < http://arduino.cc/en/Reference/HomePage> Acesso em 01 ago. 2013.

ARDUÍNO(d). **Especificações técnicas.** Disponível em: <http://arduino.cc/en/Main/ArduinoBoardUno> Acesso em: 15 jul. 2013.

ARDUÍNO(e). **Esquema elétrico.** Disponível em: <http://arduino.cc/en/uploads/Main/Arduino_Uno_Rev3-schematic.pdf> Acesso em: 15 jul. 2013.

BERNARDO, Salassier. **Manual de irrigação**. 6ª ed. Viçosa-MG: UFV, 2002.

CEFET-RN. **Redes Industriais – Visão geral dos protocolos MODBUS.** Disponível em: < http://www.ebah.com.br/content/ABAAAgBwwAE/redes-industriais-vis-geral-dos-protocolos-modbus> Acesso em 02 set. 2013.

CIMINOS. **Sistema de controle de abastecimento e irrigação de plantio.** Disponível em: <http://blog.ciminos.com.br/post/Sistema-de-Controle-de-Abastecimento-e-Irrigacao-de-Plantil.aspx>. Acesso em: 21 ago. 2013.

DENARDIN, Gustavo Weber. **Modbus.** Disponível em: <http://pessoal.utfpr.edu.br/gustavo/Modbus_apostila.pdf>. Acesso em: 27 ago. 2013.

DILLON, A. ***Do differences in the scale of irrigation projects generate different impacts on poverty and production*?**. *Journal of Agricultural Economics*. v. 62.n. 2. p. 474-492. 2011.

ELECFREAKS. ***Ultrasonic Ranging Module HC-SR04***. Disponível em: <http://elecfreaks.com/store/download/HC-SR04.pdf>. Acesso em 01 out. 2013.

EMBRAPA. **Cultivo de tomate para industrialização**. . Disponível em: < http://sistemasdeproducao.cnptia.embrapa.br/FontesHTML/Tomate/TomateIndustrial_2ed/irrigacao.htm#tabela3>. Acesso em 08 ago. 2013.

FILHO, Domingos Leite Lima. **A utilização de sistemas supervisórios em processos industriais**. 1996. Monografia para obtenção de título de especialista no curso de pós-graduação em Automação Industrial – CEFET - PR, Curitiba - PR.

FORTEC. **Modbus**. Disponível em: <http://aquiro.weebly.com/modbus.html>. Acesso em: 27 ago. 2013.

GARBRECHT, Steven D. Os benefícios de arquiteturas baseadas em objetos para SCADA e Sistemas Supervisórios. **Mecatrônica Atual – Automação Industrial de Processos e Manufatura**, Taubaté – SP, v. 59, p. 16-20, nov-dez 2012.

GONÇALVES, Carolina. **Embrapa trabalha para inserir pequenos produtores em projeto de agricultura sustentável**. Disponível em: <http://agenciabrasil.ebc.com.br/noticia/2012-03-17/embrapa-trabalha-para-inserir-pequenos-produtores-em-projeto-de-agricultura-sustentavel>. Acesso em: 12 abr. 2013.

LIMA, Jorge Enoch Furquim Werneck; FERREIRA, Raquel Scalia Alves; CHRISTOFIDIS, Demetrios. **O uso da irrigação no Brasil.** Disponível em: < http://ag20.cnptia.embrapa.br/Repositorio/irrigacao_000fl7vsa7f02wyiv80ispcrr5frxoq4.pdf>. Acesso em: 12 dez. 2013.

MANTOVANI, E. C.; BERNARDO, S.; PALARETTI, L. F**. Irrigação: princípios e métodos**. 3. ed. Viçosa - MG: UFV, 2009. 355 p.

MCROBERTS, Michael. **Arduíno Básico.** Tradução Rafael Zanolli. São Paulo – SP: Editora Novatec, 2011.

PEREIRA, Pedro Antonio Arraes. **Irrigação e fertirrigação em fruteiras e hortaliças**. Brasília – DF: Embrapa Informação Tecnológica, 2011

REICHARDT, K. **A água na produção agrícola**. São Paulo - SP: McGraw-Hill do Brasil, 1978.

RODRIGUES, Geraldo Stachetti; IRIAS, Luiz José Maria. **Considerações sobre os Impactos Ambientais da Agricultura Irrigada.** Jaguariúna – SP: Embrapa, 2004

SCADABR. **Manual do *software***. Disponível em: < http://ufpr.dl.sourceforge.net/project/scadabr/Software/Docs/Manual%20ScadaBR.pdf> Acesso em 04 ago. 2013.

SILVA, Leonardo Duarte Batista da; FOLEGATTI, Marcos Vinícius. **Manejo da irrigação.** Rio de Janeiro – RJ: UFRRJ, 2007.

SOUZA, David José de. **Desbravando o PIC – Ampliado e atualizado para PIC 16F628A**. São Paulo - SP: Editora Érica, 2009.

SUREELETRONICS. ***Temperature and Relative Humidity Sensor Module User's Guide***. Disponível em: <http://www.sureelectronics.net/pdfs/DC-SS500_Ver1.0_EN.pdf>. Acesso em 30 set. 2013.

SUZUKI, M.; HERNANDEZ, F. **Automação de Sistemas de Irrigação**. Disponível em: <http://www.agr.feis.unesp.br/curso2.htm> Acesso em: 12 abr. 2013.

TESTEZLAF, Roberto(a). **Sistemas automáticos de controle de irrigação**. Disponível em: <http://webensino.unicamp.br/disciplinas/FA876-055506/apoio/14/Automa.pdf> Acesso em: 23 mai. 2013.

TESTEZLAF, Roberto(b); MATSURA, Edson Eiji; CARDOSO, João Luiz. **Importância da irrigação no desenvolvimento do agronegócio.** Campinas – SP: Universidade Estadual de Campinas, 2002.

TESTEZLAF, Roberto(c); **Irrigação: métodos, sistemas e aplicações**. Campinas – SP: Universidade Estadual de Campinas, 2011.

GLOSSÁRIO

- *Baud rate*: medida de velocidade de transmissão de informação entre dispositivos.
- *Byte*: conjunto formado por oito *bits*.
- *Chip*: circuito integrado.
- *Clock*: velocidade de processamento.
- *Displays*: mostradores de caracteres em cristal líquido.
- *Drivers: softwares* que permitem a comunicação entre o sistema operacional e os dispositivos.
- *Ethernet*: é uma arquitetura de interconexão para redes locais - Rede de Área Local (LAN) - baseada no envio e recebimento de pacotes.
- *Firmware*: em eletrônica e computação, é o conjunto de instruções operacionais programadas diretamente no *hardware* de um equipamento eletrônico.
- *Hardware*: no âmbito eletrônico o termo "*hardware*" se aplica à unidade central de processamento, à memória e aos dispositivos de entrada e saída.
- *Interface*: conexão entre dois dispositivos, no âmbito da computação.
- *Memory*: memória.
- *Modbus*: protocolo de comunicação de dados utilizado em sistemas de automação de industrial.
- *Notebook*: microcomputador portátil.
- *Pull-up*: resistor utilizado em circuitos eletrônicos, com a finalidade de não permitir que as entradas digitais fiquem com estado lógico flutuante.
- *Software*: é uma sequência de instruções escritas para serem interpretadas por um computador com o objetivo de executar tarefas específicas.
- *Software* livre: designa qualquer programa de computador que pode ser executado, copiado, modificado e redistribuído pelos usuários. Os usuários possuem livre acesso ao código-fonte do *software* e fazem alterações conforme as suas necessidades.
- *Smartphone*: é um celular com tecnologias avançadas, o que inclui programas executados em um sistema operacional, equivalente aos computadores.
- *Tablet*: é um tipo de computador portátil, de tamanho pequeno, fina espessura e com tela sensível ao toque.
- *Upload*: enviar dados de um computador local para outro dispositivo.

- *Wireless*: em português significa rede sem fio, e refere-se a uma rede de computadores sem a necessidade do uso de cabos, funciona por meio de equipamentos que usam radiofrequência, comunicação via ondas de rádio, ou comunicação via infravermelho.

APÊNDICES

APÊNDICE A - CÓDIGO DE PROGRAMAÇÃO DO PROTÓTIPO

```
/*
 Modbus over serial line - RTU Slave Arduino Sketch


/*
 * configure_mb_slave(baud, parity, tx_en_pin)
 *
 * sets the communication parameters for of the serial line.
 *
 * baud: baudrate in bps (typical values 9600, 19200... 115200)
 * parity: a single character sets the parity mode (character frame format):
 *         'n' no parity (8N1); 'e' even parity (8E1), 'o' for odd parity (8O1).
 * tx_en_pin: arduino pin number that controls transmision/reception
 *        of an external half-duplex device (e.g. a RS485 interface chip).
 *        0 or 1 disables this function (for a two-device network)
 *        >2 for point-to-multipoint topology (e.g. several arduinos)
 */
void configure_mb_slave(long baud, char parity, char txenpin);
/*
 * update_mb_slave(slave_id, holding_regs_array, number_of_regs)
 *
 * checks if there is any valid request from the modbus master. If there is,
 * performs the action requested
 *
 * slave: slave id (1 to 127)
 * regs: an array with the holding registers. They start at address 1 (master point of view)
 * regs_size: total number of holding registers.
 * returns: 0 if no request from master,
 *             NO_REPLY (-1) if no reply is sent to the master
 *             an exception code (1 to 4) in case of a modbus exceptions
 *             the number of bytes sent as reply ( > 4) if OK.
 */
int update_mb_slave(unsigned char slave, int *regs,
unsigned int regs_size);
/* Modbus RTU common parameters, the Master MUST use the same parameters */
enum {
       MB_SLAVE = 1,            /* modbus slave id */
};
/* slave registers example */
enum {
       MB_TEMPERATURA,
       MB_UMIDADE,
       MB_DISTANCIA,
       MB_VOLUME,
```

```
    MB_AUTOMATICO,
    MB_ALIMENTACAO,
    MB_IRRIGACAO,
    MB_ESVAZIAMENTO,
    MB_VAZAOIRRIGACAO,
    MB_VAZAOESVAZIAMENTO,
    MB_ETO,
    MB_ETC,
    MB_TR,
    MB_LRN,
    MB_LTN,
    MB_VOLUMEIRRIGADO,
    MB_STATUSESVAZIAMENTO,
    MB_STATUSIRRIGACAO,
    MB_REGS                 /* total number of registers on slave */
};
int regs[MB_REGS];          /* this is the slave's modbus data map */
//Variáveis Sensor de Temperatura e Umidade
int TempLida;
int UmiLida;
unsigned long previousMillisDistancia;
long distancia;
long altura;
long volume;
//Variáveis Sensor de Vazão Irrigação
int SensorVazaoIrrigCounter = 0;   // counter for the number of Sensor presses
int SensorState = 0;         // current state of the button
int lastSensorState = 0;     // previous state of the button
unsigned long previousMillisVazaoIrrig;
int SensorVazaoIrrigCounteranterior;
int frequenciairrig;
int CalcIrrig;
//variáveis Sensor de Vazão Esvziamento
int SensorVazaoEsvazCounter = 0;   // counter for the number of Sensor presses
int SensorEsvazState = 0;         // current state of the button
int lastSensorEsvazState = 0;     // previous state of the button
unsigned long previousMillisVazaoEsvaz;
int SensorVazaoEsvazCounteranterior;
int frequenciaEsvaz;
int CalcEsvaz;
//Tabela Evapotranspiração
float ETO10a15[] = {4.6, 3.8, 3, 2.1, 1.3};
float ETO15a20[] = {5.9, 4.9, 3.8, 2.7, 1.6};
float ETO20a25[] = {7.4, 6.1, 4.7, 3.4, 2};
float ETO25a30[] = {9.1, 7.4, 5.8, 4.1, 2.5};
float ETO30a35[] = {10.9, 8.9, 6.9, 5, 3};
//Cálculo Irrigação
float ETC;
float ETO;
float TR;
float LRN;
```

```
float LTN;
float VolumeIrrigacao;
float TempoRega;
float VolumeSaidaIrrigacao;
float TempoCorrente;
unsigned long previousMillisIrrigacao1;
unsigned long previousMillisIrrigacao2;
int Terminopausa;
int NivelBaixo;
void setup()
{

      /* Modbus setup example, the master must use the same COM parameters */
      /* 115200 bps, 8N1, two-device network */
      configure_mb_slave(115200, 'n', 0);

  pinMode(2, OUTPUT); // Esvaziamento
  pinMode(3, OUTPUT); // Reserva
  pinMode(4, INPUT); //Sensor de Vazão Irrigação
  pinMode(5, INPUT); //Sensor de Vazão Esvaziamento
  pinMode(6, INPUT);
  pinMode(7, INPUT);
  pinMode(8, OUTPUT); //Alimentação
  pinMode(9, OUTPUT); //Irrigação
  pinMode(10, INPUT);
  pinMode(11, INPUT);
  pinMode(12, OUTPUT); // define o pino 12 como saida (envia)  TRIGGER
  pinMode(13, INPUT); // define o pino 13 como entrada (recebe) ECHO
 }
void loop()
{
      /* This is all for the Modbus slave */
            update_mb_slave(MB_SLAVE, regs, MB_REGS);

            /* your code goes here */
unsigned long currentMillis = millis(); //Função millis é o tempo corrente após inicialização do sistema.
//Leitura de Dados pelo SCADABR
regs[MB_TEMPERATURA] = TempLida;
regs[MB_UMIDADE] = UmiLida;
regs[MB_DISTANCIA] = distancia;
regs[MB_VOLUME] = altura * 392 * 282 /1000000; //lado x lado x altura em mm3 / 1000000 = litros
regs[MB_VAZAOIRRIGACAO] = CalcIrrig; //Vazão em l/min do sistema de irrigação
regs[MB_VAZAOESVAZIAMENTO] = CalcEsvaz; //Vazão em l/min do sistema de esvaziamento
regs[MB_ETO] = ETO;
regs[MB_ETC] = ETC;
regs[MB_TR] = TR;
regs[MB_LRN] = LRN;
regs[MB_LTN] = LTN;
regs[MB_VOLUMEIRRIGADO]= VolumeSaidaIrrigacao;
```

```
//Escrita de Dados pelo SCADABR
//Temperatura/Umidade
TempLida = analogRead(0);
UmiLida = analogRead(1);
TempLida = map(TempLida, 0, 1023, -15, 60); //Escala de -15ºC a 60ºC, para 0 a 1023
UmiLida = map(UmiLida, 0, 1023, 1, 99); //Escala de 1 a 99% de umidade relativa do ar, para 0 a 1023
//Sensor Ultrasônico
if (currentMillis - previousMillisDistancia > 500)
 {
 //seta o pino 12 com um pulso baixo "LOW" ou desligado ou ainda 0
   digitalWrite(12, LOW);
 // delay de 2 microssegundos
   delayMicroseconds(2);
 //seta o pino 12 com pulso alto "HIGH" ou ligado ou ainda 1
   digitalWrite(12, HIGH);
 //delay de 10 microssegundos
   delayMicroseconds(10);
 //seta o pino 12 com pulso baixo novamente
   digitalWrite(12, LOW);
 //pulseInt lê o tempo entre a chamada e o pino entrar em high
   long duration = pulseIn(13,HIGH);
 //Esse CalcIrrigulo é baseado em s = v . t, lembrando que o tempo vem dobrado
 //porque é o tempo de ida e volta do ultrassom
   distancia = duration /2.9411 / 2 ;
   altura = 262 - distancia; //Valor quando tanque está vazio
previousMillisDistancia = currentMillis;
 }
//Sensor de Vazão Irrigação
SensorState = digitalRead(4);
 if (SensorState != lastSensorState) // Se o estado foi alterado, incrementa o contador
 {
   if (SensorState == HIGH)       // Se o estado atual é HIGH, então o sensor foi de OFF para ON
   {
     SensorVazaoIrrigCounter++; //incrementa o contador
   }
   else {}
 }
 lastSensorState = SensorState; // Salva o estado atual como o último estado, para utilizar no próximo scan
 if (currentMillis - previousMillisVazaoIrrig > 1000) //Atualiza contagem de pulsos a cada 1 segundo para converter diretamente
para Hz
   {
     previousMillisVazaoIrrig = currentMillis; //zera o tempo para próxima temporização
     frequenciairrig = SensorVazaoIrrigCounter - SensorVazaoIrrigCounteranterior; //Frequência = pulsos - pulsosanteriores.
Atualiza a cada segundo
     SensorVazaoIrrigCounter = SensorVazaoIrrigCounteranterior; //Zera para o próximo ciclo
     CalcIrrig = frequenciairrig / 4; //Vazão = Frequência (Hz) / taxa de irrigação
   }



//Sensor de Vazão Esvaziamento
```

```
SensorEsvazState = digitalRead(5);
  if (SensorEsvazState != lastSensorEsvazState) // Se o estado foi alterado, incrementa o contador
  {
    if (SensorEsvazState == HIGH) // Se o estado atual é HIGH, então o sensor foi de OFF para ON
    {
      SensorVazaoEsvazCounter++; //incrementa o contador
    }
    else { }
  }

  lastSensorEsvazState = SensorEsvazState; // Salva o estado atual como o último estado, para utilizar no próximo scan

  if (currentMillis - previousMillisVazaoEsvaz > 1000) //Atualiza contagem de pulsos a cada 1 segundo para converter diretamente para Hz
    {
      previousMillisVazaoEsvaz = currentMillis; //zera o tempo para próxima temporização
      frequenciaEsvaz = SensorVazaoEsvazCounter - SensorVazaoEsvazCounteranterior; //Frequência = pulsos - pulsosanteriores. Atualiza a cada segundo
      SensorVazaoEsvazCounter = SensorVazaoEsvazCounteranterior; //Zera para o próximo ciclo
      CalcEsvaz = frequenciaEsvaz / 4; //Vazão = Frequência (Hz) / taxa de irrigação
    }
//Status Esvaziamento e Irrigação
if (CalcIrrig != 0)
{regs[MB_STATUSIRRIGACAO] = 1;}
else
{regs[MB_STATUSIRRIGACAO] = 0;}

if (CalcEsvaz != 0)
{regs[MB_STATUSESVAZIAMENTO] = 1;}
else
{regs[MB_STATUSESVAZIAMENTO] = 0;}
//AUTOMÁTICO
if (regs[MB_AUTOMATICO]==1)
{
  //CONTROLE DE NÍVEL
  //(permite utilizar a porção entre o máximo e o mínimo do tanque antes de ligar a alimentação)

  if(distancia > 140) //Quando estiver abeixo no mínimo seta a Alimentação
{
  digitalWrite(8,HIGH); //Alimentação
  NivelBaixo = 1; //Nível muito baixo para irrigar
}
  else if (distancia < 81) // Quando estiver acima do máximo reseta a alimentação
{
digitalWrite(8,LOW); //Alimentação
}
//IRRIGAÇÃO
if(distancia < 140) //Quando estiver acima do nível mínimo autoriza irrigação
{
  NivelBaixo = 0; //Nível acima do mínimo, autoriza irrigar.
```

```
    if ((currentMillis - previousMillisIrrigacao1 >= TempoRega) && (Terminopausa == 0)) //Após concluir a irrigação (Terminopausa = 0) o tempo de entre regas é iniciado
      {
       digitalWrite(2,HIGH); // Liga Irrigação
       Terminopausa = 1; //Pausa entre irrigação terminada
       previousMillisIrrigacao2 = currentMillis; //Zera o tempo para inicar o tempo corrente.
      }

      if ((Terminopausa == 1))
      {
      TempoCorrente = currentMillis - previousMillisIrrigacao2;  //Tempo Millis do sistema - o tempo zerado
      digitalWrite(2,HIGH); // Liga Irrigação (TESTE        TESTE  TESTE  TESTE  TESTE  TESTE  TESTE  TESTE  TESTE  TESTE    )

     if ( VolumeSaidaIrrigacao >= VolumeIrrigacao)
      {
      digitalWrite(2,LOW); //Irrigação
      Terminopausa = 0; //Reset pausa entre irrigação
      previousMillisIrrigacao1 = currentMillis; //Zera o tempo para iniciar novo ciclo de pausa
      } }
}
//CÁLCULO DE IRRIGAÇÃO

ETC = 0.85*ETO; //ETc = Kc. Eto - Evapotranspiração do tomate (ETc)mm/dia, para cada estádio de desenvolvimento. Adotado Frutificação Kc=0.85
TR = 0.8*40/ETC; //TR = DRA*Z/ETC - turno da rega (TR) para cada estádio da cultura, dado em dias, e a profundidade efetiva do sistema radicular (Z) em cm, disponibilidade real de água no solo (DRA), em função de sua textura,. Adotado Frutificação Z=40cm e DRA = 0,8
LRN = TR*ETC; //LRN = TR . Etc - Lâmina de água real necessária por irrigação (LRN)
LTN = LRN*100/70; // Correção do valor de LRN em função da eficiência do sistema de irrigação , de modo a obter a lâmina de água total necessária (LTN). Adotada Asperção convencional = eficiÊncia 70%
VolumeIrrigacao = 1 * LTN/1000 * 1000; // Volume de irrigação  = Área da Base x Altura. Adotado 1m² e LTN mm/1000 para converser em m. nultiplicado por 1000 para converter para Litros.
TempoRega = TR * 1000; //Multiplicado tempo de rega REAL, em dias, por 1000ms para enriquecer apresentação
VolumeSaidaIrrigacao = CalcIrrig * TempoCorrente/60000; // Volume = Vazão(l/min) x Tempo(1min/60000ms)
//ETO -  Evapotranspiração de referência (ETo), em mm/dia, em função da temperatura e umidade relativa do ar.
    if (((TempLida>=10) && (TempLida<=15)) && ((UmiLida>=40) && (UmiLida<=50))) {ETO = ETO10a15[0];}
else if (((TempLida>=10) && (TempLida<=15)) && ((UmiLida>=51) && (UmiLida<=60))) {ETO = ETO10a15[1];}
else if (((TempLida>=10) && (TempLida<=15)) && ((UmiLida>=61) && (UmiLida<=70))) {ETO = ETO10a15[2];}
else if (((TempLida>=10) && (TempLida<=15)) && ((UmiLida>=71) && (UmiLida<=80))) {ETO = ETO10a15[3];}
else if (((TempLida>=10) && (TempLida<=15)) && ((UmiLida>=81) && (UmiLida<=90))) {ETO = ETO10a15[4];}
else if (((TempLida>=16) && (TempLida<=20)) && ((UmiLida>=40) && (UmiLida<=50))) {ETO = ETO15a20[0];}
else if (((TempLida>=16) && (TempLida<=20)) && ((UmiLida>=51) && (UmiLida<=60))) {ETO = ETO15a20[1];}
else if (((TempLida>=16) && (TempLida<=20)) && ((UmiLida>=61) && (UmiLida<=70))) {ETO = ETO15a20[2];}
else if (((TempLida>=16) && (TempLida<=20)) && ((UmiLida>=71) && (UmiLida<=80))) {ETO = ETO15a20[3];}
else if (((TempLida>=16) && (TempLida<=20)) && ((UmiLida>=81) && (UmiLida<=90))) {ETO = ETO15a20[4];}
else if (((TempLida>=21) && (TempLida<=25)) && ((UmiLida>=40) && (UmiLida<=50))) {ETO = ETO20a25[0];}
else if (((TempLida>=21) && (TempLida<=25)) && ((UmiLida>=51) && (UmiLida<=60))) {ETO = ETO20a25[1];}
else if (((TempLida>=21) && (TempLida<=25)) && ((UmiLida>=61) && (UmiLida<=70))) {ETO = ETO20a25[2];}
else if (((TempLida>=21) && (TempLida<=25)) && ((UmiLida>=71) && (UmiLida<=80))) {ETO = ETO20a25[3];}
else if (((TempLida>=21) && (TempLida<=25)) && ((UmiLida>=81) && (UmiLida<=90))) {ETO = ETO20a25[4];}
```

```
else if (((TempLida>=26) && (TempLida<=30)) && ((UmiLida>=40) && (UmiLida<=50))) {ETO = ETO25a30[0];}
else if (((TempLida>=26) && (TempLida<=30)) && ((UmiLida>=51) && (UmiLida<=60))) {ETO = ETO25a30[1];}
else if (((TempLida>=26) && (TempLida<=30)) && ((UmiLida>=61) && (UmiLida<=70))) {ETO = ETO25a30[2];}
else if (((TempLida>=26) && (TempLida<=30)) && ((UmiLida>=71) && (UmiLida<=80))) {ETO = ETO25a30[3];}
else if (((TempLida>=26) && (TempLida<=30)) && ((UmiLida>=81) && (UmiLida<=90))) {ETO = ETO25a30[4];}
else if (((TempLida>=31) && (TempLida<=35)) && ((UmiLida>=40) && (UmiLida<=50))) {ETO = ETO30a35[0];}
else if (((TempLida>=31) && (TempLida<=35)) && ((UmiLida>=51) && (UmiLida<=60))) {ETO = ETO30a35[1];}
else if (((TempLida>=31) && (TempLida<=35)) && ((UmiLida>=61) && (UmiLida<=70))) {ETO = ETO30a35[2];}
else if (((TempLida>=31) && (TempLida<=35)) && ((UmiLida>=71) && (UmiLida<=80))) {ETO = ETO30a35[3];}
else if (((TempLida>=31) && (TempLida<=35)) && ((UmiLida>=81) && (UmiLida<=90))) {ETO = ETO30a35[4];}
  }
//Modo MANUAL
else
{
digitalWrite(8,regs[MB_ALIMENTACAO]); //Alimentação
digitalWrite(2,regs[MB_IRRIGACAO]); //Irrigação
digitalWrite(9,regs[MB_ESVAZIAMENTO]); //Esvaziamento
}

}

/**************************************************************************
 * BEGIN MODBUS RTU SLAVE FUNCTIONS
 **************************************************************************/

/* global variables */
unsigned int Txenpin = 0;       /* Enable transmission pin, used on RS485 networks */
/* enum of supported modbus function codes. If you implement a new one, put its function code here ! */
enum {
      FC_READ_REGS  = 0x03,   //Read contiguous block of holding register
      FC_WRITE_REG  = 0x06,   //Write single holding register
      FC_WRITE_REGS = 0x10    //Write block of contiguous registers
};
/* supported functions. If you implement a new one, put its function code into this array! */
const unsigned char fsupported[] = { FC_READ_REGS, FC_WRITE_REG, FC_WRITE_REGS };
/* constants */
enum {
      MAX_READ_REGS = 0x7D,
      MAX_WRITE_REGS = 0x7B,
      MAX_MESSAGE_LENGTH = 256
};
enum {
      RESPONSE_SIZE = 6,
      EXCEPTION_SIZE = 3,
      CHECKSUM_SIZE = 2
};
/* exceptions code */
enum {
      NO_REPLY = -1,
      EXC_FUNC_CODE = 1,
      EXC_ADDR_RANGE = 2,
```

```
        EXC_REGS_QUANT = 3,
        EXC_EXECUTE = 4
};
/* positions inside the query/response array */
enum {
        SLAVE = 0,
        FUNC,
        START_H,
        START_L,
        REGS_H,
        REGS_L,
        BYTE_CNT
};
/*
CRC
 INPUTS:
            buf   ->  Array containing message to be sent to controller.
            start ->  Start of loop in crc counter, usually 0.
            cnt   ->  Amount of bytes in message being sent to controller/
 OUTPUTS:
            temp  ->  Returns crc byte for message.
 COMMENTS:
            This routine CalcIrrigulates the crc high and low byte of a message.
            Note that this crc is only used for Modbus, not Modbus+ etc.
 ************************************************************************/

unsigned int crc(unsigned char *buf, unsigned char start,
unsigned char cnt)
{
        unsigned char i, j;
        unsigned temp, temp2, flag;
        temp = 0xFFFF;
        for (i = start; i < cnt; i++) {
            temp = temp ^ buf[i];

            for (j = 1; j <= 8; j++) {
                flag = temp & 0x0001;
                temp = temp >> 1;
                if (flag)
                    temp = temp ^ 0xA001;
            }
        }
        /* Reverse byte order. */
        temp2 = temp >> 8;
        temp = (temp << 8) | temp2;
        temp &= 0xFFFF;
        return (temp);
}

/************************************************************************
 *
```

```
 *          The following functions construct the required query into
 *          a modbus query packet.
 *
 *****************************************************************************/
/*
 * Start of the packet of a read_holding_register response
 */
void build_read_packet(unsigned char slave, unsigned char function,
unsigned char count, unsigned char *packet)
{
        packet[SLAVE] = slave;
        packet[FUNC] = function;
        packet[2] = count * 2;
}
/*
 * Start of the packet of a preset_multiple_register response
 */
void build_write_packet(unsigned char slave, unsigned char function,
unsigned int start_addr,
unsigned char count,
unsigned char *packet)
{
        packet[SLAVE] = slave;
        packet[FUNC] = function;
        packet[START_H] = start_addr >> 8;
        packet[START_L] = start_addr & 0x00ff;
        packet[REGS_H] = 0x00;
        packet[REGS_L] = count;
}
/*
 * Start of the packet of a write_single_register response
 */
void build_write_single_packet(unsigned char slave, unsigned char function,
        unsigned int write_addr, unsigned int reg_val, unsigned char* packet)
{
        packet[SLAVE] = slave;
        packet[FUNC] = function;
        packet[START_H] = write_addr >> 8;
        packet[START_L] = write_addr & 0x00ff;
        packet[REGS_H] = reg_val >> 8;
        packet[REGS_L] = reg_val & 0x00ff;
}
/*
 * Start of the packet of an exception response
 */
void build_error_packet(unsigned char slave, unsigned char function,
unsigned char exception, unsigned char *packet)
{
        packet[SLAVE] = slave;
        packet[FUNC] = function + 0x80;
        packet[2] = exception;
```

```
}
/*****************************************************************************
 *
 * modbus_query( packet, length)
 *
 * Function to add a checksum to the end of a packet.
 * Please note that the packet array must be at least 2 fields longer than
 * string_length.
 *****************************************************************************/

void modbus_reply(unsigned char *packet, unsigned char string_length)
{
	int temp_crc;

	temp_crc = crc(packet, 0, string_length);
	packet[string_length] = temp_crc >> 8;
	string_length++;
	packet[string_length] = temp_crc & 0x00FF;
}
/*****************************************************************************
 *
 * send_reply( query_string, query_length )
 *
 * Function to send a reply to a modbus master.
 * Returns: total number of characters sent
 *****************************************************************************/
int send_reply(unsigned char *query, unsigned char string_length)
{
	unsigned char i;

	if (Txenpin > 1) { // set MAX485 to speak mode
		UCSR0A=UCSR0A |(1 << TXC0);
		digitalWrite( Txenpin, HIGH);
		delay(1);
	}
	modbus_reply(query, string_length);
	string_length += 2;

	for (i = 0; i < string_length; i++) {
		Serial.write(byte(query[i]));
	}

	if (Txenpin > 1) {// set MAX485 to listen mode
		while (!(UCSR0A & (1 << TXC0)));
		digitalWrite( Txenpin, LOW);
	}

	return i;			/* it does not mean that the write was succesful, though */
}
/*****************************************************************************
 *
```

```
 *            receive_request( array_for_data )
 *
 * Function to monitor for a request from the modbus master.
 *
 * Returns:            Total number of characters received if OK
 * 0 if there is no request
 * A negative error code on failure
 ***************************************************************************/
int receive_request(unsigned char *received_string)
{
      int bytes_received = 0;

      /* FIXME: does Serial.available wait 1.5T or 3.5T before exiting the loop? */
      while (Serial.available()) {
            received_string[bytes_received] = Serial.read();
            bytes_received++;
            if (bytes_received >= MAX_MESSAGE_LENGTH)
                  return NO_REPLY;        /* port error */
      }

      return (bytes_received);
}
/***************************************************************************
 *
 *            modbus_request(slave_id, request_data_array)
 *
 * Function to the correct request is returned and that the checksum
 * is correct.
 *
 * Returns:            string_length if OK
 *                     0 if failed
 *                     Less than 0 for exception errors
 *
 *            Note: All functions used for sending or receiving data via
 *                  modbus return these return values.
 *
 ***************************************************************************/

int modbus_request(unsigned char slave, unsigned char *data)
{
      int response_length;
      unsigned int crc_CalcIrrig = 0;
      unsigned int crc_received = 0;
      unsigned char recv_crc_hi;
      unsigned char recv_crc_lo;

      response_length = receive_request(data);

      if (response_length > 0) {
            crc_CalcIrrig = crc(data, 0, response_length - 2);
            recv_crc_hi = (unsigned) data[response_length - 2];
```

```
            recv_crc_lo = (unsigned) data[response_length - 1];
            crc_received = data[response_length - 2];
            crc_received = (unsigned) crc_received << 8;
            crc_received =
                  crc_received | (unsigned) data[response_length - 1];

            /********** check CRC of response ***********/
            if (crc_CalcIrrig != crc_received) {
                  return NO_REPLY;
            }

            /* check for slave id */
            if (slave != data[SLAVE]) {
                  return NO_REPLY;
            }
      }
      return (response_length);
}
/*****************************************************************************
 *
 *          validate_request(request_data_array, request_length, available_regs)
 *
 * Function to check that the request can be processed by the slave.
 *
 * Returns:          0 if OK
 *                   A negative exception code on error
 *
 *****************************************************************************/
int validate_request(unsigned char *data, unsigned char length,
unsigned int regs_size)
{
      int i, fcnt = 0;
      unsigned int regs_num = 0;
      unsigned int start_addr = 0;
      unsigned char max_regs_num;

      /* check function code */
      for (i = 0; i < sizeof(fsupported); i++) {
            if (fsupported[i] == data[FUNC]) {
                  fcnt = 1;
                  break;
            }
      }
      if (0 == fcnt)
            return EXC_FUNC_CODE;

      if (FC_WRITE_REG == data[FUNC]) {
            /* For function write single reg, this is the target reg.*/
            regs_num = ((int) data[START_H] << 8) + (int) data[START_L];
            if (regs_num >= regs_size)
                  return EXC_ADDR_RANGE;
```

```
            return 0;
        }

        /* For functions read/write regs, this is the range. */
        regs_num = ((int) data[REGS_H] << 8) + (int) data[REGS_L];

        /* check quantity of registers */
        if (FC_READ_REGS == data[FUNC])
            max_regs_num = MAX_READ_REGS;
        else if (FC_WRITE_REGS == data[FUNC])
            max_regs_num = MAX_WRITE_REGS;

        if ((regs_num < 1) || (regs_num > max_regs_num))
            return EXC_REGS_QUANT;

        /* check registers range, start address is 0 */
        start_addr = ((int) data[START_H] << 8) + (int) data[START_L];
        if ((start_addr + regs_num) > regs_size)
            return EXC_ADDR_RANGE;

        return 0;                       /* OK, no exception */
}
/**************************************************************************
 *
 *          write_regs(first_register, data_array, registers_array)
 *
 *          writes into the slave's holding registers the data in query,
 * starting at start_addr.
 *
 * Returns:   the number of registers written
 **************************************************************************/
int write_regs(unsigned int start_addr, unsigned char *query, int *regs)
{
        int temp;
        unsigned int i;

        for (i = 0; i < query[REGS_L]; i++) {
            /* shift reg hi_byte to temp */
            temp = (int) query[(BYTE_CNT + 1) + i * 2] << 8;
            /* OR with lo_byte          */
            temp = temp | (int) query[(BYTE_CNT + 2) + i * 2];

            regs[start_addr + i] = temp;
        }
        return i;
}
/**************************************************************************
 *
 *          preset_multiple_registers(slave_id, first_register, number_of_registers,
 * data_array, registers_array)
 *
```

```
 *          Write the data from an array into the holding registers of the slave.
 *
 ***************************************************************************/

int preset_multiple_registers(unsigned char slave,
unsigned int start_addr,
unsigned char count,
unsigned char *query,
int *regs)
{
        unsigned char function = FC_WRITE_REGS;             /* Preset Multiple Registers */
        int status = 0;
        unsigned char packet[RESPONSE_SIZE + CHECKSUM_SIZE];

        build_write_packet(slave, function, start_addr, count, packet);

        if (write_regs(start_addr, query, regs)) {
                status = send_reply(packet, RESPONSE_SIZE);
        }

        return (status);
}

/***************************************************************************
 *
 * write_single_register(slave_id, write_addr, data_array, registers_array)
 *
 * Write a single int val into a single holding register of the slave.
 *
 ***************************************************************************/

int write_single_register(unsigned char slave,
        unsigned int write_addr, unsigned char *query, int *regs)
{
        unsigned char function = FC_WRITE_REG; /* Function: Write Single Register */
        int status = 0;
        unsigned int reg_val;
        unsigned char packet[RESPONSE_SIZE + CHECKSUM_SIZE];

        reg_val = query[REGS_H] << 8 | query[REGS_L];
        build_write_single_packet(slave, function, write_addr, reg_val, packet);
        regs[write_addr] = (int) reg_val;
/*
        written.start_addr=write_addr;
        written.num_regs=1;
*/
        status = send_reply(packet, RESPONSE_SIZE);

        return (status);
}
```

```
/********************************************************************
 *
 *            read_holding_registers(slave_id, first_register, number_of_registers,
 * registers_array)
 *
 * reads the slave's holdings registers and sends them to the Modbus master
 *
 ********************************************************************/

int read_holding_registers(unsigned char slave, unsigned int start_addr,

unsigned char reg_count, int *regs)
{
      unsigned char function = 0x03;      /* Function 03: Read Holding Registers */
      int packet_size = 3;
      int status;
      unsigned int i;
      unsigned char packet[MAX_MESSAGE_LENGTH];

      build_read_packet(slave, function, reg_count, packet);

      for (i = start_addr; i < (start_addr + (unsigned int) reg_count);
                  i++) {
                   packet[packet_size] = regs[i] >> 8;
            packet_size++;
            packet[packet_size] = regs[i] & 0x00FF;
            packet_size++;
      }

      status = send_reply(packet, packet_size);

      return (status);
}
void configure_mb_slave(long baud, char parity, char txenpin)
{
      Serial.begin(baud);

      switch (parity) {
      case 'e': // 8E1
            UCSR0C |= ((1<<UPM01) | (1<<UCSZ01) | (1<<UCSZ00));
            //      UCSR0C &= ~((1<<UPM00) | (1<<UCSZ02) | (1<<USBS0));
            break;
      case 'o': // 8O1
            UCSR0C |= ((1<<UPM01) | (1<<UPM00) | (1<<UCSZ01) | (1<<UCSZ00));
            //      UCSR0C &= ~((1<<UCSZ02) | (1<<USBS0));
            break;
      case 'n': // 8N1
            UCSR0C |= ((1<<UCSZ01) | (1<<UCSZ00));
            //      UCSR0C &= ~((1<<UPM01) | (1<<UPM00) | (1<<UCSZ02) | (1<<USBS0));
            break;
      default:
```

```
            break;
        }

        if (txenpin > 1) { // pin 0 & pin 1 are reserved for RX/TX
            Txenpin = txenpin; /* set global variable */
            pinMode(Txenpin, OUTPUT);
            digitalWrite(Txenpin, LOW);
        }

        return;
}

/*
 * update_mb_slave(slave_id, holding_regs_array, number_of_regs)
 *
 * checks if there is any valid request from the modbus master. If there is,
 * performs the action requested
 */

unsigned long Nowdt = 0;
unsigned int lastBytesReceived;
const unsigned long T35 = 5;
int update_mb_slave(unsigned char slave, int *regs,
unsigned int regs_size)
{
        unsigned char query[MAX_MESSAGE_LENGTH];
        unsigned char errpacket[EXCEPTION_SIZE + CHECKSUM_SIZE];
        unsigned int start_addr;
        int exception;
        int length = Serial.available();
        unsigned long now = millis();

        if (length == 0) {
            lastBytesReceived = 0;
            return 0;
        }

        if (lastBytesReceived != length) {
            lastBytesReceived = length;
            Nowdt = now + T35;
            return 0;
        }
        if (now < Nowdt)
            return 0;

        lastBytesReceived = 0;

        length = modbus_request(slave, query);
        if (length < 1)
            return length;
         exception = validate_request(query, length, regs_size);
```

```
    if (exception) {
            build_error_packet(slave, query[FUNC], exception,
            errpacket);
            send_reply(errpacket, EXCEPTION_SIZE);
            return (exception);
    }


    start_addr = ((int) query[START_H] << 8) +
          (int) query[START_L];
    switch (query[FUNC]) {
        case FC_READ_REGS:
            return read_holding_registers(slave,
            start_addr,
            query[REGS_L],
            regs);
        break;
        case FC_WRITE_REGS:
            return preset_multiple_registers(slave,
            start_addr,
            query[REGS_L],
            query,
            regs);
        break;
        case FC_WRITE_REG:
            write_single_register(slave,
            start_addr,
            query,
            regs);
        break;
    }
}
```

ANEXOS

ANEXO A – EMBRAPA - CULTIVO DE TOMATE PARA INDUSTRIALIZAÇÃO

IRRIGAÇÃO

Da semeadura à emergência das plântulas, as irrigações devem ser leves e frequentes, de modo a manter os primeiros 10 cm do solo sempre umedecidos. Nessa fase, o turno de rega deve ser de 1 a 2 dias, dependendo do tipo de solo e das condições climáticas. Em solos arenosos e/ou em regiões de temperatura elevada e de baixa umidade relativa do ar, o turno de rega deve ser diário. Irrigações frequentes também são recomendadas por ocasião do transplante. Neste caso, deve-se irrigar preferencialmente pela manhã, quando a temperatura é mais amena e as plantas estão geralmente túrgidas.

Dependendo do tipo de solo e do clima da região, as irrigações devem ser paralisadas 20 a 30 dias antes do início da colheita, quando as plantas apresentarem cerca de 20% de frutos maduros. Essa medida visa concentrar a maturação de frutos e aumentar a concentração de sólidos solúveis. Entretanto, em termos de produção de frutos, maiores produtividades podem ser obtidas irrigando-se até a ocasião em que cerca de 50% dos frutos estiverem maduros.

Dentre os vários critérios existentes para o manejo da irrigação, nenhum pode ser considerado padrão nem indicado para todas as situações. Métodos clássicos que permitem um controle bastante criterioso da irrigação – como o do balanço hídrico e da tensão de água no solo –, baseiam-se no conhecimento das características fisico-hídricas do solo, das necessidades específicas da cultura e de fatores climáticos relacionados à evapotranspiração. Dependem ainda do uso de equipamentos para o monitoramento da umidade do solo (tensiômetros, blocos de resistência elétrica, etc.) ou de equipamentos para a estimativa da evapotranspiração (tanque Classe A, termômetros, higrômetros, etc.). Essas informações e equipamentos, além de não estarem, em geral, ao alcance do irrigante, exigem conhecimentos técnicos específicos para seu manuseio.

Um método aproximado e que dispensa o uso de equipamentos é o do turno de rega. A seguir é apresentada uma sequência de passos que permitem a determinação da frequência e da lâmina de água a ser aplicada por irrigação, em

cada estádio de desenvolvimento do tomateiro. Simultaneamente, é apresentado um exemplo de sua utilização, considerando-se a seguinte situação:

- Solo: Latossolo Vermelho-Escuro, textura argilosa;
- Clima: temperatura de 20,5 °C, umidade relativa de 54% (média para o mês de julho - no Planalto Central);
- Estádio: frutificação;
- Profundidade efetiva do sistema radicular: 40 cm;
- Eficiência de irrigação: 70 % (aspersão convencional).

Passo 1: Utilizando a Tabela 1, determinar a evapotranspiração de referência (ETo) em função de dados históricos de temperatura e média mensal da umidade relativa do ar, para os meses em que o tomateiro será cultivado. Esses dados podem ser obtidos no Serviço de Extensão Rural (Emater).
Pela Tabela 1, para a temperatura de 20,5 ºC e 54% de umidade relativa, tem-se que a ETo é de 6,1 mm/dia.

Passo 2: Determinar a evapotranspiração do tomate (ETc), para cada estádio de desenvolvimento, pela seguinte equação:
ETc = Kc. Eto
em que ETc é dado em mm/dia, e Kc é obtido na Tabela 2. Na fase de frutificação, Kc é igual a 0,85.
Assim:
ETc = 0,85 x 6,1 = 5,2 mm/dia

Passo 3: Determinar a disponibilidade real de água no solo (DRA), em função de sua textura, através da Tabela 3.
Para solo argiloso de cerrado, tem-se que DRA é de 0,8 mm/cm3.

Passo 4: Determinar o turno da rega (TR) para cada estádio da cultura, sendo:

$$TR = \frac{0,8 \text{ x } 40}{5,2} = 6 \text{ dias}$$

Passo 5: Determinar a lâmina de água real necessária por irrigação (LRN), pela seguinte expressão:

LRN = TR . Etc
em que LRN é dada em mm.

No exemplo em questão, tem-se que:

LRN = 6 x 5,2 = 31,2 mm

Passo 6: Corrigir o valor de LRN em função da eficiência do sistema de irrigação (Ei), de modo a obter a lâmina de água total necessária (LTN), fazendo:

$$LTN = \frac{100 \cdot LRN}{Ei}$$

onde LRN é dada em mm e Ei em % (Ex.: pivô-central: 80-90 %; aspersão convencional: 60% - 70%).
A lâmina de água a ser aplicada no estádio de frutificação será:

$$LTN = \frac{100 \text{ x } 31,2}{70} = 44,6\text{mm}$$

De modo geral, as irrigações na região do Cerrado são feitas por aspersão, utilizando-se o pivô-central. No Vale do São Francisco, usa-se a irrigação por sulco, que consiste na distribuição de água por meio de sulcos paralelos às fileiras de plantio. A água é geralmente conduzida por um canal principal, de onde é derivada para os sulcos, utilizando-se tubos plásticos denominados de sifões, com diâmetro de uma a duas polegadas. A distribuição da água pode ser feita também por tubos janelados, que possuem diversas aberturas reguláveis que permitem o controle da quantidade de água aplicada em cada sulco de irrigação.

O comprimento dos sulcos e sua declividade são determinados em função da textura do solo.

Os sulcos devem ter 15 a 20 cm de profundidade e 25 a 30 cm de largura. As maiores dimensões são utilizadas para solos de baixa velocidade de infiltração.

Tabela 1. Evapotranspiração de referência (ETo), em mm/dia, em função da temperatura e umidade relativa média mensal do ar.

Temperatura (°C)	Umidade relativa (%)				
	40 a 50	50 a 60	60 a 70	70 a 80	80 a 90
10 a 15	4,6	3,8	3,0	2,1	1,3
15 a 20	5,9	4, 9	3,8	2,7	1,6
20 a 25	7,4	6,1	4,7	3,4	2,0
25 a 30	9,1	7,4	5,8	4,1	2,5
30 a 35	10,9	8,9	6,9	5,0	3,0

Fonte: Obtido a partir da equação de Ivanov (JENSEN, 1973).

Tabela 2. Coeficiente de cultura (Kc), profundidade efetiva média do sistema radicular (Z) e problemas associados à irrigação inadequada nos diferentes estádios de desenvolvimento da cultura do tomateiro.

Estádio de desenvolvimento(10)	Duração (dias)	Kc	Z (2) (cm)	Problemas associados à irrigação
Inicial	10-20	0,55	10	Irrigações deficitárias ou em excesso reduzem o 'estande'
Vegetativo	30	0,65	20 a 30	Irrigações abundantes favorecem o crescimento excessivo e a maior incidência de doenças
Frutificação	40	0,85	40	A falta de água reduz o peso e o número de frutos. O excesso favorece a maior incidência de doenças
Maturação	30	0,65	40	Irrigações neste estádio prejudicam a qualidade dos frutos e reduzem o Brix

(1) Inicial: do plantio até dois pares de folhas ou pegamento de mudas. Vegetativo: até o início do florescimento. Frutificação: até o início da maturação de frutos. Maturação: até a colheita. **(2)** Avaliar de preferência no próprio local de cultivo.

Fonte: Adaptado de Marouelli *et al*. (1996).

Tabela 3. Disponibilidade real média de água no solo para tomateiro, para diferentes tipos de solos.

Textura	Classe textural (exemplos)	Disponibilidade real (mm/cm3)
Grossa	Areia, areia franca Franco-arenoso	0,5
Média	Franco, franco-siltoso Franco-argilo-arenoso Muito argiloso	0,8
Fina	Argila, argila-siltosa Franco-argilo-siltoso	1,0

Obs.: Em geral, mesmo os solos de cerrado com textura fina apresentam disponibilidade real de cerca de 0,8 mm/cm3.

Fonte: Adaptado de Marouelli *et al*. (1996).