Universidade Federal de Campina Grande

Centro de Engenharia Elétrica e Informática

Curso de Graduação em Engenharia Elétrica

CÂNDIDO DA NÓBREGA FERREIRA NETO

SISTEMA DE CONTROLE DE IRRIGAÇÃO RESIDENCIAL

Campina Grande, Paraíba
Maio de 2015

CÂNDIDO DA NÓBREGA FERREIRA NETO

SISTEMA DE CONTROLE DE IRRIGAÇÃO RESIDENCIAL

Trabalho de Conclusão de Curso submetido à Unidade Acadêmica de Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Instrumentação Eletrônica

Orientador:
Professor Jaidilson Jó da Silva, D. Sc.

Campina Grande, Paraíba
Maio de 2015

CÂNDIDO DA NÓBREGA FERREIRA NETO

SISTEMA DE CONTROLE DE IRRIGAÇÃO RESIDENCIAL

Trabalho de Conclusão de Curso submetido à Unidade Acadêmica de Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Instrumentação Eletrônica

Aprovado em / /

Professor Avaliador
Universidade Federal de Campina Grande
Avaliador

Professor Jaidilson Jó da Silva, D. Sc.
Universidade Federal de Campina Grande
Orientador, UFCG

Aos meus pais, pela compreensão, paciência e apoio dedicados a mim por todos esses anos.

AGRADECIMENTOS

Agradeço primeiramente a Deus, por me dar a sabedoria de reconhecer que nada é possível sem Ele, e pela força dada a mim para que fosse possível concluir essa jornada.

Agradeço à minha família pelo apoio, paciência e carinho, sempre presente em todos os momentos. Em especial aos meus pais, Cândido e Socorro, que nunca mediram esforços para me oferecer uma educação de qualidade, fornecendo-me totais condições e apoio para realizar meus sonhos, e à minha irmã Carolina, pela fé sempre depositada no meu sucesso e transmissão de confiança e amor incondicional.

Agradeço aos meus amigos Saulo, Dario, Caio, Leonardo, Coriolano, Jonas, Daniel, Tarcisio, Hugo, Herbet, Priscila, e tantos outros, pela compreensão e valorosos conselhos, momentos de estudos e de lazer, e por toda contribuição que tiveram para o meu sucesso.

Agradeço à minha namorada Isabelle, pelo apoio em todos os momentos difíceis que enfrentei nesta reta final, amor, sincera torcida e palavras de incentivo, fundamentais para que alcançasse meus objetivos.

Presto sentimento de gratidão ao professor e orientador Jaidilson Jó da Silva, pela contribuição no desenvolvimento deste trabalho.

Agradeço ao prof. Adriano e ao técnico Lúcio Hélio, bem como a equipe do Laboratório de Prototipagem e Instrumentação Eletrônica da UAF/UFCG.

Aqueles, que não por menor importância, não foram citados, mas também tiveram grande contribuição na realização do sonho de adquirir o título de Bacharel em Engenharia Elétrica.

RESUMO

Segundo relatório da Unesco (Organização das Nações Unidas para a Educação, a Ciência e a Cultura) divulgado em março de 2015, estima-se que as reservas hídricas do mundo podem encolher 40% até 2030. O documento afirma que há no mundo água suficiente para suprir as necessidades de crescimento do consumo, contanto que haja uma mudança no uso do recurso, pois o caminho em que o homem se encontra levará a uma crise hídrica. O projeto de automação residencial apresentado propõe uma solução em pequena escala para esse problema, consistindo de um sistema de controle automático de irrigação para plantas. Será utilizado como unidade de processamento central a plataforma de desenvolvimento Arduino, além de contar com comunicação *bluetooth*, sensoriamento, aquisição de dados, e atuadores. Outra característica é o baixo custo do projeto, bem como seu retorno financeiro a curto prazo, além da sua possibilidade de expansão para soluções de maior escala.

Palavras-chave: Irrigação, automação residencial, Arduino, *bluetooth*, aquisição de dados.

ABSTRACT

According to a report from UNESCO (United Nations Educational, Scientific and Cultural Organization) released in March 2015, it is estimated that the water reserves in the world can shrink up to 40% by 2030. The document states that in the world there is enough water to meet the growing needs of the consumer, provided there is a change in the use of this resource, because the way in which man find himself will lead to a water crisis. The home automation project presented here proposes a small-scale solution to this problem, consisting of an automatic control system for plant irrigation. The central processing unit will be the Arduino development board, besides having *bluetooth* communication, sensing, data acquisition and actuators. Another characteristic is the low cost of the project, as well as its short-term financial returns and its ability to expand to larger scale solutions.

Keywords: Irrigation, home automation, Arduino, *bluetooth*, data acquisition.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Dados presentes em relatório da Unesco divulgado em março de 2015 apontam para uma eventual crise hídrica caso não aconteça uma mudança drástica no uso, gerenciamento e compartilhamento deste recurso, cujo consumo vem em crescente nas últimas décadas e poderá levar à indisponibilidade deste recurso no futuro [1].

Antes cenário de filmes de ficção, residências completamente automatizadas são parte do contexto atual da maioria das grandes cidades [2]. Tendo surgido no século 20, o conceito de "Casa do Futuro" é algo real [3], tornado possível graças ao advento dos sistemas de automação residencial, que permite que seu usuário monitore a segurança do seu lar, controle à distância variáveis como luz, som e temperatura ambiente, além de verificar as tarefas realizadas automaticamente durante o dia, tal qual a irrigação do seu jardim [4].

Segundo dados da Associação Brasileira de Automação Residencial (Aureside), o crescimento projetado do mercado de automação residencial no Brasil é de 30% ao ano, com faturamento superior a R$ 3,7 bilhões em 2014 [5]. Esta constante expansão, que já perdura por anos, se deve principalmente à redução nos custos das tecnologias empregadas nesses sistemas e ao aumento do número de empresas e serviços especializados na área [6].

Outro fator que incentiva o uso de automação residencial e a implementação de projetos neste segmento é o desenvolvimento tecnológico do setor. O avanço e barateamento dos microprocessadores e periféricos como sensores e atuadores, e principalmente o desenvolvimento das tecnologias de comunicações, como internet e telefonia móvel. Todos estes itens popularizaram o conceito de automação residencial, antes exclusivo a um mercado de luxo e mais exclusivo, tornando possível que qualquer um possa ter o seu sistema, ou até mesmo desenvolver um projeto próprio.

Este trabalho apresenta como proposta a implementação de um sistema de irrigação automática residencial, um segmento no ramo de automação residencial, utilizando como unidade central de processamento a plataforma Arduino.

1.1 MOTIVAÇÃO

De toda a água doce consumida no Brasil, 80% é destinada à agropecuária e 13% às residências, sendo 11% consumida na área urbana. Os estudos indicam que, enquanto o volume de água destinado à irrigação em larga escala é bem maior que o consumo urbano, nas cidades o custo se eleva, visto que a água utilizada na manutenção de jardins é geralmente potável [7].

Nesse sentido, em que pese o consumo maior na agropecuária em relação ao consumo residencial urbano, é fato que, economicamente, o segundo merece igual atenção, uma vez que é qualitativamente mais oneroso, em função do processo difícil e de elevado custo de tratamento da água nas ETAs (Estações de Tratamento de Água). O consumo de água potável nas áreas jardinadas alcança 7% dos gastos totais de uma residência, podendo chegar até mesmo a 50% deste total em dias mais quentes[8].

Esse projeto apresenta como proposta para a economia dos gastos de água um sistema de controle de irrigação residencial, com monitoramento constante das variáveis físicas que envolvem o ambiente onde a planta se localiza (temperatura ambiente, umidade relativa do ar e umidade do solo), realizando o controle automático da sua irrigação, evitando desperdícios e mantendo a planta saudável.

1.2 OBJETIVOS

O objetivo principal deste trabalho é desenvolver um protótipo do sistema de controle de irrigação residencial, capaz de realizar a irrigação automática de uma planta de pequeno porte, monitorar variáveis físicas do meio através da utilização de sensores e fornecer esses dados ao usuário, permitindo que este acompanhe o *status* do sistema a qualquer instante.

1.3 ESTRUTURA DO TRABALHO

O presente trabalho está estruturado em capítulos além desta introdução (capítulo 1), conforme exposto a seguir:

- Capítulo 2: apresenta-se uma visão geral da arquitetura do sistema proposto, com seu funcionamento descrito;

- Capítulo 3: detalha-se todos os componentes utilizados no projeto, descrevendo suas principais características e conceitos teóricos envolvidos;
- Capítulo 4: relata-se as etapas presentes no desenvolvimento do protótipo final;
- Capítulo 5: apresenta-se os testes realizados e resultados obtidos;
- Capítulo 6: reúne-se as conclusões e considerações finais, bem como trabalhos futuros a serem realizados.

2 ARQUITETURA DO SISTEMA

O sistema de controle de irrigação residencial proposto visa automatizar a rega de uma planta de pequeno porte, implicando na economia de água através do uso inteligente deste recurso, e proporcionando ao seu proprietário acompanhar o status do sistema através de um *smartphone*.

É importante que o sistema de modo geral seja de baixo custo e eficiente, além de pouco invasivo. Na Figura 1 apresenta-se um diagrama geral da arquitetura proposta para o desenvolvimento do sistema.

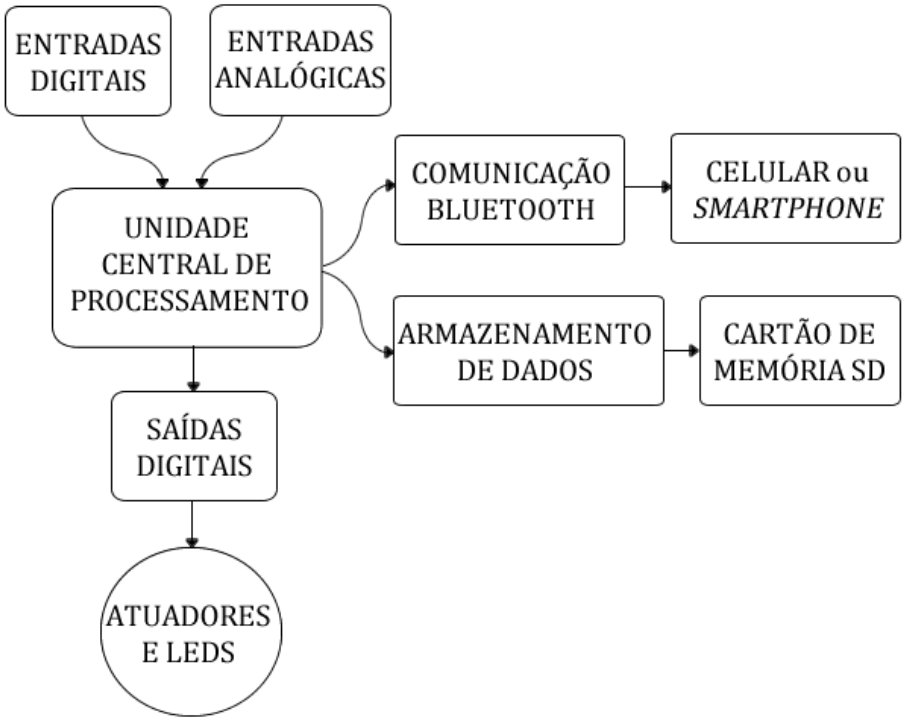


Figura 1: Diagrama geral da arquitetura proposta para o sistema.

O sistema possui uma unidade central de processamento, onde os sinais de entrada recebidos dos sensores ou através de protocolo de comunicação são processados para gerar as respectivas saídas desejadas. Os sinais de saída são enviados para módulos de acionamento de carga ou interfaces com o usuário, como LED. A interface entre o *smartphone* e o sistema se dá através de um módulo de comunicação *bluetooth*.

O funcionamento geral do sistema consiste na coleta dos dados, através dos sensores e armazenamento destas informações a cada 30 minutos. Além disso, seu atuador (válvula solenoide) pode ter acionamento automático, em função do valor lido pelo sensor de umidade do solo, ou manual, através de comando via *bluetooth* enviado pelo *smartphone*, que é capaz também de receber os dados das leituras dos sensores a qualquer momento. Esta descrição é representada na forma de fluxograma na Figura 2.

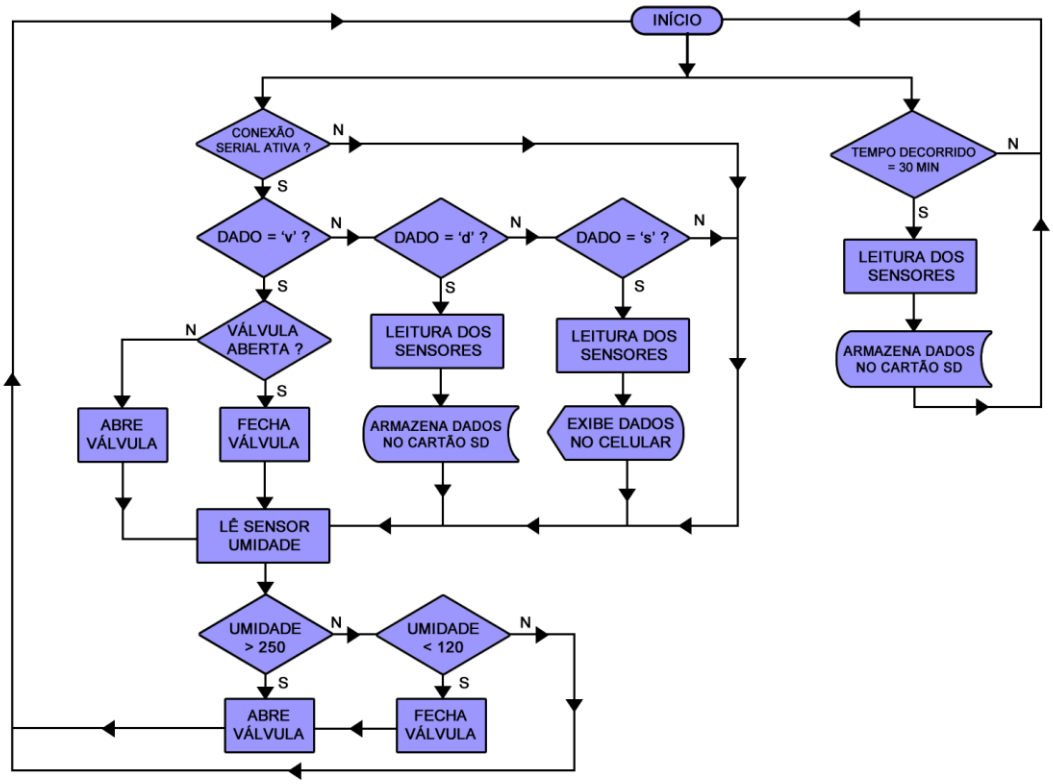


Figura 2: Fluxograma do funcionamento do sistema proposto.

3 COMPONENTES DO SISTEMA

Nesse capítulo serão apresentados todos os componentes integrantes do sistema de controle de irrigação, previamente apresentados na visão geral da arquitetura do projeto, bem como a descrição detalhada dos mesmos, suas aplicações no trabalho, além de conceitos teóricos referentes a protocolos e dispositivos utilizados no desenvolvimento.

3.1 ARDUINO UNO

A plataforma Arduino foi desenvolvida em 2005 na Itália, pela equipe liderada pelo professor Massimo Banzi e constituída também por David Cuartielles, Tom Igoe, Gianluca Martino e David Mellis. Seu objetivo era abstrair ao máximo a parte eletrônica do desenvolvimento de projetos, mensurando o mundo físico através do processamento de entradas e saídas, facilitando assim o desenvolvimento de projetos por entusiastas. Seu nome tem origem na palavra germânica *Hardwin*, onde *hard*(forte) e *win*(amigo), e adaptado para o italiano transformou-se em Arduino [9].

Atualmente já existem diversas versões de placas Arduino, sendo a mais conhecida e amplamente utilizada a Arduino UNO, atualmente na sua versão R3, sendo esta a utilizada neste projeto, e exposta na Figura 3.

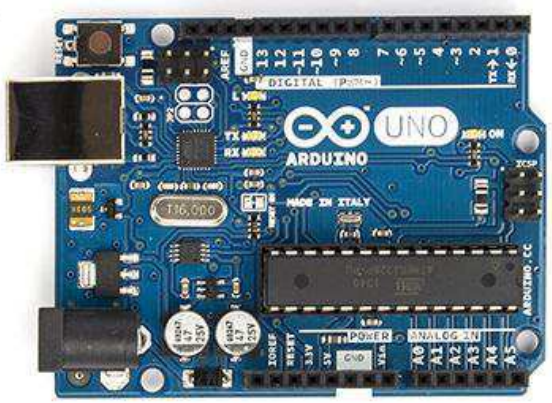


Figura 3: Foto da placa Arduino UNO R3.

Uma das principais características desta plataforma é sua documentação aberta, com todos os esquemáticos de *hardware* e diversas bibliotecas disponibilizadas abertamente, possibilitando que qualquer pessoa replique a placa e crie a sua própria versão. Outro ponto destacado é seu caráter modular, caracterizado pelas *shields*, placas com circuitos específicos que podem ser acoplados ao Arduino, expandindo suas funcionalidades[10]. Na Figura 4, são expostos alguns exemplos de *shields* para Arduino.

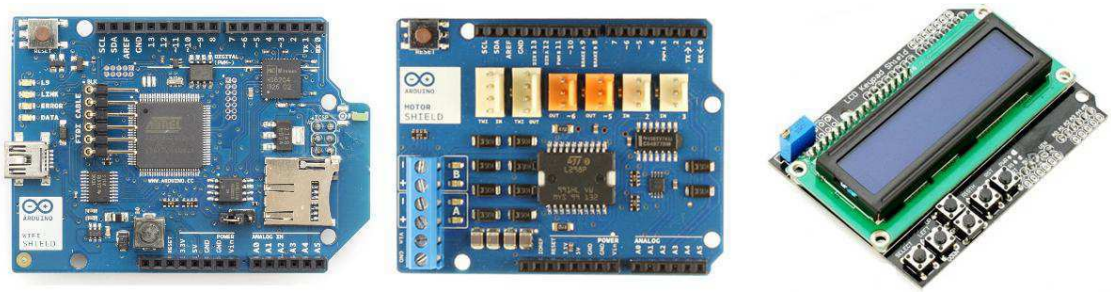


Figura 4: Exemplos de *shields* comerciais para Arduino: *WiFi Shield*, *Motor Shield*, e *LCD Keypad Shield*.

3.1.1 CARACTERÍSTICAS DE HARDWARE

Podemos ainda destacar como principais características do Arduino UNO:

- Microcontrolador ATmega328;
- 14 entradas/saídas digitais (das quais 6 podem ser utilizadas como saída PWM);
- 6 entradas analógicas com 10 bits de resolução cada;
- Tensão de operação: 5 V;
- Conexão USB e ICSP;
- Comunicação serial, SPI e i2c;
- 16 MHz de clock.

O microcontrolador presente como unidade central de processamento da placa Arduino, o ATmega328, pode ter seu funcionamento resumido através do diagrama funcional representado na Figura 5, que apresenta as estruturas de hardware do dispositivo[11].

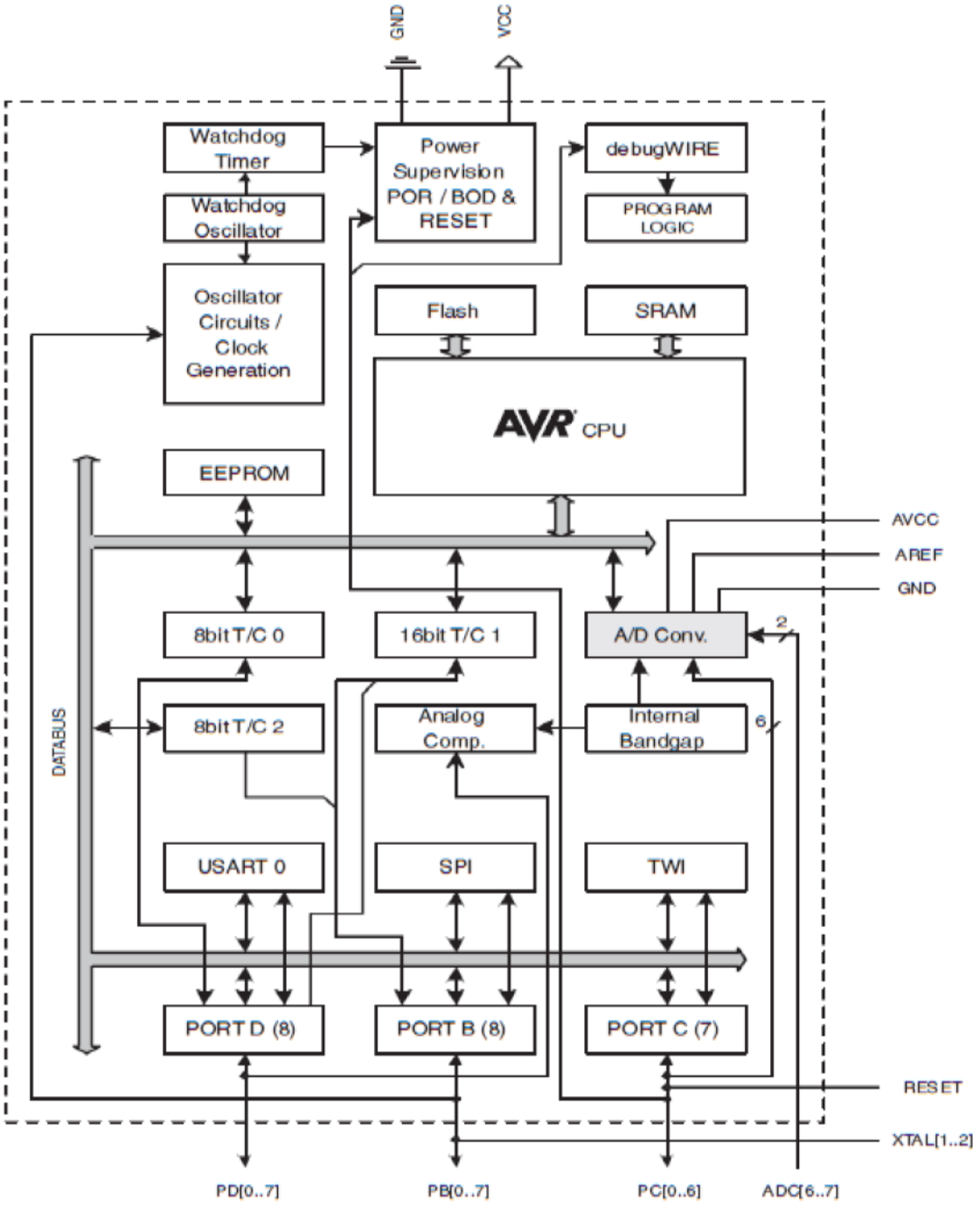


Figura 5: Diagrama de blocos do Atmega328.

O ATMega328 possui 32 registradores de uso geral, três temporizadores/contadores com modos de comparação, interrupções interna e externa, serial programável USART, uma porta SPI, *watchdog timer* com oscilador interno, e conversor A/D de 10 bits, representando 2^{10} níveis de tensão distintos. A resolução desse conversor pode ser expressa por:

$$Resolução = \frac{V_{REF}}{2^{10}}, \quad (1)$$

Sendo a tensão de referência, representada por V_{REF}[V], igual a 5 V, temos para o conversor A/D do Arduino, uma resolução de:

$$Resolução = \frac{5}{1024} \cong 5\ mV\ , \quad (2)$$

Pode-se afirmar, portanto, que o Arduino pode detectar variações de 5 mV. Há, contudo, a possibilidade de alterar-se o V_{REF}, através do pino de entrada A_{REF}, caso seja necessário para a aplicação desejada.

Ele é extremamente versátil, com elevada capacidade de processamento e diversificado conjunto de periféricos, e quando utilizado na plataforma Arduino (a plataforma consiste basicamente em uma solução de suporte para *hardware* e *software*, contribuindo para a programação e desenvolvimento de projetos com esse microcontrolador) se torna a solução ideal para o projeto proposto nesse trabalho, haja vista que é portátil, tem um conjunto de entradas e saídas mais que satisfatório, documentação aberta e disponível sem custos adicionais, além do seu caráter modular que permite facilmente alterações na proposta do projeto e expansão de funcionalidades[12].

3.1.2 CARACTERÍSTICAS DE SOFTWARE

A plataforma de *software* do Arduino, modelada na linguagem de programação *Processing*[1], denominada IDE (*Integrated Development Enviroment*), trata-se de um programa executado em um computador, que permite a codificação de programas em linguagem específica, *sketches*, a serem carregados na plataforma do Arduino[13].

É importante destacar que o microcontrolador ATmega328 trabalha com programação em C. A IDE do Arduino traduz o *sketch* escrito pelo desenvolvedor em linguagem C, e na sequência realiza a gravação do código no microcontrolador.

Desta maneira, o ciclo de programação de um Arduino pode ser resumido nesses passos:

1. Escreve-se o *sketch*;
2. Compila-se o *sketch*;
3. Liga-se a placa a uma porta USB do computador;
4. Envia-se o *sketch* à placa, através da porta USB;

[1] *Processing* é uma linguagem de programação de código aberto e ambiente de desenvolvimento integrado desenvolvido em 2001 pelo MIT (Massachusetts Institute of Technology), construído para comunidades de projetos visuais com o objetivo de ensinar noções básicas de programação de computador em um contexto visual.

5. A placa executa o programa.

O IDE do Arduino possui interface amigável e familiar, podendo ser executado em qualquer sistema operacional. Sua interface está exposta na Figura 6, a seguir:

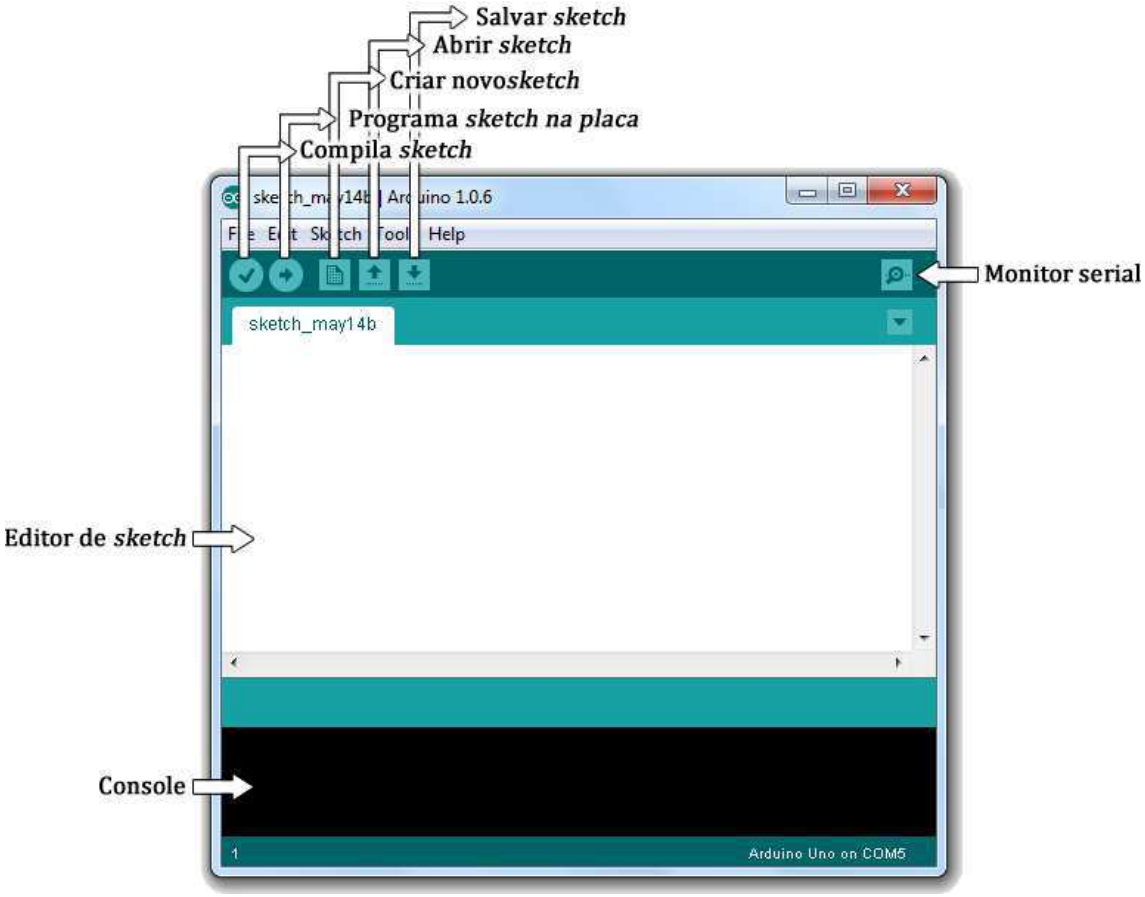


Figura 6: Plataforma de software (IDE) do Arduino.

3.2 ARDUINO ETHERNET SHIELD

Conforme supramencionado, uma das principais vantagens da plataforma Arduino é seu caráter modular, cujo principal expoente são seus *shields*, módulos que, quando acoplados ao Arduino, expandem suas funcionalidades. São placas de circuito impresso que agregam recursos de *hardware* ao microcontrolador, e apresentam como mais marcante característica o mesmo design da placa Arduino UNO, para um encaixe perfeito.

Para esse projeto, foi utilizado o Arduino Ethernet Shield, representado na Figura 7, e responsável por tornar o Arduino capaz de se conectar à internet. Como

todas as placas Arduino, este *shield* possui toda a sua documentação aberta e disponível de graça. Os principais motivos para a escolha desta *shield* para o projeto foram: a presença de um adaptador para cartão de memória SD, e oferecer a possibilidade de expansão de funcionalidades do projeto.

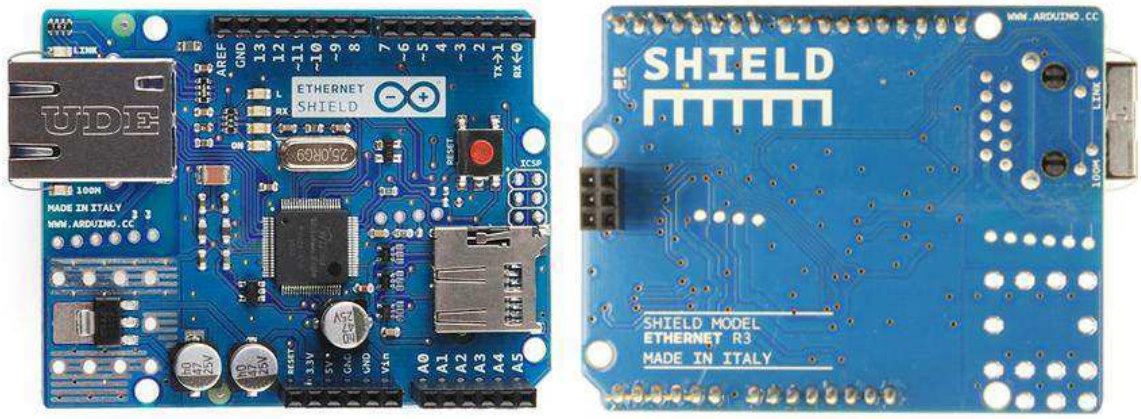


Figura 7: Foto da placa Arduino *Ethernet Shield*.

Esta placa é baseada no chip ethernet Wiznet W5100, responsável por prover acesso à internet, utilizando o protocolo de comunicação TCP/IP, e tem ainda como características que merecem destaque:

- Velocidade de conexão 10/100 Mb;
- Conector padrão RJ45;
- Conexão com o Arduino via protocolo SPI.

O SPI (*Serial Peripheral Interface*) é um protocolo de dados serial síncrono, utilizado por microcontroladores para comunicação rápida com um ou mais dispositivos periféricos em curtas distâncias, além de também poder ser usado para comunicação entre dois microcontroladores, e sendo usado pela *shield* também para armazenar dados e realizar a leitura do cartão de memória SD presente na placa.

Em uma comunicação SPI, sempre há um mestre e um ou mais escravos, que se comunicam entre si através de uma conexão *full-duplex*. Os seguintes fios constituem a conexão SPI:

- MOSI (*Master Out Slave In*): Canal responsável pelo envio de dados do mestre para o escravo;
- MISO (*Master In Slave Out*): Canal responsável pelo envio de dados do escravo para o mestre;
- SCK (*Serial Clock*): Canal para envio do sinal de clock;

- SS (*Slave Select*): Canal utilizado pelo mestre para habilitar ou desabilitar a conexão com o escravo. Se na conexão existir mais de um escravo, todos compartilham os canais citados anteriormente, com exceção deste. Cada escravo possui seu SS. Quando este pino estiver em nível baixo, há conexão com o mestre e quando estiver em nível alto, o escravo ignora as informações do barramento.

Na comunicação SPI são utilizados dois registradores de deslocamento para a troca de informações, um pertencente ao mestre e o outro ao escravo. Os bits são deslocados e enviados um por vez, entre os registradores. À medida que um bit é transmitido pelo canal MOSI, outro é recebido pelo canal MISO, e após todos serem enviados, os registradores do mestre e do escravo estarão com os dados trocados [14].

Para utilizar o *shield ethernet* com o Arduino UNO R3, faz-se necessário o uso de três bibliotecas de *software* disponibilizadas gratuitamente no site do Arduino, e que são instaladas junto com a IDE de programação dos *sketches*. Essas bibliotecas são a Ethernet.h, cujas funções não serão utilizadas a princípio no projeto, a SPI.h, responsável por ler e escrever comandos SPI, e a SD.h, que contém as funções necessárias para trabalhar com o cartão de memória SD. A Figura 8 apresenta quais pinos do Arduino a *shield* utiliza, desta maneira estes pinos não podem ser utilizados para outra função no projeto, a fim de se evitar conflitos e mau funcionamento[15].

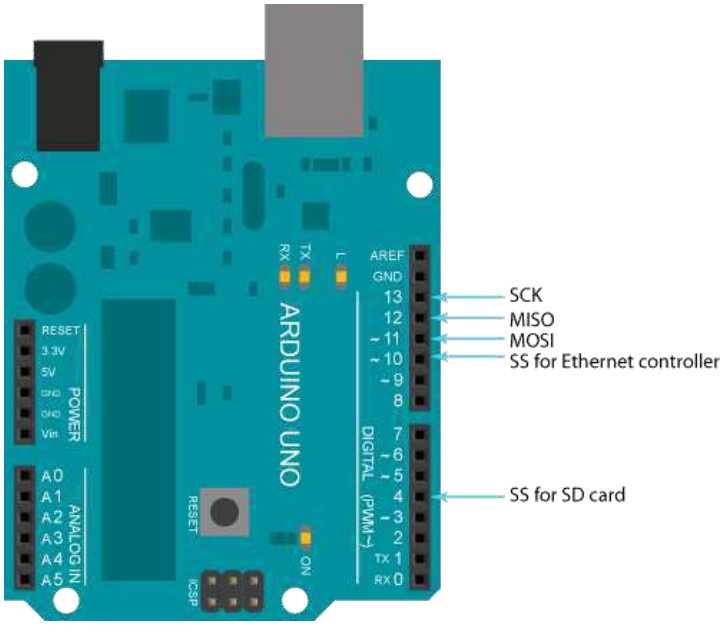


Figura 8: Representação dos pinos utilizados pela *shield ethernet*.

3.3 MÓDULO BLUETOOTH HC-06

Uma das necessidades do projeto é o monitoramento constante de dados, bem como o controle e ativação de recursos do sistema, a serem feitos de maneira remota. Uma solução para isso é o uso da comunicação sem fio, e neste caso optou-se pelo uso do protocolo de comunicação *Bluetooth*. Este modelo realiza de forma simples e prática a comunicação do Arduino com um dispositivo celular, no caso um aparelho com sistema operacional *Android*.

O *bluetooth* utiliza a faixa de frequência de 2,4 GHz para comunicação sem fio, e se caracteriza por ser uma solução de baixo custo, baixo consumo de energia, e baixa complexidade de configuração para interligar dispositivos diversos. Por estes motivos, e por atender a todas as necessidades do projeto, foi escolhido este protocolo para a comunicação. Outra opção seria o uso do *ZigBee*, mas sua configuração com *smartphones* não é tão simplificada como no caso do *bluetooth*, além de ter um custo mais elevado.

Para este projeto, foi escolhido o módulo genérico *bluetooth* HC-06, representado na Figura 9, de baixo custo e interface de comunicação simples. Ele se comunica através de comunicação serial RS-232 com o microcontrolador, fazendo uso, portanto, dos pinos RX e TX do Arduino. O módulo já possui o *hardware* de suporte implementado para o funcionamento correto do *bluetooth*.

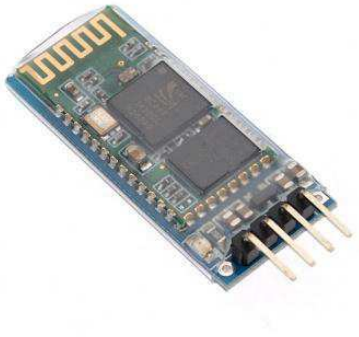


Figura 9: Representação do módulo *bluetooth* HC-06.

Este dispositivo possui 4 pinos, sendo dois destinados à comunicação serial, RX e TX, além do pino de alimentação que deve receber 5 V DC, e um terra. Ele opera a uma taxa de transmissão de 9600 bps, portanto é importante configurar no *software* do projeto a transmissão para esta mesma taxa[16].

Os dados enviados pelo módulo serão recebidos por um *smartphone* Android, e lidos através de um aplicativo de terminal *bluetooth*¸ semelhante a *softwares* terminais de comunicação serial.

3.4 SENSOR DE UMIDADE DO SOLO HL-69

Dada a natureza do projeto e o tipo de plantação que é objeto dele (plantas de pequeno porte em ambiente residencial), foi escolhido um sensor resistivo de umidade do solo, de baixo custo e baixo consumo. Há, no mercado, opções superiores tanto em qualidade como em preço, mas a opção escolhida satisfaz as necessidades do projeto, sem elevar os gastos. Sensores de umidade do solo mais simples apresentam normalmente um alto nível de oxidação, mas a curto prazo esse não é um problema a ser considerado, em especial para um protótipo. Como o funcionamento dos diversos tipos de sensores costumam ser semelhantes, a substituição dele por um de maior qualidade pode ser feita facilmente, logo, sem interferência no projeto.

O sensor HL-69[17], apresentado na Figura 10, consiste de duas sondas, que são submetidas a uma diferença de potencial fixa, permitindo que seja calculada a resistência no meio em que elas se encontram. No caso em estudo, iremos medir a umidade relativa do solo. Quanto maior a quantidade de água no solo, maior será sua condutividade, e portanto, menor será a resistência medida. Ele possui também um módulo de suporte de hardware, que possui duas saídas A0 e D0, analógica e digital, respectivamente, que fornecem ao Arduino o valor lido pelo sensor. A saída digital, que utiliza um chip LM393, cujos amplificadores operacionais realizam a comparação do sinal medido com um de referência, possui um potenciômetro para ajuste da tensão a ser comparada.

Neste projeto, faremos uso apenas da saída analógica, que apresenta um resultado mais preciso.

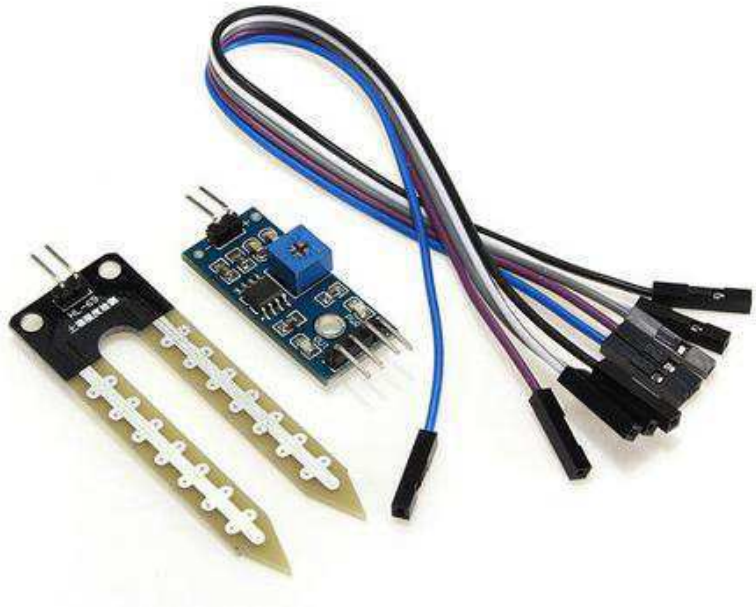


Figura 10: Representação do sensor HL-69 e seu módulo de suporte.

Este dispositivo opera com tensão entre 3,3 e 5 V, e possui esquema elétrico de acordo com o descrito na Figura 11[17].

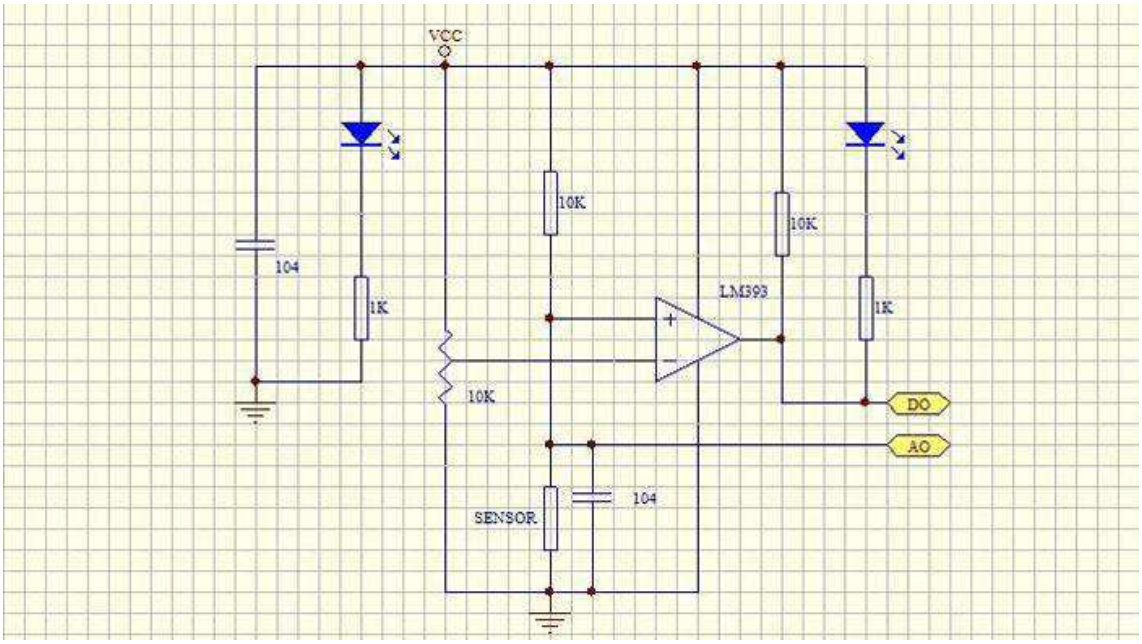


Figura 11: Esquema elétrico do sensor HL-69.

3.5 SENSOR DE UMIDADE E TEMPERATURA DHT11

O sensor DHT11, representado na Figura 12, é um sensor de temperatura e umidade que permite realizar leituras de temperaturas entre 0 e 50 °C, e umidade

relativa entre 20 a 90%, apresentando ainda precisão de ± 2 °C e ± 5%, o que satisfaz as necessidades do projeto[18].

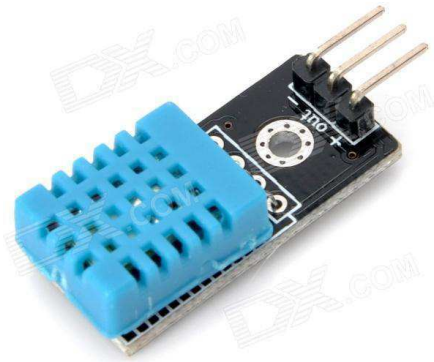


Figura 12: Representação do sensor DHT11.

Este dispositivo opera com tensão de 3,3 a 5 V DC, e conforme indicado no seu esquema elétrico na Figura 13, utiliza o protocolo de comunicação 1-wire[2][19].

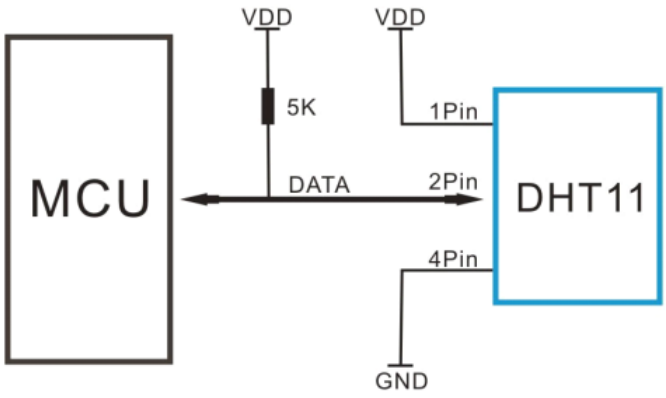


Figura 13: Esquema de conexão do DHT11.

No caso do DHT11, um único processo de comunicação dura aproximadamente 4ms. Os dados consistem de 40 *bits*, sendo 16 *bits* para o valor da umidade, 16 *bits* para o da temperatura, e 8 *bits* de *checksum*. Tendo posse destes dados fornecidos pelo fabricante do sensor, é possível enviar o sinal de início para o sensor, que responderá confirmando o recebimento do comando, e em seguida irá enviar os dados da sua leitura, pelo mesmo canal pelo qual recebeu seu sinal de comando[18].

[2] Desenvolvido pela Maxim, este protocolo se caracteriza, como indica seu nome, pelo uso de apenas um canal, de forma bi-direcional, na comunicação entre os dispositivos. Tal processo é iniciado e controlado pelo dispositivo *Mestre*, um microcontrolador, de acordo com as especificações do *Escravo*.

Para o Arduino, existe a biblioteca DHT.h, de documentação aberta e disponibilizada gratuitamente, que abstrai a parte de comandos, pulsos de sinais enviados e leitura de dados recebidos, facilitando o trabalho de leitura do sensor.

A utilização deste sensor possibilita ao projeto um monitoramento constante de variáveis importantes do ambiente que cerca o objeto do projeto.

3.6 VÁLVULA SOLENOIDE W09

Válvulas solenoides são registros (torneiras) com acionamento automático através de um comando elétrico. Funcionam da seguinte maneira: a bobina solenoide ao ser percorrida por uma corrente elétrica magnetiza um contato mecânico que abre ou fecha a válvula, permitindo ou não a passagem de água. O modelo das válvulas utilizadas neste projeto está representado na Figura 14.

Figura 14: Foto da válvula solenoide W09.

O modelo utilizado possui diâmetro de 1/2”, com resistência interna de 9 Ω a 20 °C, tensão de operação de 3,6 V DC, pressão de trabalho de 0,2 – 1 MPa[20].

Seu funcionamento é da seguinte maneira: aplicando um pulso de 3,6 V durante 30ms, em uma determinada polaridade, ela se abre e se mantém aberta, haja vista que esse dispositivo memoriza a posição da válvula, o que contribui na economia de energia. A mesma lógica se aplica ao fechamento da válvula, quando aplicado o mesmo pulso, na polaridade oposta.

Existem diversos modelos de válvulas, porém a maioria delas é alimentada com tensão superior a 5 V DC, o que implicaria em uso de alimentação externa para a mesma. É uma opção viável, mas para fins de pesquisa e de protótipo, o modelo escolhido cumpre com o seu papel e atende as necessidades do projeto.

Desta maneira, para gerar o sinal que irá controlar a válvula, a solução encontrada foi a utilização de uma ponte H, presente no circuito integrado L293D descrito na sessão a seguir.

3.7 PONTE H L293D

Para a ativação das válvulas solenoides, é necessário o uso de uma ponte H. Esse é um circuito de eletrônica de potência que, através do acionamento de determinadas chaves, permite a circulação da corrente em sentidos opostos, de acordo com o desejado. O circuito da Figura 15 ilustra o funcionamento de uma ponte H, usada para alimentar um motor DC, a título de exemplo, com o caminho percorrido pela corrente no circuito destacado em vermelho.

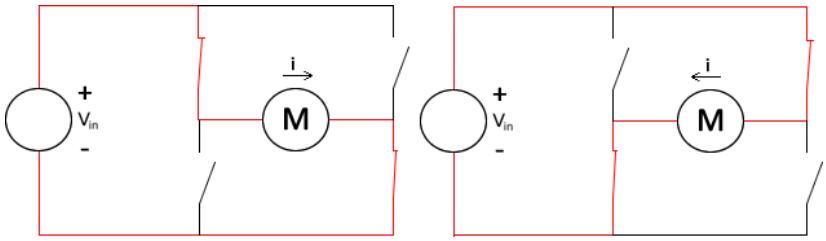


Figura 15: Esquema de funcionamento de uma ponte H alimentando um motor DC.

Como a tarefa proposta envolve o controle de duas válvulas solenoides, é necessário o uso de duas pontes para o controle independente delas. Desta maneira, optou-se pelo circuito integrado L293D. Esse CI carateriza-se principalmente por ter duas pontes, atendendo às necessidades do projeto. Na Figura 16 (A), podemos observar o seu diagrama de pinos, bem como a foto do componente real na Figura 16 (B).

Figura 16: (a) Diagrama de pinos do L293D; (b) Imagem do circuito integrado L293D.

De utilização muito comum em relés, motores DC e solenoides, esse circuito apresenta a opção de alimentação externa (para o caso, por exemplo, de motores com alimentação superior a 5 V), e possui uma tabela de funções para suas saídas, de acordo com a tabela 1[21].

Tabela 1 : Funcionamento lógico de cada driver do L293D

ENTRADAS		SAÍDA
A	**EN**	**Y**
1	1	1
0	1	0
X	0	0

3.8 MÓDULO RELÓGIO DE TEMPO REAL DS1302

Como será realizado o monitoramento de sensores, aquisição e armazenamento de dados, é importante a utilização de um relógio em tempo real no sistema, a fim de se armazenar horários em que eventos importantes aconteceram, além de controlar o monitoramento dos sensores, haja vista que a leitura dos mesmos é feia a cada 30 minutos, contagem esta a ser feia pelo relógio.

Para este projeto, foi escolhido o módulo RTC (*Real Time* Clock) DS1302, exibido na Figura 17, que consiste em um chip DS1302, com funções de hora, data e calendário. Ele é capaz de fornecer informações de data, mês e ano, e ajusta automaticamente dados referentes dos meses com menos de 31 dias, além de anos bissextos. Desta maneira, ele é mais que suficiente para atender as necessidades do projeto[22].

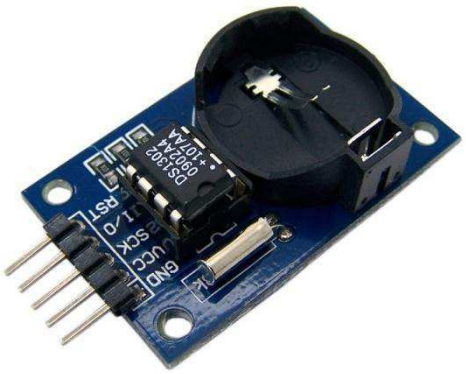


Figura 17: Foto do módulo DS1302 com *hardware* de suporte.

Na Figura 18 é exposto o diagrama de um circuito típico do DS1302. É importante destacar a presença de duas fontes de alimentação. Uma delas, a V_{cc2}, é a alimentação fornecida pelo Arduino, e a outra, V_{cc1}, é a tensão de 3,3 V fornecida pela bateria externa, responsável por manter o relógio funcionando mesmo quando o Arduino estiver desligado. Deve-se usar um oscilador cristal de 32.768 kHz. O DS1302

utiliza 3 pinos para realizar a comunicação com o microcontrolador: RST (Reset ou CE – Chip Enable), DATA ou I/O (Entrada/Saída) e SCLK (Serial Clock).

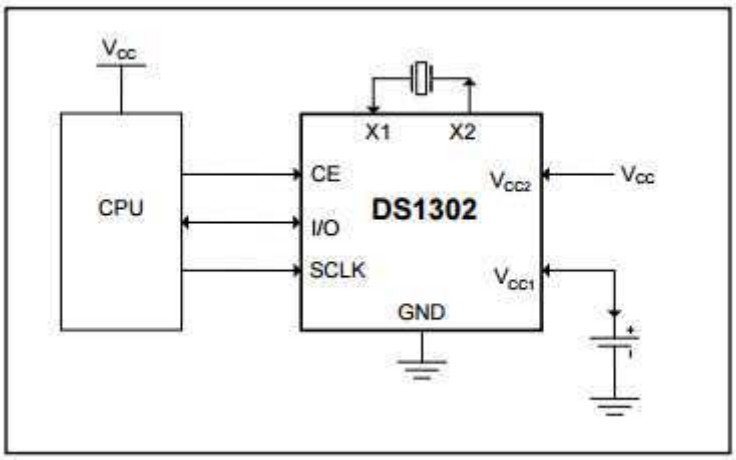


Figura 18: Esquema de circuito do DS1302.

A comunidade de desenvolvedores para Arduino desenvolveu uma biblioteca de documentação aberta e gratuita, denominada *virtuabotixRTC*, que facilita o trabalho de realizar a comunicação byte a byte com o DS1302, configurá-lo e extrair os dados[23].

Esse é um componente de baixo custo e apesar de não ter uma longa vida útil e apresentar problemas a longo prazo, a curto prazo, como protótipo, satisfaz todas as necessidades do projeto.

4 DESENVOLVIMENTO DO PROTÓTIPO

Com o intuito de validar o modelo de sistema de controle de irrigação proposto, foi projetado um protótipo contendo todos os elementos integrantes do projeto.

Teve-se como objetivo verificar o correto funcionamento do sistema em todas as suas etapas, desde a aquisição de dados, processamento de dados, saídas e acionamento de atuadores. Para isso, o protótipo pensado deveria ter as seguintes características:

- Portabilidade: ter um tamanho reduzido, facilitando seu transporte;
- Baixo custo: tendo fins de pesquisa principalmente, o protótipo deve ter baixo custo, reaproveitando-se inclusive materiais de outras montagens;

Tendo em mente essas características, foi elaborado primeiramente o circuito elétrico completo, contendo todos os componentes, já descritos na seção anterior deste trabalho. Este esquema é apresentado na Figura 19, e em mais detalhes na seção de anexos.

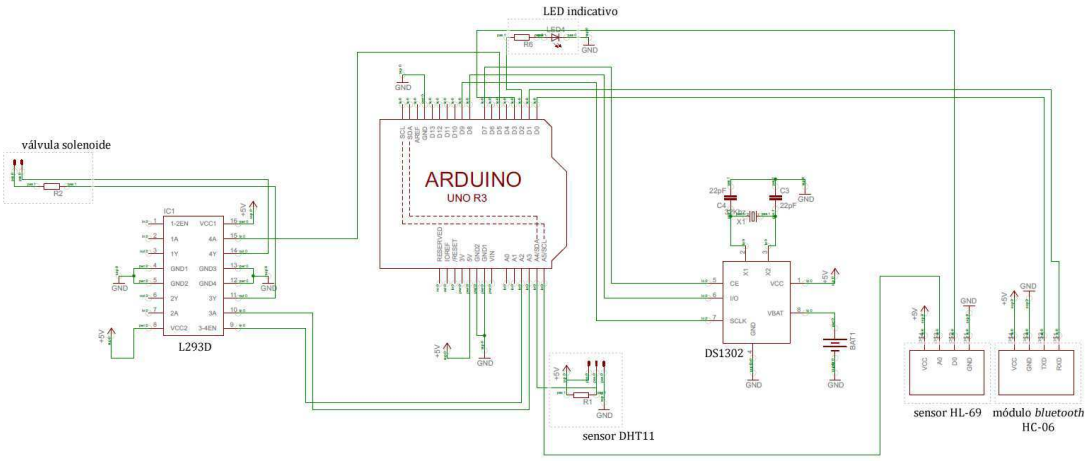


Figura 19: Esquema elétrico do protótipo.

O circuito poderia ser montado em uma *protoboard*, mas para que atendesse com mais fidelidade ao seu propósito, optou-se por confeccionar uma *shield* própria para Arduino, contendo todos os elementos externos componentes do sistema. Como os esquemáticos e especificações do Arduino são de domínio público, foi possível criar uma *shield* com as mesmas dimensões da placa Arduino UNO R3, o que proporcionou um encaixe perfeito no próprio Arduino bem como entre outras shields comerciais.

4.1 CONFECÇÃO DA *SHIELD* PRÓPRIA

O objetivo desta etapa era confeccionar uma PCI (placa de circuito impresso), nas dimensões do Arduino UNO R3, capaz de integrar os seguintes elementos externos do circuito:

- Sensor de umidade e temperatura DHT11;
- Sensor de umidade do solo HL-69;
- Módulo de comunicação *bluetooth* HC-06;
- Circuito integrado L293D;
- Circuito integrado DS1302;
- Bateria de 3,3 V;
- Válvula solenoide W09.

Desta maneira, com posse do esquema elétrico já exposto do circuito, utilizou-se o *software* EAGLE[3], para confecção da placa, cujo design é exposto na Figura 20[24].

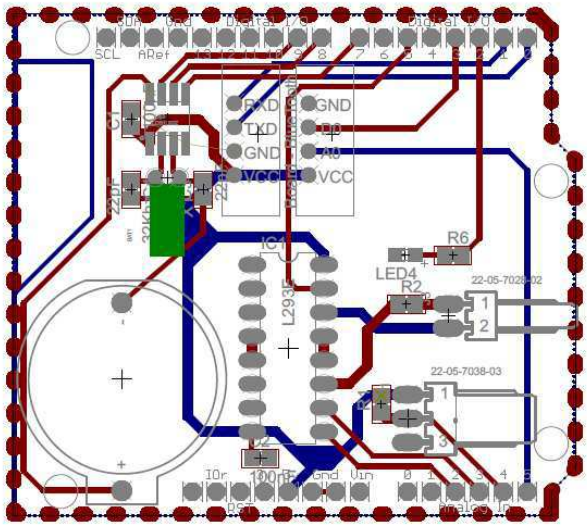


Figura 20: Design da placa da *shield* do protótipo.

Após a elaboração do *design* da placa, a mesma foi confeccionada utilizando a LPKF ProtoMat S42, uma máquina de prototipagem de PCIs, capaz de produzir placas de face-dupla, satisfazendo desta maneira o design proposto para a *shield*. A máquina realiza sua comunicação com o computador via USB ou RS-232, e sua fabricante

[3] Desenvolvido pela empresa alemã CADSOFT, é uma das ferramentas mais utilizadas para design de PCIs, possui interface intuitiva e familiar a usuários com experiência em outros *softwares* de CAD.

fornece o *software* responsável por ler os arquivos *Gerber* gerados pelo EAGLE e realizar a confecção da placa, esta exposta na Figura 21[25].

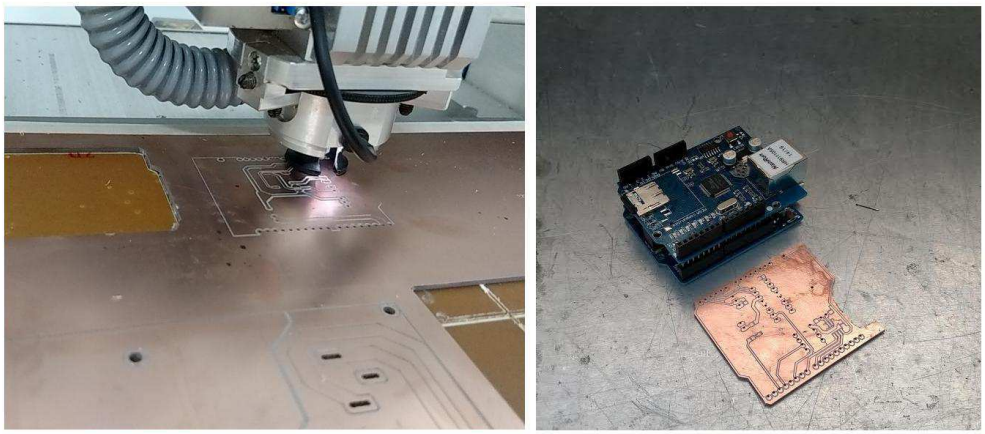


Figura 21: À esquerda, processo de prototipagem da placa, e à direita, o *shield* confeccionado ainda sem os componentes, ao lado do Arduino.

Tendo sido feito o levantamento completo de componentes do circuito, e com a placa de circuito impresso em mãos, foi feita a soldagem dos componentes principais (sensores, atuadores, etc) e auxiliares (resistores, capacitores, conectores, etc) na placa. O resultado final foi extremamente satisfatório, o produto final atendeu às expectativas no que se refere ao seu caráter de *shield*, e está exposto nas Figura 22 e Figura 23. É importante destacar o modelo de pinos escolhidos para esta placa, pois estes permitem o encaixe de mais *shields* acima da confeccionada, mantendo o caráter expandível e modular do projeto.

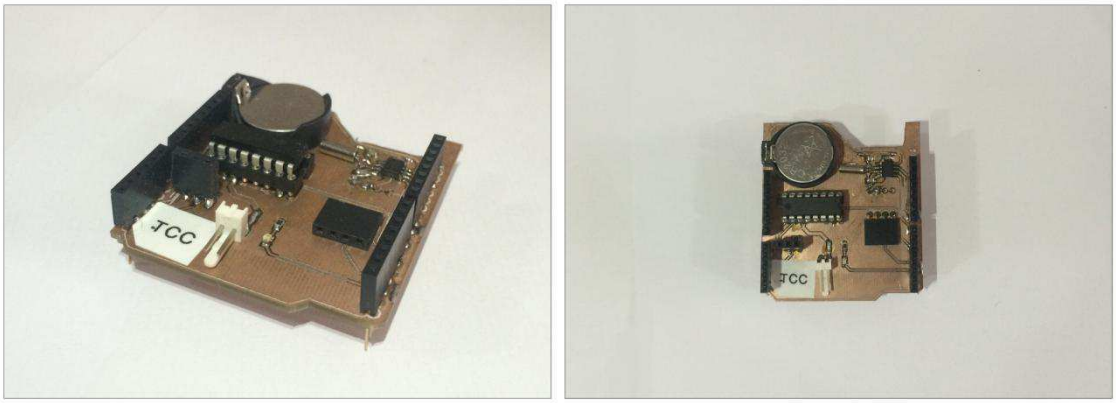


Figura 22: Face superior da *shield* confeccionada.

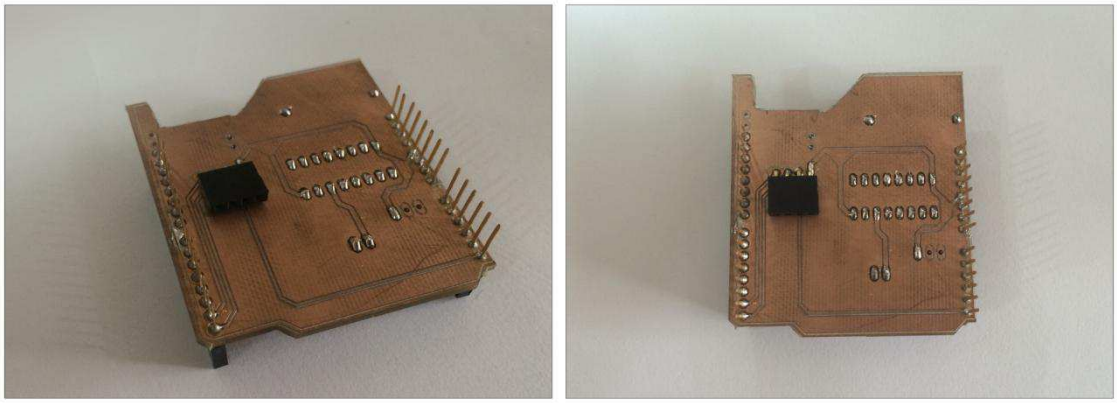


Figura 23: Face inferior da *shield* confeccionada.

4.2 PROTÓTIPO FINAL

Com a *shield* confeccionada pronta, denominada de "TCC", foi possível montar o protótipo final, contendo todos os componentes do projeto. A Figura 24 retrata a montagem das duas *shields* sobre o Arduino UNO R3 (estão ausentes nesta representação a válvula solenoide w09 e as sondas do sensor de umidade do solo). O produto final, com todos os componentes integrantes, é detalhado na Figura 25.

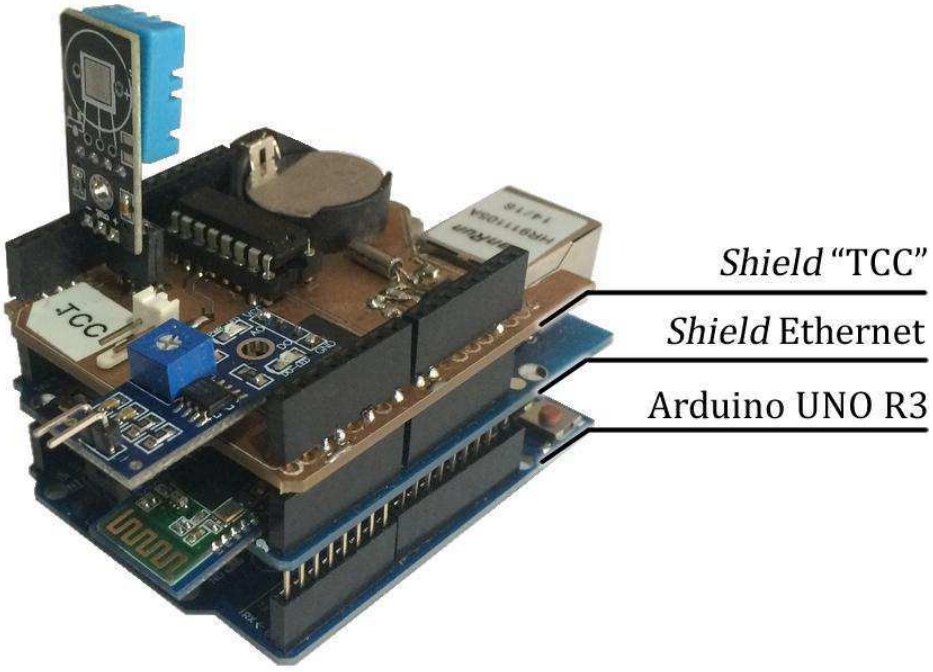


Figura 24: Foto do Arduino UNO R3 com as *shields* Ethernet e TCC acopladas ao mesmo.

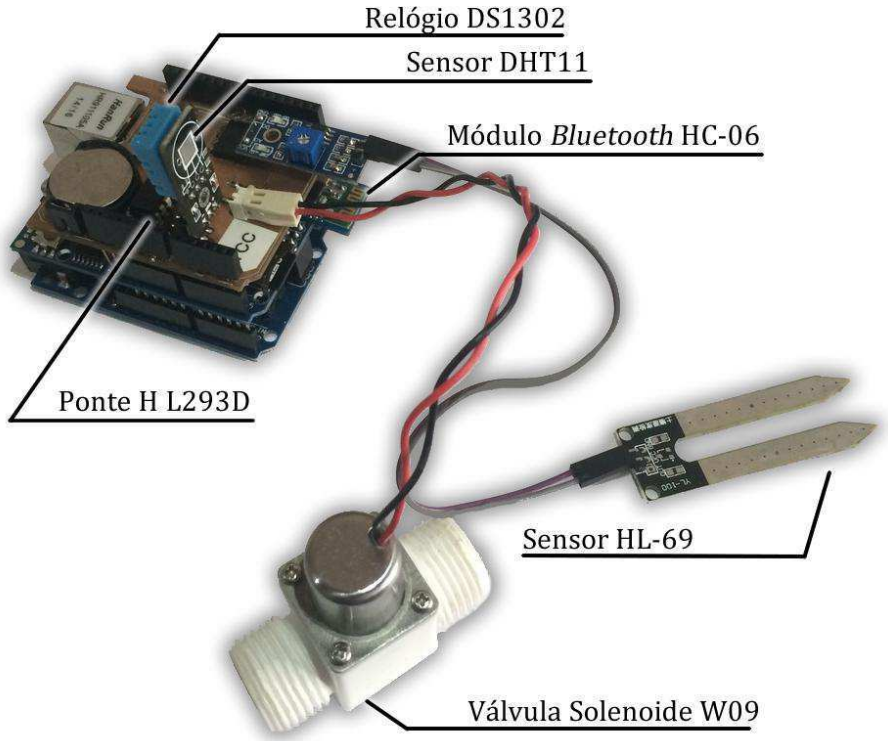


Figura 25: Produto final, com todos os seus componentes integrantes periféricos detalhados.

5 RESULTADOS

Neste capítulo serão apresentados os testes que foram realizados para validar o sistema proposto, bem como os resultados obtidos.

Primeiramente, foi realizado um teste do módulo *bluetooth* HC-06. Sua conexão com o Arduino é direta, fazendo uso dos pinos RX e TX da comunicação serial, que deve ser configurada para taxa de 9600 bps, conforme supracitado. Foi utilizado o aplicativo S2 Terminal for Bluetooth, disponível para *Android* na *Google Play Store* gratuitamente. Trata-se de um terminal de comunicação *bluetooth*, com funcionamento análogo ao *HyperTerminal*, software do *Windows* que realiza a comunicação serial do computador.

O módulo HC-06 vem pré-configurado de fábrica, com código PIN de comunicação "1234" e nome "HC-06". Não foi necessária a alteração destes parâmetros ou quaisquer outros do módulo, por se tratar de um protótipo, mas pode ser desejável, para um produto final, trocar o código PIN para fins de segurança. A conexão com o celular é direta e simples, e a interface do aplicativo está exposta na Figura 26, com o teste de comunicação tendo sido realizado com sucesso.

Figura 26: Interface do terminal *bluetooth* no aparelho Android.

Conforme pôde ser observado na figura 26, foi realizado o teste de comunicação do sistema, onde foi recebido e enviado dados com sucesso, além do teste de inicialização do sistema e leitura dos sensores. A interface *bluetooth* pensada para o protótipo contempla 3 funções, conforme descrito no fluxograma de funcionamento geral do sistema, no Capítulo 2 deste trabalho, sendo elas ativadas pelos seguintes comandos:

- 's' – realiza a leitura dos sensores do sistema e envia estes dados via *bluetooth* para o aplicativo terminal;
- 'v' – realiza a ativação/desativação da válvula;
- 'd' – realiza a leitura dos sensores e armazena os dados obtidos no cartão de memória.

Quanto ao sensor de umidade do solo, os valores lidos pelo Arduino na saída analógica do sensor podem variar de 0 a 1023, onde 0 é o valor lido quando o sensor está completamente imerso em água, e 1023 quando há apenas ar entre as suas sondas. Após testes experimentais, observou-se que, em solo extremamente seco, se situa em torno de 300, enquanto que, em solo úmido, este valor se aproxima de 120. Desta maneira, o algoritmo desenvolvido visou manter a umidade do solo tal qual o valor lido pelo sensor não ultrapassasse 250, mantendo um ambiente propício para o desenvolvimento sadio da planta.

Posteriormente, foi testada a válvula solenoide W09 através do comando via *bluetooth*. O funcionamento da mesma foi de acordo com o esperado, porém é importante ressaltar que este modelo de válvula utilizado apenas funciona com água sob pressão, i.e. não foi possível utilizar apenas a força da gravidade e conectar a válvula diretamente a um reservatório. Foi necessário conectar a válvula a uma mangueira, e por estar em constante pressão, foi importante uma conexão segura entre as partes (torneira, válvula e mangueira), para evitar vazamentos que pudessem danificar o protótipo.

Em seguida, conforme o algoritmo do código-fonte desenvolvido, exposto na seção de anexos, foi testada a funcionalidade de armazenamento dos dados obtidos no módulo de cartão de memória SD presente na *shield Ethernet*. Foram armazenadas as leituras dos sensores em um arquivo denominado "data.txt", e posteriormente esses dados foram observados com êxito.

Tendo sido testados todos os componentes do protótipo de forma individual, foi elaborado um arranjo experimental para irrigação de uma planta de pequeno porte,

plantada em um vaso. Este arranjo, que incluiu a planta e o protótipo com todos os seus periféricos integrantes, está exposto na Figura 27.

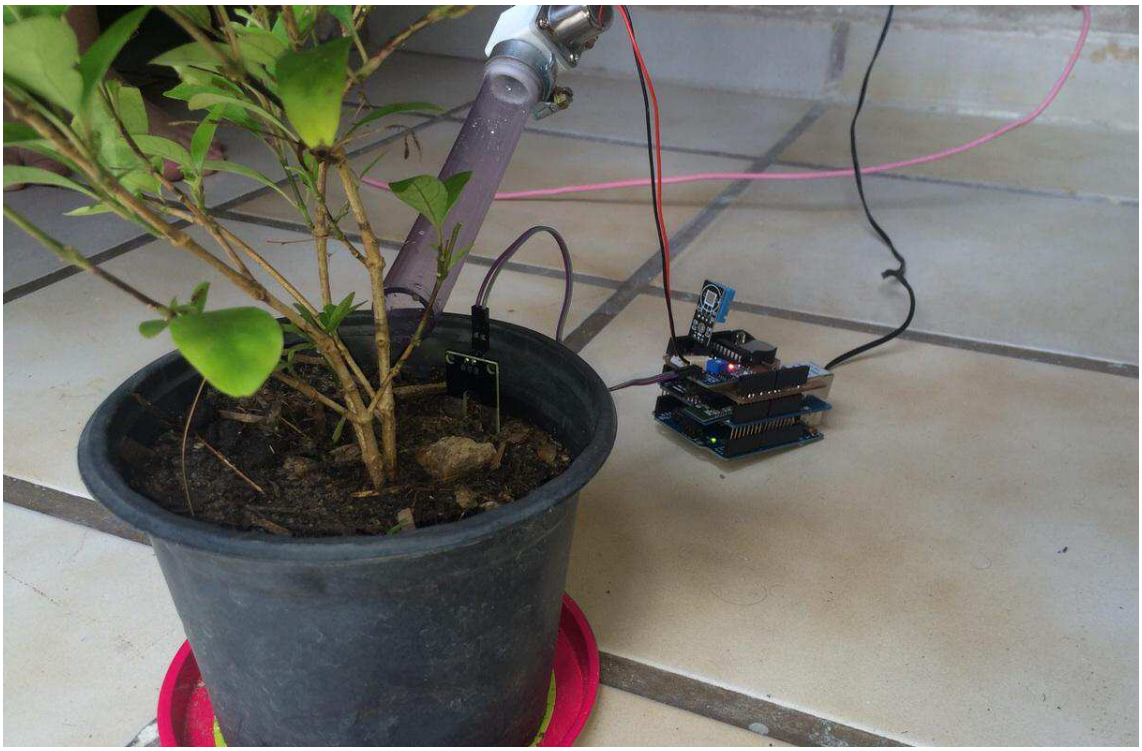


Figura 27: Arranjo experimental proposto para irrigação da planta.

Inicialmente, foi testada a irrigação da planta manualmente através do comando via *bluetooth*, e garantida a segurança do modelo contra vazamentos, o sistema funcionou em modo autônomo, funcionando por 3 (três) dias consecutivos.

Durante esses três dias, todos os dados dos sensores – umidade do solo, temperatura ambiente e umidade relativa do ar – foram armazenados periodicamente (com intervalos de 30 minutos) no cartão de memória, para análise posterior. Após o período de testes ter se concluído, os dados presentes no cartão de memória foram processados e foram gerados gráficos a respeito das variáveis monitoradas e do comportamento do protótipo. Para fins de visualização, os gráficos foram divididos em três, um por dia, podendo ser observados nas Figura 28 a Figura 30.

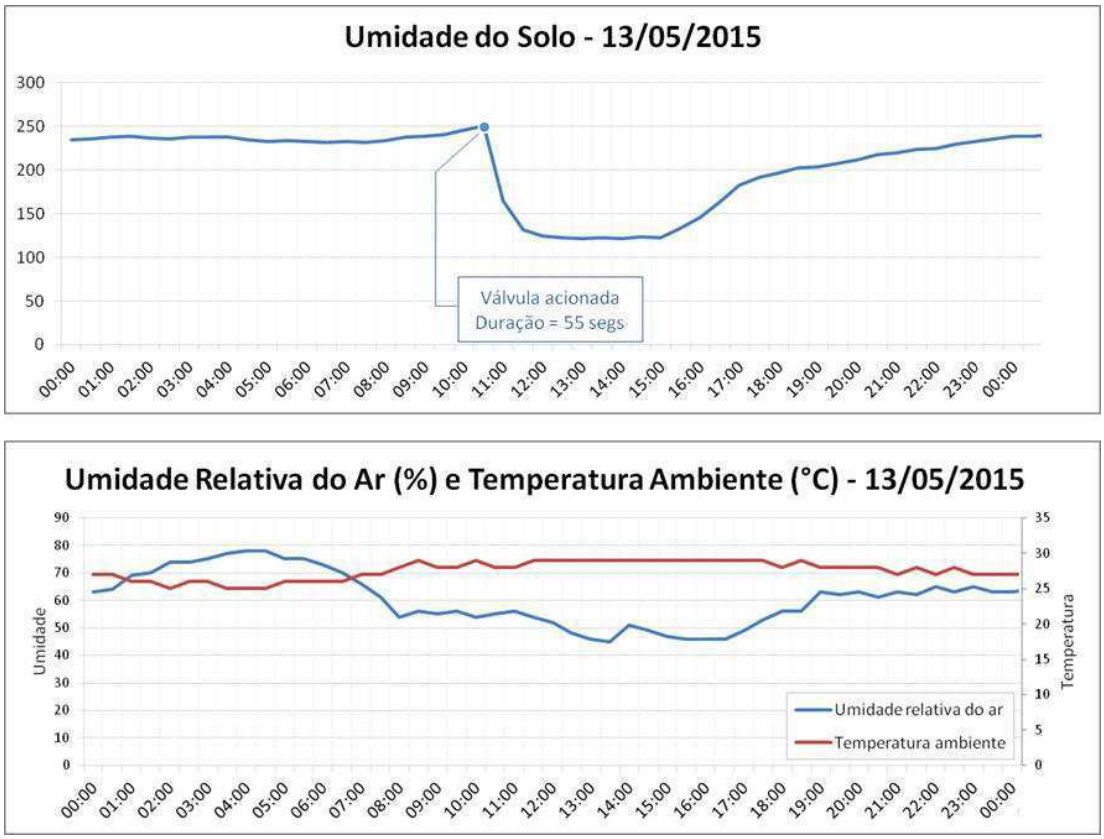


Figura 28: Gráficos referentes à umidade do solo, umidade relativa do ar e temperatura ambiente no dia 13/05/2015.

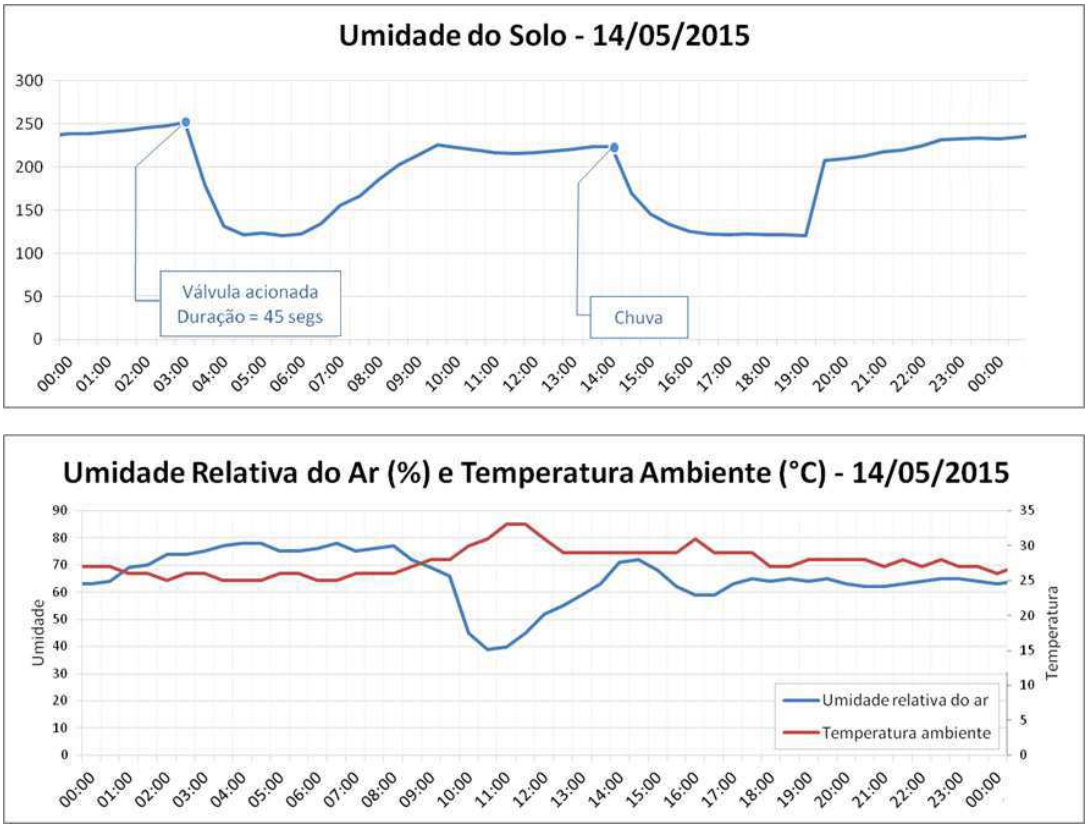


Figura 29: Gráficos referentes à umidade do solo, umidade relativa do ar e temperatura ambiente no dia 14/05/2015.

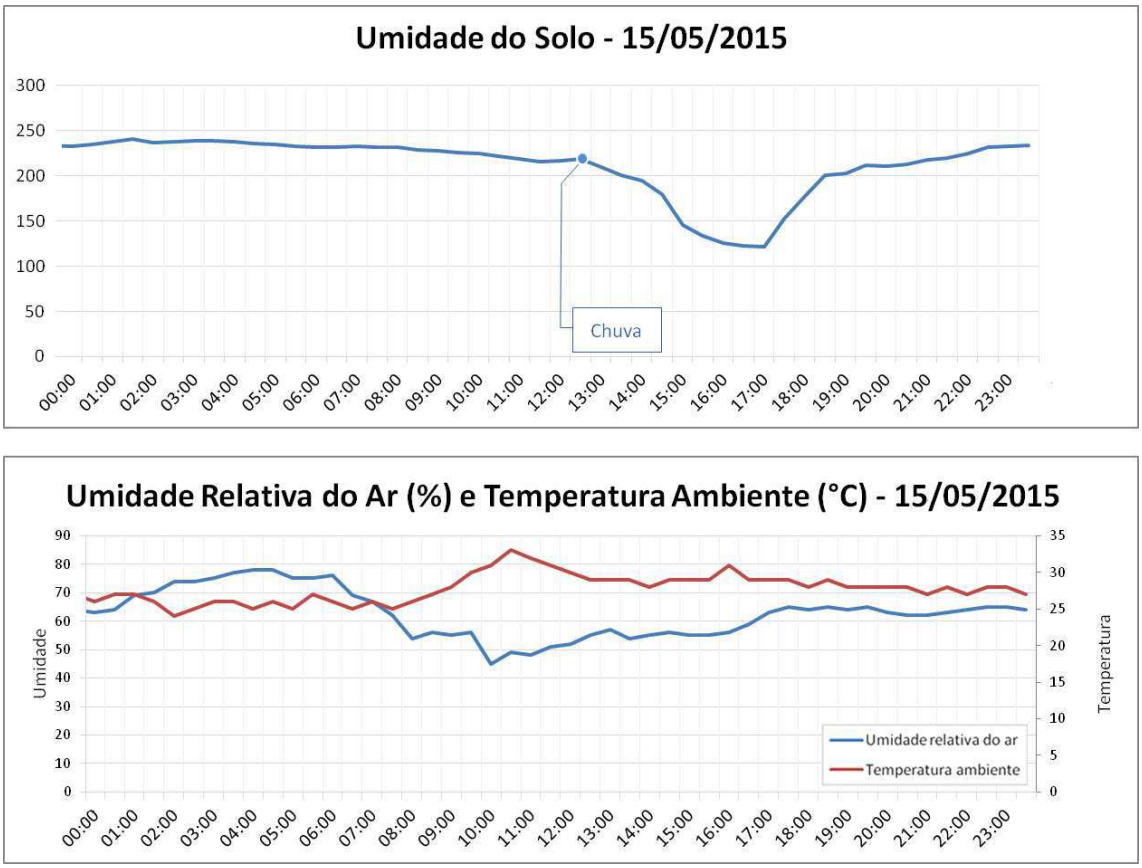


Figura 30: Gráficos referentes à umidade do solo, umidade relativa do ar e temperatura ambiente no dia 15/05/2015.

Analisando os gráficos obtidos, foi possível observar sucesso no funcionamento do protótipo, haja vista que o nível de umidade do solo foi mantido dentro dos limites estipulados previamente. É importante ressaltar que, em dias em que ocorreu chuva, como esperado, não foi necessário o acionamento da válvula. Outro fator que merece destaque foi o volume de água usado: em três dias, o tempo em que a válvula ficou ativa foi inferior a 2 (dois) minutos, a uma vazão de 0,3 l/min, totalizando um gasto de 0,5 litros de água.

O monitoramento do sistema e armazenamento de dados no cartão de memória SD se provou extremamente útil para uma melhor apresentação do comportamento do protótipo, validando-o e documentando seu comportamento.

6 CONCLUSÕES

Neste trabalho foi realizado um projeto de um sistema de controle de irrigação residencial utilizando um Arduino como unidade central de processamento, além de periféricos como sensores e atuadores. Merece destaque o desenvolvimento e confecção de uma *shield* própria, englobando todos os periféricos a serem conectados ao Arduino, conferindo ao sistema projetado portabilidade e modularidade.

Foi desenvolvido um protótipo para proporcionar a validação do sistema projetado, que teve como objetivo realizar a irrigação automática de uma planta de pequeno porte. O sistema proporcionou ao seu usuário praticidade, ao passo em que é autônomo e capaz de monitorar e controlar a irrigação da planta sozinho, além do seu principal objetivo que foi a economia de água, com retorno financeiro para seu proprietário e ecológico para o meio-ambiente. Além disso, o monitoramento das variáveis do ambiente que cercam a planta se mostrou uma ferramenta útil, proporcionando ao usuário vitais informações que podem auxiliá-lo no cultivo do jardim.

O projeto desenvolvido atingiu sua meta de ser de baixo-custo, pois seus gastos, onde incluem-se Arduino, *ethernet shield*, componentes e manufatura da *shield* "TCC", foram inferiores a R$ 200,00, podendo ter seu valor reduzido ainda mais em caso de produção em larga escala. Desta forma, considerando-se os fins ao qual o projeto se propôs, pode-se afirmar que obteve-se êxito em todo o processo de desenvolvimento, contemplando todos os pontos necessários para tal.

Para projetos futuros, existem alguns pontos no sistema que serão melhorados, são eles:

- melhor aproveitamento da *shield ethernet*, através da criação de uma interface via *web* com monitoramento de dados e controle do sistema;
- criação de um aplicativo para Android, melhorando a interface do sistema com o *smartphone*;
- ampliação para um sistema com mais válvulas e sensores, capaz de lidar com uma quantidade maior de plantas independentemente.

O presente trabalho aborda diversos aspectos de um projeto de engenharia elétrica, ao se debater sobre temas como eletrônica e microeletrônica, arquiteturas de

sistemas digitais e programação de processadores. Também contempla a montagem experimental, resultando no desenvolvimento de um protótipo que pode ser usado como modelo para o desenvolvimento de outros produtos, e no levantamento para construção de protótipos e atividades experimentais, todos esses aspectos importantes no desenvolvimento e formação de um profissional de engenharia elétrica.

7 BIBLIOGRAFIA

[1] **Relatório da ONU alerta para possível crise mundial de água**. Disponível em: http://veja.abril.com.br/noticia/ciencia/relatorio-da-onu-alerta-para-possivel-crise-mundial-de-agua/. Acesso em: 29/03/2015.

[2] Revista VEJA. **Um lar que obedece ao dono**. Edição 2299, 12 de dezembro de 2012, p. 176-178.

[3] MEDEIROS, H. **Casa do futuro**. Revista Téchne. 2009. Disponível em: http://www.aureside.org.br/artigos/techne.pdf. Acesso em: 20/10/2010.

[4] TEZA V. **Alguns aspectos sobre a automação residencial – domótica**. 2002. 106 p. Dissertação (Mestrado em Ciências da Computação). Universidade Federal de Santa Catarina, Florianópolis.

[5] Abinee. **ISC Brasil complete 10 anos com novidades para um mercado em crescimento**. Disponível em: http://www.abinee.org.br/noticias/com344.htm. Acesso em 12/05/2015.

[6] CAVALCANTI, Thales. **Integração de Sistemas Residenciais**. Aureside. Disponível em: http://www.aureside.org.br/imprensa/default.asp?file=50.asp. Acesso em 12/05/2015.

[7] **Economia de água no jardim**. Disponível em: http://www.jardimdasideias.com.br/505-%20economia_de_agua_no_jardim. Acesso em 12/05/2015.

[8] **Como reduzir o consumo de água doce ?** Disponível em: http://www.cultivando.com.br/saude_meio_ambiente_agua_reduzindo_o_consumo.html. Acesso em 12/05/2015.

[9] KUSHNER, David. **The Making of Arduino**. IEEE Spectrum. Disponível em: http://spectrum.ieee.org/geek-life/hands-on/the-making-of-arduino. Acesso em13/05/2015.

[10] **Arduino Uno**. Disponível em: http://www.arduino.cc/en/main/arduinoBoardUno. Acesso em 14/05/2015.

[11] McROBERTS, M. **Arduino básico**. Trad. Rafael Zanolli. São Paulo: Novatec, 2011.

[12] ATMEL Corporation. **ATmega48A/48PA/88A/88PA/168A/168PA/328/328P Data Sheet**. [S.l.] 2010.

[13] **A short introduction to the Processing software and projects from the community**. Disponível em: https://processing.org/overview/. Acesso em 15/05/2015.

[14] **Arduino Ethernet Shield**. Disponível em: http://www.arduino.cc/en/Main/ArduinoEthernetShield. Acesso em 15/05/2015.

[15] **Ethernet Library**. Disponível em: http://www.arduino.cc/en/reference/ethernet. Acesso em 15/05/2015.

[16] Guangzhou HC Information Technology Co. **HC-06 Product Data Sheet**. [S.l] 2011.

[17] **Moisture Sensor (SKU:SEN0114)**. Disponível em: http://www.dfrobot.com/wiki/index.php?title=Moisture_Sensor_(SKU:SEN0114). Acesso em 15/05/2015.

[18] D-Robotics UK. **DHT11 Umidity & Temperature Sensor Data Sheet**. [S.l.] 2010.

[19] PEREIRA, Fábio. **Microcontroladores PIC. Programação em C**. 2 ed. São Paulo: Érica, 2003.

[20] **W09 1/2" PE Impulse Solenoid Valve**. Disponível em: http://www.dx.com/p/w09-1-2-pe-impulse-solenoid-valve-white-silver-229707. Acesso em 16/05/2015.

[21] Texas Instruments Incorporated. **L293, L293D Quadruple Half-H Drivers**. [S.l.] 2004.

[22] Maxim Integrated/Dallas Semicondutor. **DS1302 Trickle-Charge Timekeeping Chip**. [S.l.] 2015

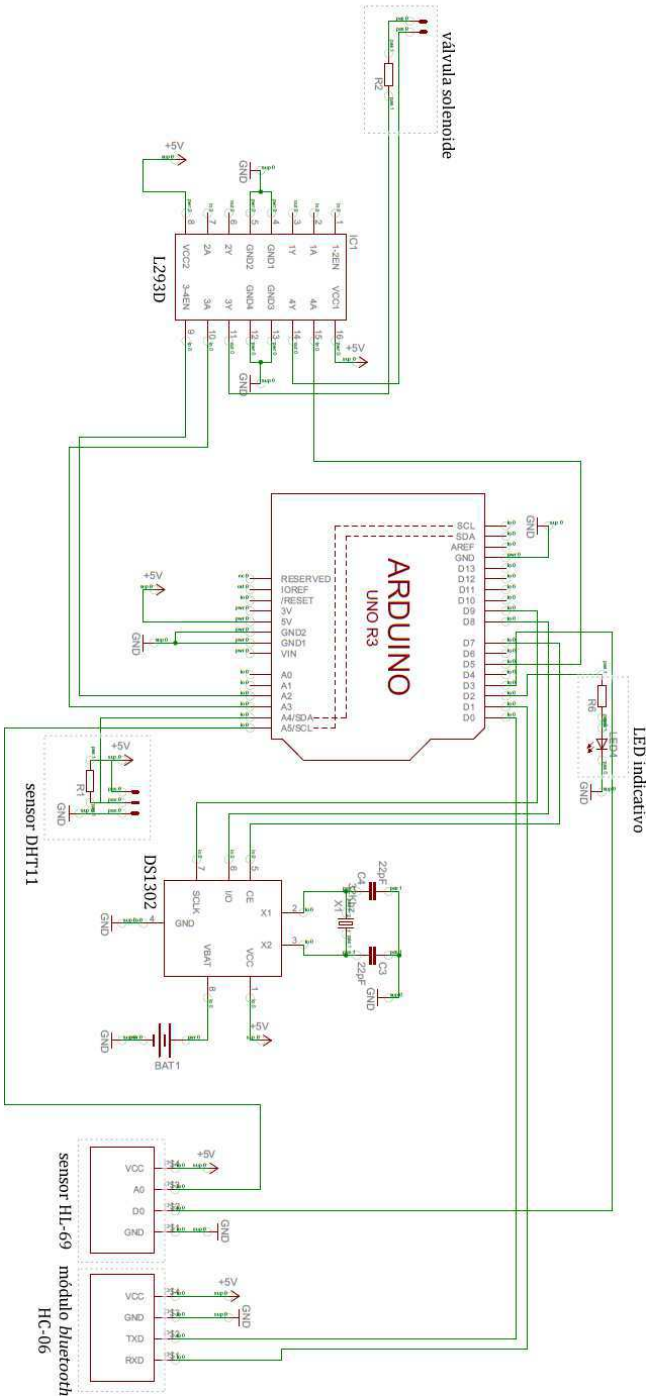
[23] **DS1302 Real Time Clock**. Disponível em: http://playground.arduino.cc/Main/DS1302. Acesso em 16/05/2015.

[24] **About EAGLE PCB Design Software**. Disponível em: http://www.cadsoftusa.com/eagle-pcb-design-software/about-eagle/. Acesso em 18/04/2015.

[25] LPKF Laser & Electronics. **Manual ProtoMat S42**. [S.l.] 2006.

8 ANEXOS

Esquema elétrico do protótipo



Lista de Materiais

1. **Arduino UNO R3**
2. **Arduino *Ethernet Shield***
3. **Módulo *Bluetooth* HC-06**
4. **Sensor de umidade e temperatura DHT11**
5. **Sensor de umidade do solo HL-69**
6. **Válvula solenoide W09**
7. **Ponte H L293D**
8. **RTC DS1302**

Código-fonte

```
// Trabalho de Conclusão de Curso - versão final
// Título: Sistema de Controle de Irrigação
// Autor: Cândido da Nóbrega Ferreira Neto
// Contato: candidoneto89@gmail.com

// ÚLTIMA ATUALIZAÇÃO: 17/05/2015

// Descrição: a cada 30 minutos, a leitura dos sensores é salva no car
tão de memória.
//            Caso a leitura do sensor de umidade do solo ultrapasse 2
50, ativa a válvula para irrigação.
// Comandos bluetooth: v - abre/fecha válvula (manual)
//                     d -
 realiza leitura dos sensores e salva log no SD
//                     s -
 realiza leiture dos sensores e exibe no terminal Bluetooth

// Bibliotecas utilizadas:
#include "DHT.h"
#include <SPI.h>
#include <SD.h>
#include <virtuabotixRTC.h>

// Definição de pinos utilizados pelo sensor DHT11 e pelo DS1302
#define DHTPIN A4
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
virtuabotixRTC myRTC(9, 8, 7); // myRTC(clock, data, rst)

// Definição das variáveis globais utilizadas no programa
char buffer;
int flag=0,flag30min=1,valvula=0,h,t,hora,minuto,segundo,dia,mes,ano;

const int chipSelect = 4; // Pino D4 ligado ao chipSelect do cartão SD
na Ethernet Shield

int enable = A2; // Pino A2 ligado ao pino 9 da ponte H
int In3 = A3; // Pino A3 ligado ao pino 10 da ponte H
int In4 = 5; // Pino D5 ligado ao pino 15 da ponte H

int led = 2; // Pino D2 ligado ao LED de indicação

int SoloHumidA0 = A5; // Pino A5 ligado ao pino A0 do sensor de
umidade do solo YL-100
int SoloHumidD0 = 3; // Pino D3 ligado ao pino D0 do sensor de umidade
do solo YL-100

// Configurações de SETUP inicial
void setup() {
  // Define os pinos como saída ou entrada
  pinMode(In3, OUTPUT);
  pinMode(In4, OUTPUT);
  pinMode(enable, OUTPUT);
  pinMode(led, OUTPUT);

  pinMode(DHT11, INPUT);
```

```
  pinMode(SoloHumidD0, INPUT);
  pinMode(SoloHumidA0, INPUT);

  pinMode(10, OUTPUT);

  // Declara as funções que serão usadas pelo programa
  abrirValvula();
  fecharValvula();
  dhtRead();
  atualizaHora();
  datalog();
  BTReadSensors();

  // Inicializa o sensor DHT11 e a comunicação serial
  dht.begin();
  Serial.begin(9600);
  Serial.println("Inicializando sistema...");
  // Realiza a checagem se o cartão de memória está no slot da shield
Ethernet
  if (!SD.begin(chipSelect)) {
    Serial.println("Cartão SD falhou!");
    return;
  }
  Serial.println("Cartão SD inicializado.");

  // Comentar o código abaixo após a primeira gravação no Arduino
(configura a hora do DS1302)
  // (segundos, minutos, hora, dia da semana, dia do mes, mes, ano)
  // myRTC.setDS1302Time(00, 23, 8, 3, 13, 05, 2015);
  Serial.println("Sistema inicializado.");
}

// Programa que fica se repetindo
void loop() {
  atualizaHora(); // lê as informações de hora e data fornecidas pelo
DS1302
// caso tenha se passado 30 minutos, realiza a leitura dos sensores e
salva esses dados no cartão SD
  if (flag30min) {
    if ((minuto == 30) || (minuto == 0)) {
      datalog();
      flag30min=0;
    }
  }
  else if (!flag30min) {
    if ((minuto != 30) && (minuto != 0)) {
      flag30min=1;
    }
  }

  digitalWrite(enable, HIGH);
  flag=0;

  // Lê os comandos enviados via bluetooth através do celular:
  if(Serial.available() > 0)
  {
    buffer = Serial.read();
    flag=1;
  }
  if (buffer == 'v') {
    if (flag) {
```

```
        if (valvula) fecharValvula();
        else if(!valvula) abrirValvula();
        flag=0;
      }
  }
  else if (buffer == 'd') {
    if (flag) {
      datalog();
      flag=0;
    }
  }
  else if (buffer == 's') {
     if (flag) {
       BTReadSensors();
       flag=0;
     }
  }
  else {
      digitalWrite(In3, LOW);
      digitalWrite(In4, LOW);
   }

  if(analogRead(SoloHumidA0)>250) abrirValvula();
  else if(analogRead(SoloHumidA0)<120) fecharValvula();
}

// Funções utilizadas no código principal:

void abrirValvula() { // Abre a válvula
  digitalWrite(In3, HIGH);
  digitalWrite(In4, LOW);
  delay(30);
  digitalWrite(In3, LOW);
  digitalWrite(In4, LOW);
  digitalWrite(led, HIGH);
  valvula=1;
}

void fecharValvula() { // Fecha a válvula
  digitalWrite(In3, LOW);
  digitalWrite(In4, HIGH);
  delay(30);
  digitalWrite(In3, LOW);
  digitalWrite(In4, LOW);
  digitalWrite(led, LOW);
  valvula=0;
}

void dhtRead() { // Armazena os valores lidos do sensor DHT11
  h = dht.readHumidity();
  t = dht.readTemperature();
}

void atualizaHora() { // Lê os dados do relógio DS1302 e os armazena
nas variáveis correspondentes
  myRTC.updateTime();
  hora = myRTC.hours;
  minuto = myRTC.minutes;
  segundo = myRTC.seconds;
  dia = myRTC.dayofmonth;
  mes = myRTC.month;
```

```
  ano = myRTC.year;
}

void datalog() {
  // Realiza a leitura dos sensores e armazena esses dados
  // no cartão de memória, em um arquivo denominado data.txt
  atualizaHora();
  dhtRead();
  File dataFile = SD.open("data.txt", FILE_WRITE);
  if (dataFile) {
    dataFile.print("Data: ");
    if (dia<10) dataFile.print("0");
    dataFile.print(dia);
    dataFile.print("/");
    if (mes<10) dataFile.print("0");
    dataFile.print(mes);
    dataFile.print("/");
    if (ano<10) dataFile.print("0");
    dataFile.print(ano);
    dataFile.print(" Hora: ");
    if (hora<10) dataFile.print("0");
    dataFile.print(hora);
    dataFile.print(":");
    if (minuto<10) dataFile.print("0");
    dataFile.print(minuto);
    dataFile.print(":");
    if (segundo<10) dataFile.print("0");
    dataFile.println(segundo);

    if (valvula) dataFile.println("Válvula: aberta");
    if (!valvula) dataFile.println("Válvula: fechada");

    dataFile.print("Umidade solo: ");
    dataFile.println(analogRead(SoloHumidA0));

    dataFile.print("Umidade ambiente: ");
    dataFile.print(h);
    dataFile.print(" % ");
    dataFile.print("Temperatura ambiente: ");
    dataFile.print(t);
    dataFile.println(" *C");
    dataFile.println("");
    dataFile.close();

    Serial.println("Dados armazenados.");
  }
  else {
    Serial.println("Erro criando data.txt");
  }
}

void BTReadSensors() { // Realiza a leitura dos sensores e envia estes
dados via Bluetooth
  dhtRead();
  atualizaHora();
  Serial.print("Data: ");
  if (dia<10) Serial.print("0");
  Serial.print(dia);
  Serial.print("/");
  if (mes<10) Serial.print("0");
  Serial.print(mes);
```

```
  Serial.print("/");
  if (ano<10) Serial.print("0");
  Serial.print(ano);
  Serial.print(" Hora: ");
  if (hora<10) Serial.print("0");
  Serial.print(hora);
  Serial.print(":");
  if (minuto<10) Serial.print("0");
  Serial.print(minuto);
  Serial.print(":");
  if (segundo<10) Serial.print("0");
  Serial.println(segundo);

  if (valvula) Serial.println("Válvula: aberta");
  if (!valvula) Serial.println("Válvula: fechada");

  Serial.print("Umidade solo: ");
  Serial.println(analogRead(SoloHumidA0));

  Serial.print("Umidade ambiente: ");
  Serial.print(h);
  Serial.print(" % ");
  Serial.print("Temperatura ambiente: ");
  Serial.print(t);
  Serial.println(" *C");
  Serial.println("");
}
```