UNIVERSIDADE DE SÃO PAULO

ESCOLA DE ENGENHARIA DE SÃO CARLOS

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

# TRABALHO DE CONCLUSÃO DE CURSO

# “Desenvolvimento de uma estufa controlada e monitorada remotamente”

BEATRIZ MIDENA CAPELLI

São Carlos – SP

Novembro/2014

**BEATRIZ MIDENA CAPELLI**

# DESENVOLVIMENTO DE UMA ESTUFA CONTROLADA E MONITORADA REMOTAMENTE

Trabalho de Conclusão de Curso apresentado

à Escola de Engenharia de São Carlos,

da Universidade de São Paulo

Curso de Engenharia de Computação

Orientador: Evandro Luis Linhari Rodrigues

São Carlos

2014

M236d

Midena Capelli, Beatriz
Desenvolvimento de uma estufa controlada e monitorada remotamente / Beatriz Midena Capelli; orientador Evandro Luís Linhari Rodrigues. São Carlos, 2014.

Monografia (Graduação em Engenharia de Computação) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2014.

1. Arduino. 2. Estufa. 3. Ubuntu. I. Título.

# FOLHA DE APROVAÇÃO

**Nome:** Beatriz Midena Capelli

**Título:** "Desenvolvimento de uma estufa automática controlada remotamente"

**Trabalho de Conclusão de Curso defendido em** 17/11/2014.

| **Comissão Julgadora:** | **Resultado:** |
|---|---|
| Prof. Associado Evandro Luís Linhari Rodrigues (Orientador) - SEL/EESC/USP | Aprovada |
| Prof. Dr. João Navarro Soares Júnior SEL/EESC/USP | APROVADO |
| Prof. Associado Adilson Gonzaga SEL/EESC/USP | APROVADA |

***Coordenador do Curso Interunidades - Engenharia de Computação:***

Prof. Associado Evandro Luís Linhari Rodrigues

## Resumo

Este trabalho foi motivado pela crescente utilização da Internet no dia a dia das pessoas nos últimos anos, além da grande disponibilidade de módulos eletrônicos comercializados ao redor do mundo a baixo custo. A união desses dois fatos favorece a construção de protótipos automatizados conectados à rede. Neste documento é detalhada a criação de uma estufa automatizada, controlada remotamente pela Internet através de uma página *online*. O sistema foi construído em módulos sendo baseado na utilização do sistema operacional Ubuntu 12.04 para disponibilizar o acesso remoto via internet, e plataforma Arduino para controlar uma série de sensores e atuadores. As formas de comunicação entre os módulos, o desenvolvimento da página *web* e a montagem em geral atenderam aos requisitos da proposta inicial, permitindo assim a conclusão com êxito do projeto.

Palavras-chave: Linux, Estufa, Arduino, Automação via *web*.

## *Abstract*

This work has been motivated by the expanding daily use of the Internet by people in recent years, in addition to the wide availability of low-cost electronic modules traded all over the world. These two factors combined promote the building of network-connected automated prototypes. This paper details the creation of an automated greenhouse, which is remotely controlled by an online webpage. The system was built in modules, being based in the Ubuntu 12.04 operational system and the Arduino plataform in order to control a series of sensors and actuators. The means of communication between the modules, the developing of the webpage and the assembly in general met the requirements of the initial proposal, thus enabling the successful completion of the project.

Keywords: Linux, Greenhouse, Arduino, Web Automation

# Lista de Abreviaturas

| | |
|---|---|
| **Web** | *"World Wide Web"* – Sistema hipertextual que opera através da Internet. |
| **RISC** | *"ReducedInstruction Set Computer"* – Computador com um conjunto reduzido de instruções. |
| **EEPROM** | *"Electrically-ErasableProgrammableRead-OnlyMemory"* – *Chip* de armazenamento não volátil, que pode ser apagado eletricamente. |
| **SRAM** | "*StaticRandom Access Memory*" – Memória estática de acesso aleatório, usada em memórias cache. |
| **USART** | "*Universal SynchronousAsynchronousReceiverTransmitter*" – Transmissor/Receptor Universal Síncrono e Assíncrono (comunicação serial). |
| **SPI** | "*Serial Peripheral Interface*"– Protocolo de comunicação entre o microcontrolador e outros componentes. |
| **A/D** | Analógico-Digital. |
| **IDEs** | "*IntegratedDevelopmentEnvironment*" – Ambiente de desenvolvimento integrado. |
| **ROM** | "*ReadOnlyMemory*" – Memória apenas de leitura, mantém os programas. |
| **E/S** | Entrada e saída |

# Sumário

# 1. Introdução

## 1.1. Objetivo

O objetivo deste trabalho é propor a construção de uma estufa para plantas, cujas variáveis do ambiente e desenvolvimento do vegetal sejam controlados remotamente pela Internet através de sensores, atuadores e uma *webcam* conectados a dois dispositivos eletrônicos principais.

## 1.2. História

O presente trabalho, inicialmente desenvolvido durante o segundo semestre de 2013 no âmbito da disciplina SEL0630 – Aplicações de Microprocessadores II, consiste na automatização de uma estufa utilizando diferentes tecnologias que englobam desde projeto e esquematização de *hardware* até aplicações *web*.

A proposta na disciplina era a de construir uma estufa automática, monitorada e controlada remotamente. As funcionalidades escolhidas para o projeto foram: Sensoriamento remoto de temperatura, umidade do ar, umidade do solo e nível de água no reservatório, além de controle remoto da iluminação, irrigação e temperatura. Todos esses periféricos estavam ligados à dois “cérebros”: A plataforma Arduino e um computador equipado com o sistema operacional Ubuntu 12.04 e conectado à Internet.

O Arduino controlava a leitura dos sensores, além de se comunicar com o computador via comunicação serial para receber os comandos aos atuadores. O computador mantinha a página *web* no ar, recebendo comandos enviados pelo usuário através de um painel de controle, e os repassando ao Arduino. Mais tarde foi inserida uma *webcam* no projeto, também controlada pelo computador.

O primeiro protótipo da estufa – construído durante SEL630 – implementava quase todas as funcionalidades descritas nos parágrafos anteriores, com exceção da iluminação e do controle da temperatura, porém o protocolo de comunicação desenvolvido não estava funcionando corretamente, apresentando vários erros graves (perda de sincronismo, comandos recebidos pela metade, etc.). Esse protótipo estava

inteiramente desenvolvido sobre uma matriz de contatos (*protoboard*), o que dava um aspecto amador ao projeto.

Durante o primeiro semestre desse Trabalho de Conclusão de Curso, o objetivo era terminar de construir as funcionalidades citadas – adicionar uma lâmpada apropriada e desenvolver o controle de temperatura. Feito isso, os esforços seriam concentrados na implementação de um protocolo de comunicação funcional e confiável. Com todos os periféricos funcionando, e o Arduino se comunicando perfeitamente com o computador, o próximo passo seria a fabricação de um circuito impresso para diminuir ao máximo o espaço necessário para a eletrônica do projeto. Para melhorar a organização do projeto e, eventualmente, facilitar a construção do circuito impresso, foi desenvolvido o esquemático completo dos periféricos na ferramenta Eagle [1]. Cada periférico possui as indicações de conexão com os pinos de alimentação e controle do Arduino.

Para o segundo semestre, as tarefas foram um pouco diferentes: Com *hardware* e *softwares* prontos, os problemas passam ao próximo nível: biológico. A estufa teve que ser preparada para abrigar todo o desenvolvimento de uma planta. Exemplos de questões a serem abordadas nessa época foram vedação do reservatório de água, adaptação da mangueira de irrigação para melhor eficiência, escolha de um vegetal que conseguisse sobreviver nas condições fornecidas, preparação do solo, preparação dos parâmetros do ambiente, etc. Após a plantação, uma foto do vegetal foi capturada diariamente através da *webcam* para construção de um *timelapse*[1] do seu crescimento.

## 1.3. Motivação

A ideia do trabalho pode se apoiar no fato de que a tecnologia está cada vez mais presente no dia-a-dia das pessoas. Aquelas que possuem alguma plantação que precisa de cuidados diários e eventualmente precisam viajar, por exemplo, podem se utilizar do projeto para manter plantas em um ambiente com temperatura adequada e umidade e iluminação controladas pela Internet, além de poder requisitar fotos através de uma *webcam*.

[1] Vídeo que reúne várias fotos tiradas em sequência e na mesma posição, que vão dar a impressão de um vídeo acelerado do desenvolvimento da planta.

## 1.4. Organização do trabalho

Esta monografia será dividida da seguinte forma: O Capítulo 2 descreve o Embasamento Teórico do trabalho, o Capítulo 3 detalha os Materiais e Métodos utilizados, apresentando todos os sensores, atuadores e detalhes do projeto. O Capítulo 4 detalha a implementação tanto do *hardware,* quanto dos *softwares* do Arduino e do computador, e discussões acerca dos Resultados obtidos. No Capítulo 5, finalmente, encontram-se as Conclusões acerca dos trabalhos realizados. Ao final da monografia estão as Referências Bibliográficas citadas e os Apêndices.

# 2. Embasamento Teórico

Os dois “cérebros” que controlam todas as funcionalidades da estufa são: Um microcontrolador, cujos pinos de E/S controlam sensores e atuadores digitais e pinos de comunicação serial que conversam com uma interface serial em um computador; esse computador *(um notebook* convencional), onde um programa em Python é executado, cria uma interface serial sobre o sistema operacional Ubuntu 12.04. Ao mesmo tempo ele é o responsável pela conexão com a Internet. Esses dispositivos serão detalhados abaixo.

## 2.1. Computador com Ubuntu 12.04

Um computador pode ser dividido em duas partes: *Hardware* e *Software*. O *hardware* é o conjunto de todas as peças eletrônicas, como memórias, disco rígido, etc. O *software* é o conjunto de programas que roda em cima dessas peças e faz a interface entre a eletrônica e o usuário.

O *software* mais importante de um computador é o Sistema Operacional. Ele é uma coleção de programas para gerenciar as funções do processador, como a entrada e saída de informações, o controle de dispositivos, o armazenamento de programas, etc. [2].

O Ubuntu 12.04 [3] é um sistema operacional baseado em Linux desenvolvido por programadores voluntários. Ele é *open-source*, ou seja, gratuito, e para sistemas embarcados oferece maior flexibilidade para programação: toda a programação em Windows tem que ser feita por meio de IDEs, como o acesso a *webcam* por exemplo. No Ubuntu 12.04, o acesso é imediato (no caso da *webcam*, o diretório “/dev/video0” mantém a conexão). No projeto, o sistema operacional foi instalado em um *notebook* convencional.

A função do Ubuntu 12.04 no projeto é manter um programa em Python executando e a página *web online*. O programa em Python estabelece comunicação serial com o microcontrolador (vide seção 2.2) – através da porta serial criada automaticamente por ele, ao ser conectado ao *notebook*, e permanece recebendo as informações dos sensores (como a temperatura do ambiente) e enviando comandos

relativos às alterações causadas por alterações nos botões *online* (como a irrigação das plantas).

O *notebook* equipado com Ubuntu 12.04 utilizado no projeto, que executa o programa em Python e mantém o servidor *web* será a partir de agora chamado de "computador".

## 2.2. Microcontrolador ATMega328

Com um microcontrolador pode-se construir um computador. Todos os computadores (independentemente de ser um computador de mesa, ou um grande *mainframe*) possuem várias características em comum: Todos possuem uma Unidade Central de Processamento, ou CPU, memórias, tanto de dados quanto de programa e dispositivos de entrada e saída de informações. No caso de um computador de mesa, um exemplo de dispositivo de entrada é o teclado, e um de saída, o monitor. No caso de um microcontrolador, um dispositivo de entrada pode ser um sensor, e o de saída, um atuador qualquer [4].

Microcontroladores são produzidos na forma de "*chips*" conforme visto na Figura 1. Esses *chips* possuem pinos dedicados à alimentação, à gravação de programas em sua memória ROM e pinos reservados aos periféricos de E/S.

O microcontrolador utilizado no projeto é o ATMega328 da Atmel (Figura 1), que utiliza a microarquitetura Harvard de 8 bits, é RISC, possui 32KB de memória *flash*, 1KB de EEPROM, 2KB de SRAM, 32 registradores de uso geral, 3 temporizadores/contadores, uma USART, portas para comunicação SPI, 6 conversores A/D de 10 bits e um *watchdog timer*, entre outras características. Ele controla a leitura dos sensores e lógica dos atuadores, além da comunicação serial com o computador.

Para facilitar a manipulação desses *chips*, surgiram plataformas que criam a interface entre o *chip* e um computador através de uma porta USB, por exemplo. Assim, a gravação de um novo programa em um microcontrolador pode ser feita em alguns segundos. A plataforma que utiliza o microcontrolador ATMega328 chama-se "Arduino" e será detalhada na seção 3.2. O microcontrolador será chamado de "Arduino" neste documento, por questão de facilidade.

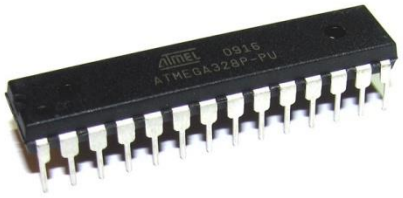


**Figura 1– Microcontrolador ATMega328[5]**

# 3. Materiais e Métodos

O *hardware* da estufa é formado inicialmente por uma caixa de isopor e um tanque de água. A tampa da caixa de isopor foi trocada por uma de madeira para servir como apoio para quase toda a eletrônica utilizada: Um Arduino UNO com ATMega328, um sensor de temperatura e umidade do ar, um sensor de umidade do solo, um sensor de nível de água, uma pastilha de Peltier envolta por dois pares *cooler*-dissipador, três módulos relês, um *display* LCD e uma lâmpada. Além dos itens citados, o projeto conta com uma bomba d'água posicionada no tanque de água, um computador – como descrito na seção 2.1, que pode ser posicionado em qualquer lugar, desde que perto de um cabo de rede ou roteador, e uma *webcam*.

Em questões de *software,* tem-se o Arduino controlando sensores e atuadores e o computador executando um servidor *web* para uma página construída, além de um programa em Python que implementa a comunicação serial. A página recebe e exibe as informações do ambiente lidas pelo Arduino e apresenta botões para controle dos atuadores.

Nas próximas seções são descritas com mais detalhes as funcionalidades de cada componente utilizado no projeto.

## 3.1. Estufa

Na Figura 2 é apresentada a estrutura geral do projeto. As funcionalidades da estufa são listadas na Tabela 1.

.

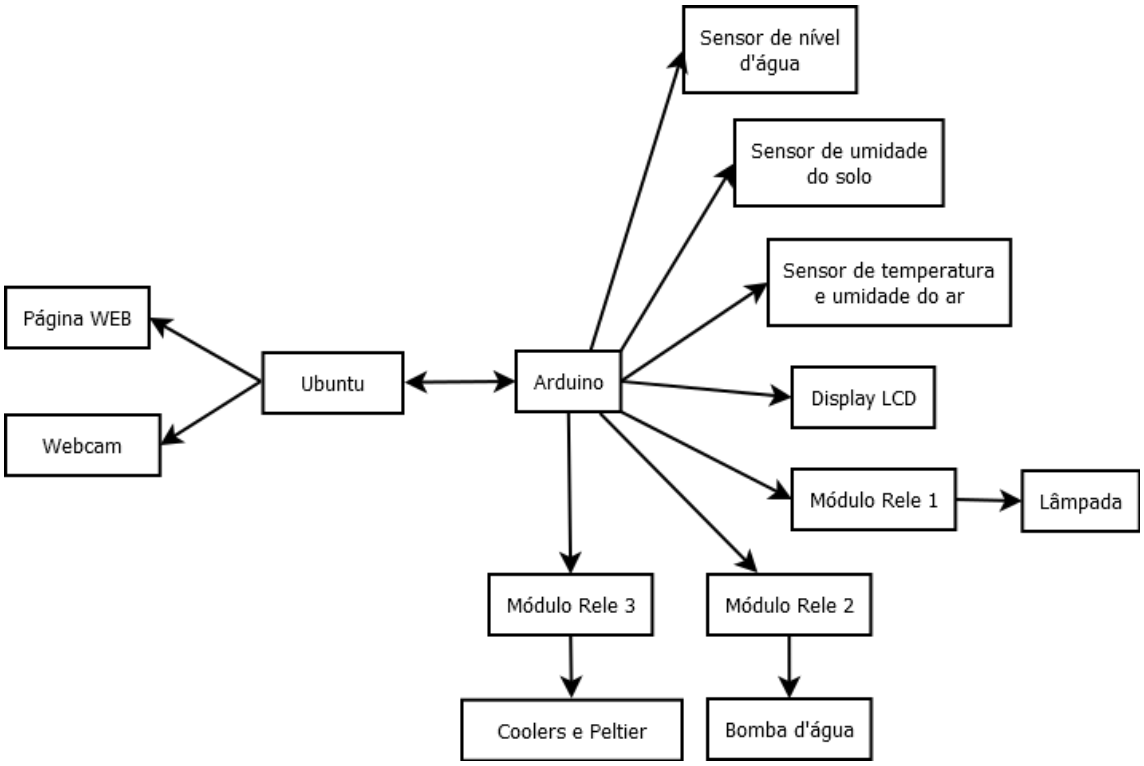


**Figura 2 - Esquema dos componentes do projeto.**

**Tabela 1 - Funcionalidades da estufa.**

| **Funcionalidades** |
|---|
| Ativar a bomba d'água pela página *web*. |
| Acender e apagar a lâmpada pela página *web*. |
| Tirar uma foto do interior da estufa pela página *web*. |
| Aumentar ou diminuir a temperatura desejada pela página *web*. |
| Checar os valores medidos pelos sensores através da página *web*. |
| Checar os valores medidos pelos sensores através do *display* LCD. |

Todos os elementos descritos na Figura 2 são energizados por uma fonte de computador, que possui todas as saídas necessárias para o projeto (5 V, 12 V e 110 ou 220 V – dependendo de onde a mesma está ligada). Um esquemático das conexões da fonte pode ser visto na Figura 3. Todos os pinos de alimentação dos próximos esquemáticos a serem ilustrados se referem a essa fonte.

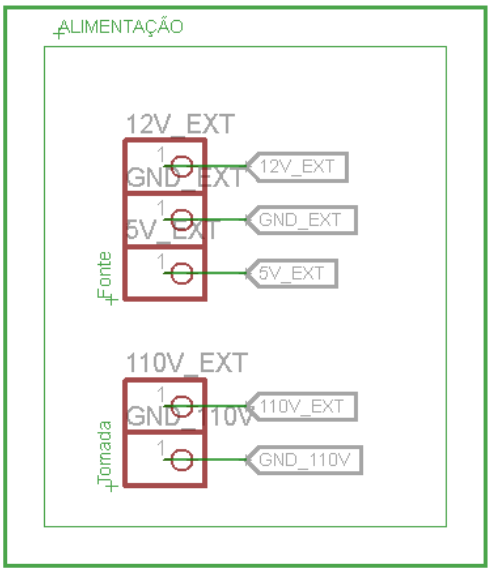


**Figura 3 - Esquemático de conexão da fonte externa.**

O código embarcado no Arduino é um *loop* infinito onde ocorre a leitura dos sensores (ver seção 3.3), o envio e recebimento de mensagens via serial e o controle dos atuadores (ver seção 3.4). No computador, existem dois módulos: Um servidor Apache que mantém a página *web* no ar (ver seção 3.8) e um programa em Python que faz a comunicação serial (ver seção 3.6).

## 3.2. Arduino

O Arduino é uma plataforma de prototipação eletrônica *open-source* flexível que utiliza na versão UNO o microcontrolador ATMega328, que é o modelo utilizado neste projeto e pode ser visto na Figura 4. O Arduino pode receber sinais elétricos de vários sensores e lidar com esses dados para controlar motores, luzes, relês, e quaisquer outros atuadores.


**Figura 4– Arduino UNO. [6]**

O *software* que controla os pinos e ações do Arduino pode ser desenvolvido no programa “Arduino IDE”, disponibilizado para *download* no site oficial, que conta com uma interface gráfica simples e aceita códigos em C/C++.

O código embarcado deve estar no formato “ino” e contar com as seguintes chamadas de procedimento: “setup()”, que ajusta as configurações das portas de entrada e saída e a comunicação serial, e “loop()”, que é um laço infinito onde o usuário cria a rotina do seu programa.

O esquemático da pinagem do Arduino pode ser visto na Figura 5. Os pinos de alimentação são referentes à fonte externa utilizada, descrita na seção 3.1.

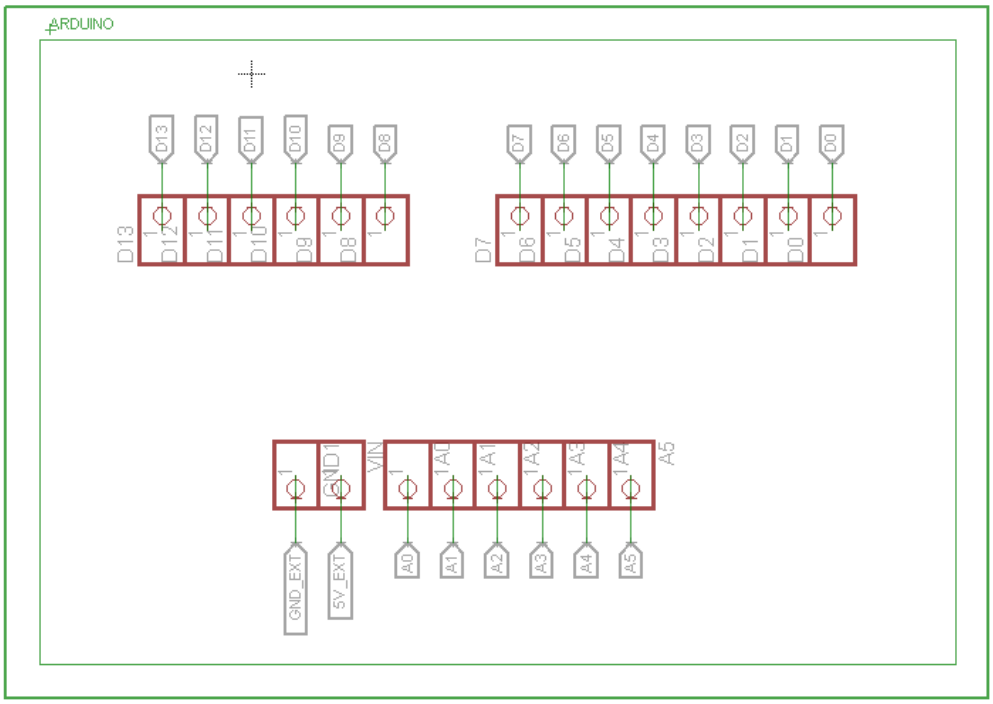


**Figura 5 - Esquemático de pinagem do Arduino.**

## 3.3. Sensores

Sensores são dispositivos eletrônicos que captam informações do ambiente e as transformam em sinais elétricos que podem ser interpretados pelo Arduino.

### 3.3.1. Sensor de temperatura e umidade do ar

O sensor de temperatura e umidade do ar utilizado foi o DHT11, que tem precisão de 1ºC e 0,5% de umidade e pode ser visto na Figura 6. Esse sensor foi escolhido pela existência de bibliotecas já programadas para o Arduino: Um simples comando “DHT11.read()” obtém as medidas do ambiente.

O sensor foi conectado ao pino A0 do Arduino. O esquemático da ligação pode ser visto na Figura 7.

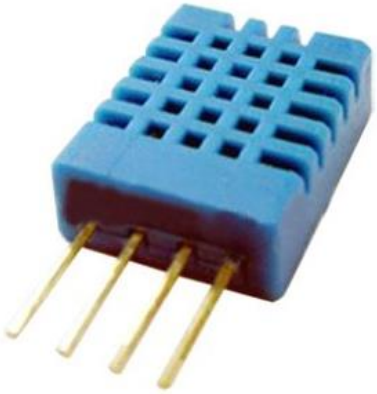

**Figura 6 - Sensor DHT11. [7]**

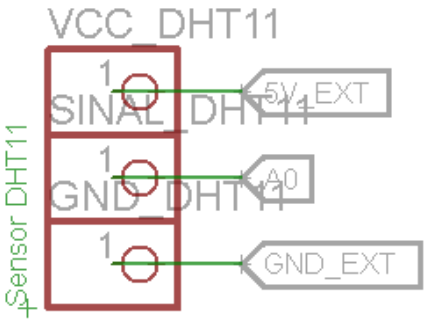


**Figura 7 - Esquemático de conexão do sensor DHT11.**

### 3.3.2. Sensor de nível de água

O sensor de nível de água foi construído baseado no fato de a água conduzir eletricidade: Dois terminais de um circuito elétrico (fios) são posicionados estrategicamente no nível de água que desejamos manter no tanque, indicando a presença ou ausência de corrente elétrica entre eles.

Um resistor de *pull-down*[2] é conectado ao pino D12 do Arduino (entrada digital), mantendo o mesmo em nível lógico 0, e é aplicado nível lógico 1 no pino D13 do Arduino (saída digital). A leitura do pino D12 indica a presença (nível 1) ou ausência (nível 0) de água naquela altura.

Uma imagem que representa o sensor de nível de água pode ser visto na Figura 8. Um esquemático de ligação do circuito pode ser visto na Figura 9.

[2] Resistor de *pull-down* é utilizado para fixar o nível lógico de uma entrada de dados em 0 – se não utilizarmos um desses (ou de *pull-up* – que mantém o nível lógico em 1), a leitura do pino fica comprometida por ruído. O resistor deve ser ligado entre o pino desejado e o GND do circuito.

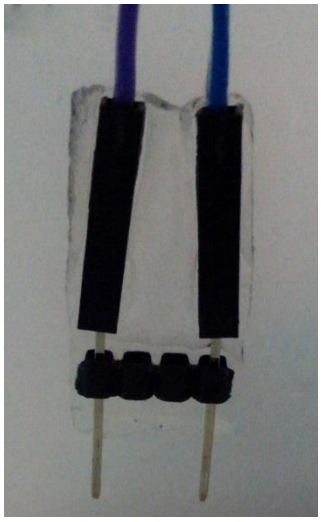

**Figura 8 - Sensor de nível de água.**

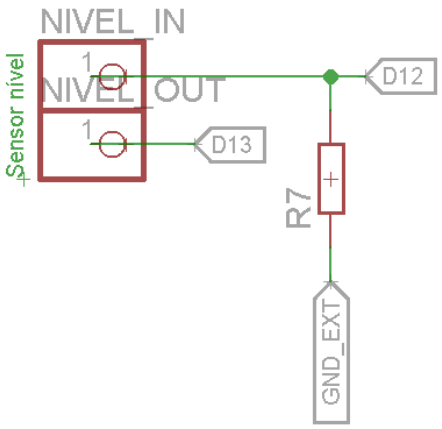


**Figura 9– Esquemático de conexão do sensor de nível de água.**

### 3.3.3. Sensor de umidade do solo

O sensor de umidade do solo segue o mesmo funcionamento do sensor de nível de água, mas, ao invés de dois fios, foram utilizados dois parafusos, que devem ser fixados ao solo e ligados a dois pinos do Arduino (uma entrada analógica com *pull-down* – A2 - e uma saída analógica – A1). Aplicado o nível lógico 1 na saída, a leitura da entrada indica, com valores entre 0 e 1023, a intensidade da umidade do solo.

Uma foto do sensor de umidade do solo pode ser visto na Figura 10. Um esquema de conexão do circuito pode ser visto na Figura 11.

**Figura 10 - Sensor de umidade do solo.**

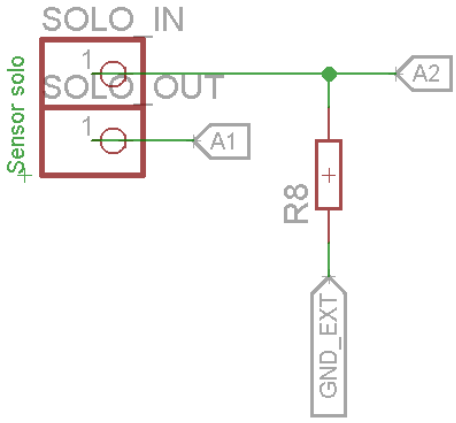


**Figura 11 – Esquemático de conexão do sensor de umidade do solo.**

### 3.4. Atuadores

Atuadores são dispositivos eletrônicos que recebem sinais elétricos e realizam alguma ação no sistema.

Todos os atuadores do sistema funcionam em tensão diferente de 5 V, e necessitam corrente maior do que a máxima oferecida pelo Arduino. Assim, o controle

desses periféricos será feito através de uma fonte externa (retirada de um computador antigo, com saídas de 5 V, 12 V e 110 ou 220 V) e reles.

Reles são interruptores eletromecânicos. Quando uma corrente elétrica percorre as bobinas de um rele, cria um campo magnético que atrai ou repulsa uma chave metálica, que fecha um circuito elétrico. Dessa forma, podem-se isolar tensões diferentes em um mesmo circuito.

No caso do projeto, como o Arduino não fornece corrente suficiente para alimentar as bobinas dos reles, a solução é que ele excite a base de um transistor bipolar, cujo coletor está conectado ao rele, gerando uma corrente maior na bobina do mesmo.

A bobina está em paralelo com um diodo, conhecido como “diodo de roda livre”. Esse diodo serve para a proteção do transistor: Quando a corrente na bobina é interrompida, ela pode gerar picos de tensão em seus terminais, que podem queimar o transistor. O diodo é posicionado de forma que, se houver pico de tensão no coletor do transistor, a corrente passe por ele, e não seja forçada pelo transistor. [8]

A corrente na bobina gera um campo magnético que fecha (mecanicamente) o circuito entre o GND de algum periférico, como a bomba d’água, e o GND da fonte externa, ativando o circuito. O esquemático de conexão de um rele pode ser visto na Figura 12.

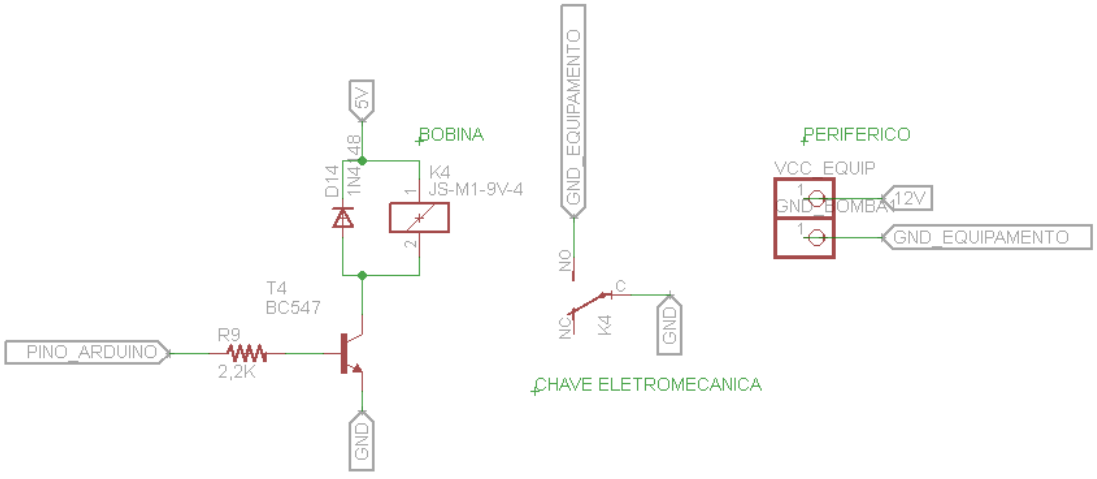


**Figura 12 – Esquemático da conexão de um rele.**

Mais detalhes sobre o funcionamento de cada atuador são dados nas seções a seguir.

### 3.4.1. Luz

Para a lâmpada do sistema foi adquirida uma lâmpada de aquário, com um filamento branco e um filamento azul. Esse modelo foi escolhido pelo fato de a luz azul ser a mais importante no crescimento das plantas, por atuar diretamente na fotossíntese. A luz branca do outro filamento da lâmpada fornece todos os outros espectros necessários para o crescimento da planta: O vermelho é importante para o desenvolvimento de flores, e o verde é importante para dar coloração aos vegetais (eles refletem essa cor, por isso são verdes em sua maioria) [9].

O modelo pode ser visto na Figura 13.

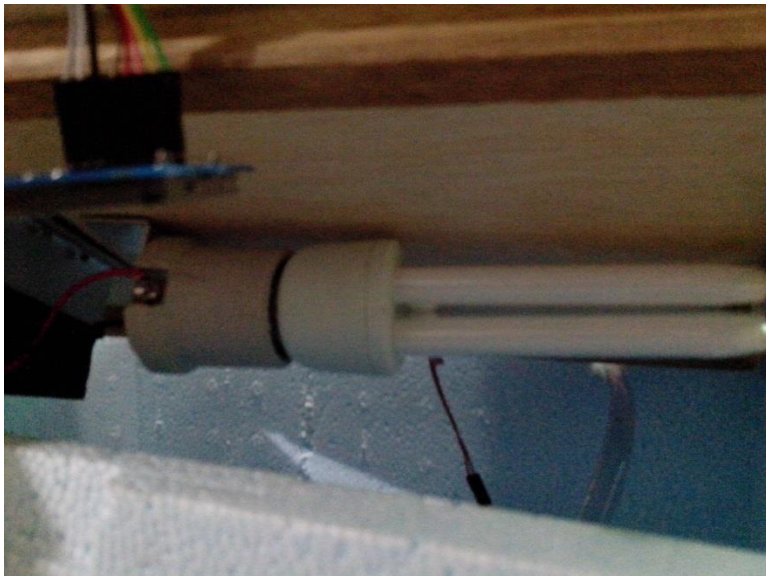

**Figura 13 - Lâmpada adquirida para o projeto.**

A lâmpada é conectada ao Arduino através de um rele, pois funciona em 110 ou 220 V[3]. O circuito de acionamento da lâmpada pode ser visto na Figura 14. O

[3] A lâmpada é feita para funcionar em 220 V, mas não é danificada quando ligada a 110 V – e para o rele é indiferente a tensão controlada.

controle é feito a partir do pino A4 do Arduino. Quando esse pino é “setado” para nível lógico “1”, a lâmpada é ativada.

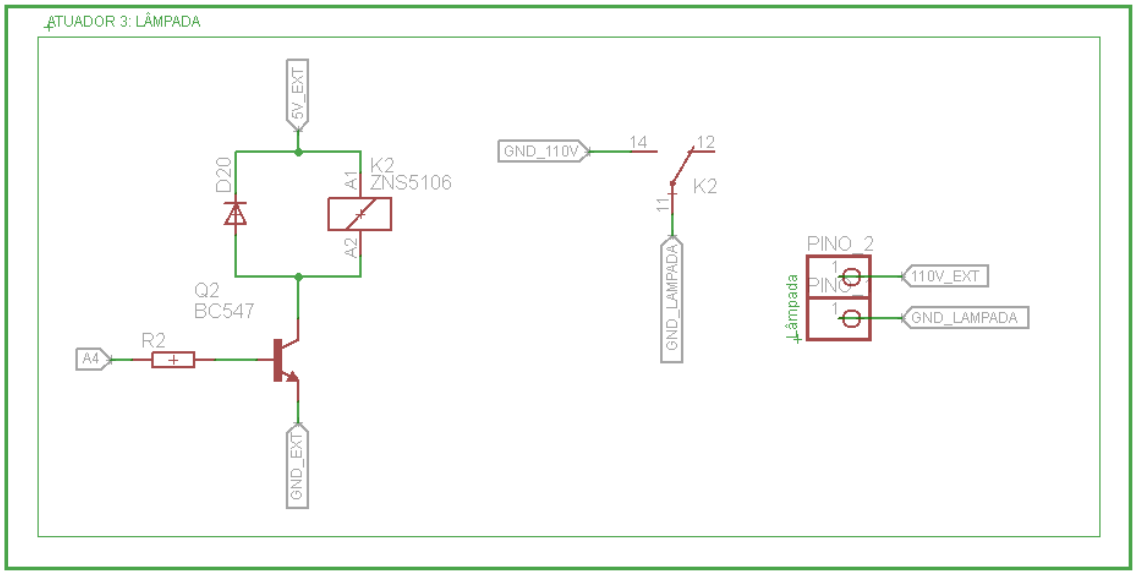


**Figura 14 - Esquemático de conexão do atuador lâmpada.**

O controle da lâmpada é bastante intuitivo: apertar o botão com o desenho de uma lâmpada na página *web* acende ou desliga a luz.

### 3.4.2. Bomba d’água

Para a bomba d’água do sistema foi adquirida uma normalmente utilizada em aquários. O modelo pode ser visto na Figura 15. Assim como a lâmpada, ela é controlada pelo Arduino através de um rele, pois funciona em 12 V.

**Figura 15 - Bomba d'água do sistema. [10]**

O circuito de acionamento da bomba d'água pode ser visto na Figura 16. O controle é feito a partir do pino A5 do Arduino. Quando esse pino é "setado" para nível lógico "1", a bomba d'água é ativada. Para que a irrigação aconteça, uma mangueira que é conectada à bomba d'água e chega ao solo foi instalada.

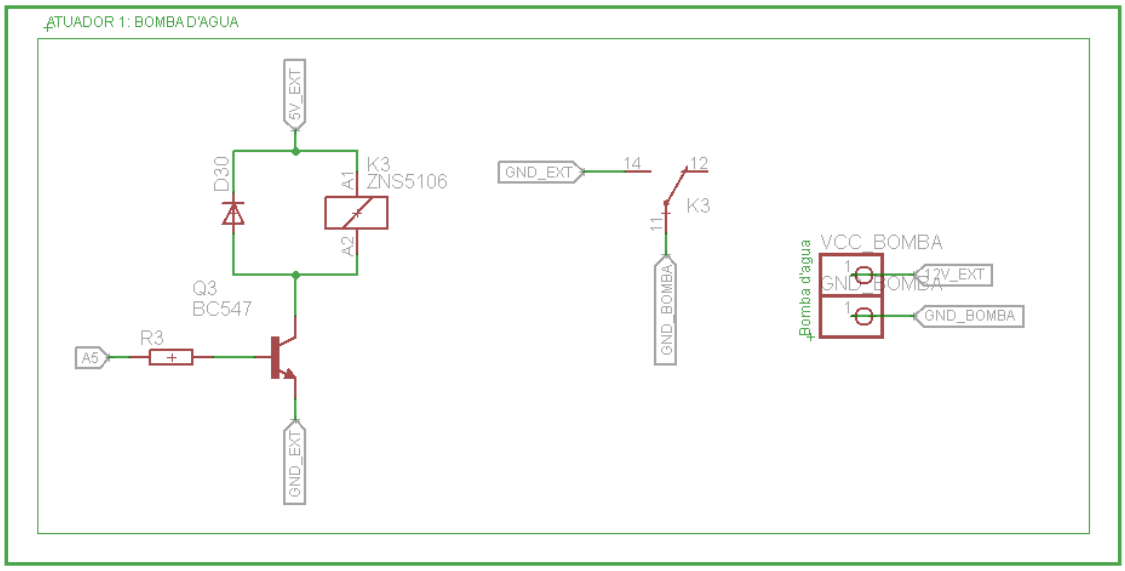


**Figura 16 - Esquemático de conexão do atuador bomba d'água.**

O controle da bomba d'água, assim como da luz, é bastante intuitivo: Apertar o botão com desenho de um regador na página *web* aciona a bomba d'água por 2 segundos (tempo determinado por inspeção visual – não tem relação com limite superior de umidade do solo atingido).

### 3.4.3. Pastilha de Peltier

Pastilha de Peltier é um dispositivo termoelétrico composto de duas junções de dois condutores ou semicondutores. Quando percorrida por corrente elétrica, ela gera o chamado “Efeito Peltier”, que produz um gradiente de temperatura entre as duas faces da pastilha. [11]

A escolha da Peltier como método de manutenção de temperatura da estufa deve-se à flexibilidade quanto a aquecer e resfriar o ambiente com apenas um dispositivo: Pode-se inverter a polaridade dos fios da pastilha para que as temperaturas das faces também se invertam (face fria passa a ser quente, e vice versa). Assim, basta fixar a pastilha com uma das faces voltadas para dentro da estufa e a outra para fora, e fazer o controle de temperatura com um circuito em Ponte H[4].

Uma imagem da pastilha adquirida para o projeto pode ser vista na Figura 17. Ela funciona em 12 V.

**Figura 17 - Pastilha de Peltier adquirida para o sistema. [12]**

A pastilha de Peltier é conhecida por precisar de boa quantidade de corrente elétrica para funcionar, pois ela possui resistência interna baixa, cerca de 2Ω (6 A

[4] Mesmo tipo de circuito usado para controlar motores: Um sinal de controle determina a polaridade do circuito (qual fio será ligado ao VCC da fonte e qual fio será ligado ao GND da fonte).

necessários). Isso torna a pastilha bastante potente (12 V * 6 A = 72 W), o que justifica seu uso em bebedouros industriais[5].

Para dissipar o calor que a pastilha de Peltier gera, foram utilizados dois pares *cooler*-dissipador. Não é recomendado que a pastilha seja energizada sem ao menos um dissipador.

Detalhes sobre a fixação do conjunto *coolers*-Peltier e o circuito de controle do mesmo podem ser vistos na seção 3.4.4.

### 3.4.4. *Coolers*

O *cooler* adquirido para o sistema pode ser visto na Figura 18.

**Figura 18 - Cooler adquirido para o sistema. [13]**

Os *coolers* e a pastilha de Peltier são acionados pelo Arduino (sempre juntos) através de um único rele controlado pelo pino A3. O circuito em questão pode ser visto na Figura 19.Como se pode observar, não foi utilizado um circuito ponte H para controle da pastilha. A explicação encontra-se nos Resultados, na seção 4.1.1.

[5]Por isso que na parte externa de um bebedouro o metal fica quente no local onde a Peltier está instalada, a outra face dela permanece fria, gelando a água.

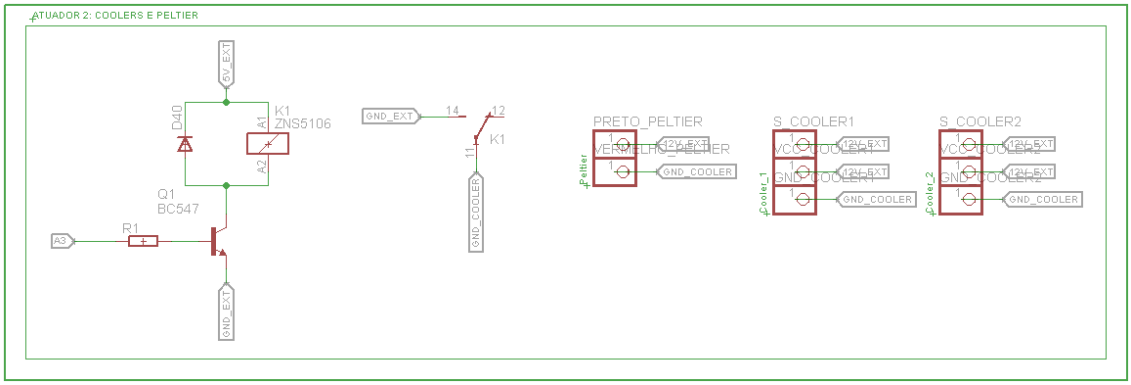


**Figura 19 – Esquemático de conexão dos *coolers* e Peltier.**

## 3.5. *Display* LCD

Para inspeção dos valores dos sensores sem acesso à página *web*, foi inserido no projeto um *display* LCD da Nokia. Uma figura do *display* já exibindo as informações pode ser visto na Figura 20.


**Figura 20 - *Display* LCD Nokia exibindo as informações da estufa.**

O modelo do *display* é o PCD8544, e a interface dele com o Arduino é feita através de uma biblioteca padrão. Para impressão na tela, são usadas funções de alto nível, como “display.println()”.

O circuito de conexão entre eles é direto e pode ser visto na Figura 21.

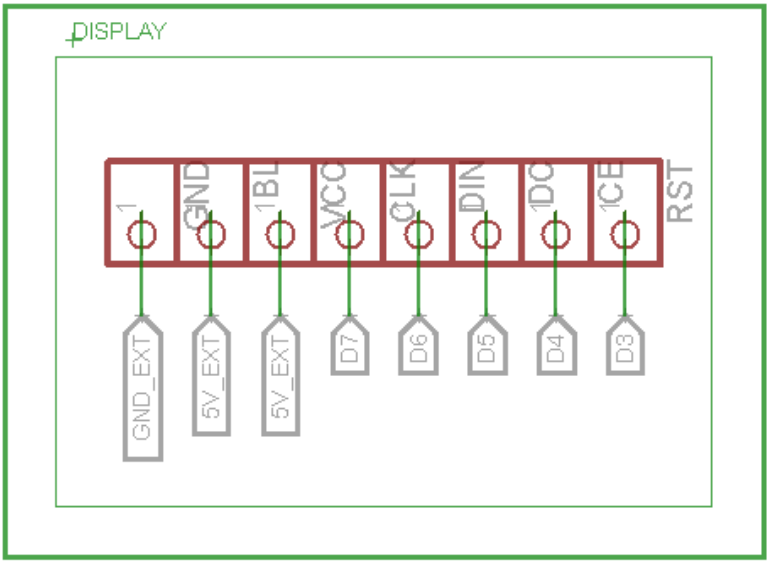


**Figura 21 - Esquemático de conexão do display LCD.**

## 3.6. Comunicação

A comunicação entre o Arduino e o computador é feita através de um cabo USB com comunicação serial.

O Arduino já possui a implementação da comunicação serial, com funções bastante intuitivas, como “Serial.read()” e “Serial.println()” que funcionam em alto nível, com “char” e “string” respectivamente. O computador não possui isso implementado, portanto foi desenvolvido um programa em linguagem Python que faz a interface necessária. Foi obtida de [14] a biblioteca “pySerial”, que permite a utilização de portas seriais do computador em um programa de alto nível. Ver seção 3.3.1 para mais detalhes.

## 3.7. Câmera

A *webcam* utilizada no projeto é um modelo comum e pode ser vista na Figura 22.

**Figura 22 - *Webcam* utilizada no projeto.**

Na página *web*, existe um botão com o desenho de uma câmera que, quando apertado, executa um comando no computador que captura uma imagem da *webcam* e a salva no diretório em que o servidor *web* está atuando (geralmente "/var/www"), sempre com o mesmo nome ("foto1.jpeg"). Na página também existe um espaço reservado para a imagem, que sempre exibe o arquivo "foto1.jpeg", qualquer que seja a última foto capturada.

O acesso da *webcam* pelo servidor é feito via linha de comando no terminal, com o auxílio do *softwareopen-source* "fswebcam" [15]. O quadro abaixo mostra a linha completa executada pelo servidor. A opção "-r" pede a resolução desejada, "-d" espera o local onde o dispositivo está montado e por fim é informado o nome da nova foto.

```
fswebcam  -r 640x480 –d /dev/video1 foto1.jpeg
```

O intuito dessa funcionalidade é a inspeção remota do desenvolvimento da planta, sem a necessidade de abrir a estufa e desequilibrar o ambiente .

### 3.8. Página *web*

A página *web* foi criada para ser bastante intuitiva ao usuário. São apenas três elementos na tela. Ela foi desenvolvida utilizando a combinação das linguagens HTML, PHP e CSS. Mais detalhes podem ser vistos na seção 4.3.2.

O primeiro elemento é uma tabela que exibe os valores de temperatura do ar, umidade do ar, umidade do solo, e estado da luz (ON/OFF) – todos esses valores são enviados pelo Arduino a cada *loop* do programa, ou seja, a taxa de atualização deles é bem rápida. Uma imagem dessa tabela pode ser visto na Figura 23.

| | |
|---|---|
| TEMPERATURA | 21 oC |
| UMIDADE | 52 % |
| UMIDADE DO SOLO | 63 % |
| LUZ | OFF |
| NIVEL D'AGUA | OK |

**Figura 23 - Tabela de valores da página *web*.**

O segundo elemento é um painel de controle disponível ao usuário. Esse painel contém botões que podem aumentar ou diminuir a temperatura desejada, acionar a bomba d'água para irrigação, acender ou apagar a luz, e tirar uma foto através da *webcam*. Uma imagem desse painel de controle pode ser visto na Figura 24.

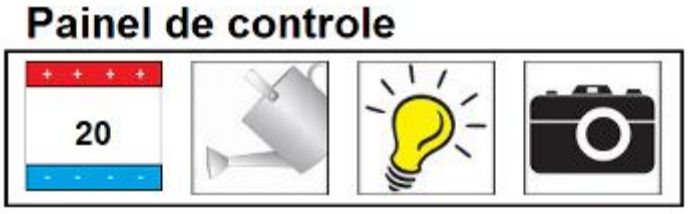


**Figura 24 - Painel de controle da página *web*.**

O terceiro e último elemento é a última foto armazenada no servidor. Essa foto é atualizada cada vez que o botão da *webcam* é apertado. Uma imagem dessa foto pode ser vista na Figura 25.

**Figura 25 - Foto da página *web*.**

Na Figura 26 pode-se observar a página completa, com o posicionamento dos três elementos citados acima.


**Figura 26 - Página *web* completa.**

# 4. Resultados

## 4.1. *Hardware*

A montagem do *hardware* levou em conta algumas considerações:

- Para utilização da Peltier de forma adequada, os *coolers* devem cada um estar encostado em uma das faces da pastilha, ou seja, “de costas” um para o outro.
- Um par dissipador-*cooler* deve estar posicionado no interior da estufa e o outro no exterior.
- O dissipador aquece bastante – o material que o prende não pode derreter.
- Os sensores devem estar no interior da estufa, e o Arduino no exterior.
- A bomba d’água deve estar dentro do reservatório de água. A mangueira que sai dela deve chegar até o interior da estufa.
- O sensor de nível de água deve estar fixado dentro do reservatório de água, em uma altura conveniente.

Para cumprir os requisitos da manutenção da temperatura, a solução escolhida foi uma caixa de isopor com uma tampa de madeira furada, por onde passam os *coolers*.

### 4.1.1. Controle de Temperatura

Para o efeito desejado (dissipação do calor nas duas faces da pastilha), os *coolers* foram fixados de costas um para o outro, com a pastilha entre eles. Um esquema da fixação desenvolvido na ferramenta SketchUp [16] pode ser visto na Figura 27.

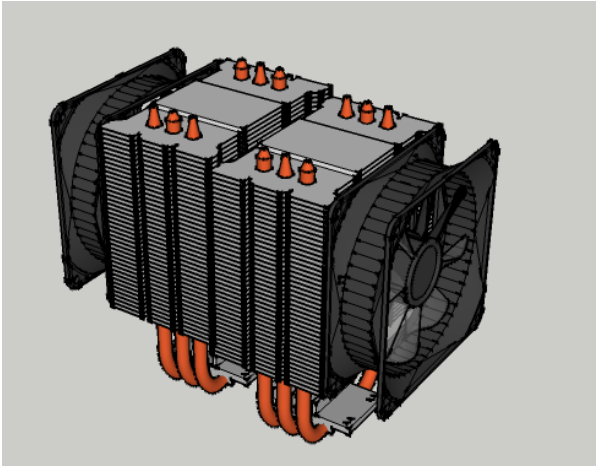

**Figura 27 - Esquema de fixação dos *coolers*. A pastilha de Peltier está entre eles.**

Os *coolers* foram parafusados um ao outro e duas chapas de alumínio foram adicionadas à estrutura, formando abas laterais que seguram um dos pares dissipador-*cooler* na parte externa da estufa (vide detalhes na Figura 28).

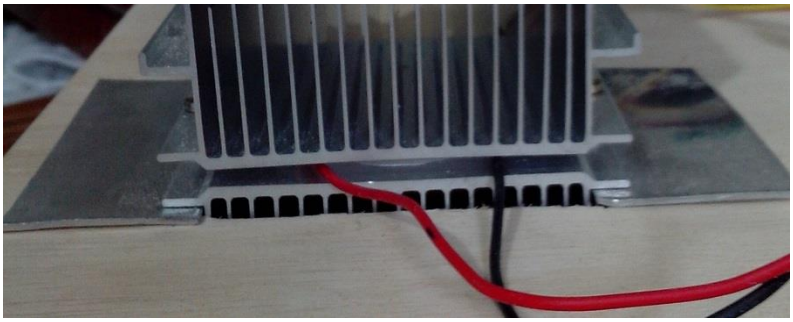

**Figura 28 - Detalhe da fixação dos coolers.**

O Arduino e toda a eletrônica foram posicionados na parte superior da tampa, para facilitar a manipulação. Os fios dos sensores – que saem do Arduino e precisam atingir o solo, por exemplo – passam por uma pequena fenda feita entre o isopor e a tampa. A mesma coisa acontece com os fios da lâmpada e a mangueira da bomba d'água.

Devido à alta corrente drenada pela pastilha Peltier, foi feito o estudo de um circuito Ponte H que pudesse controlar até 6 A. Quando a pastilha foi adquirida e ligada pela primeira vez, através de um multímetro percebeu-se que a corrente que estava sendo solicitada por ela estava na faixa dos 2 A, inutilizando o circuito

planejado, pois existem controladores prontos no mercado que conseguem controlar 2 A, e já possuem interface direta para o Arduino.

Apesar de o fato de a pastilha ser menos potente (12 V * 2 A = 24 W) tornar seu controle mais fácil, o seu desempenho é bastante prejudicado: aos primeiros testes percebeu-se que o resfriamento da caixa de isopor não estava bom. Foram feitas, então, medições durante 1 hora nas duas situações: polaridade normal (face quente virada para dentro da estufa) e polaridade invertida (face fria virada para dentro da estufa). Os resultados podem ser vistos na Figura 29 e na Figura 30.

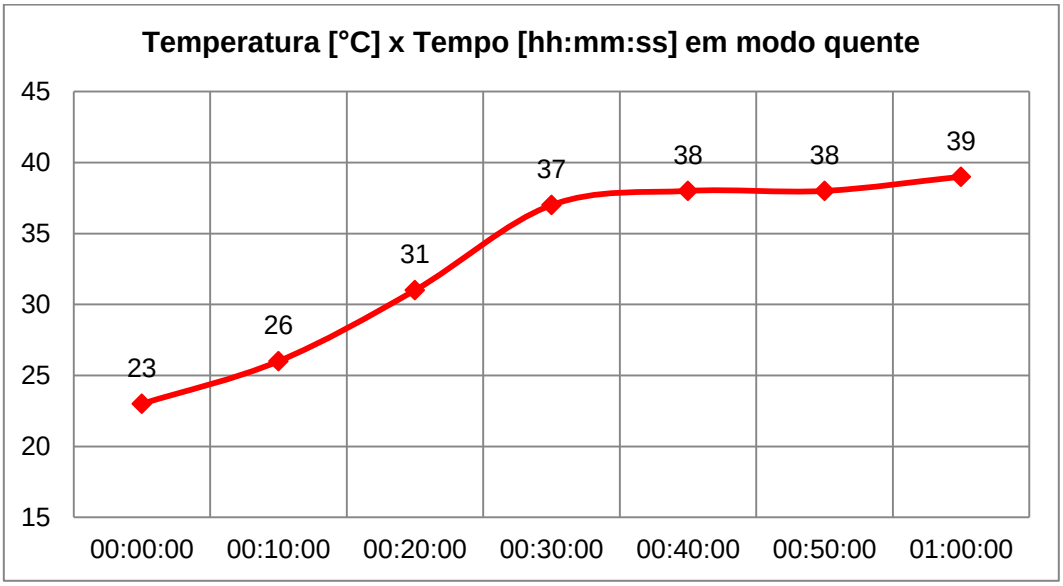


**Figura 29 - Gráfico da temperatura da estufa pelo tempo em modo de aquecimento.**

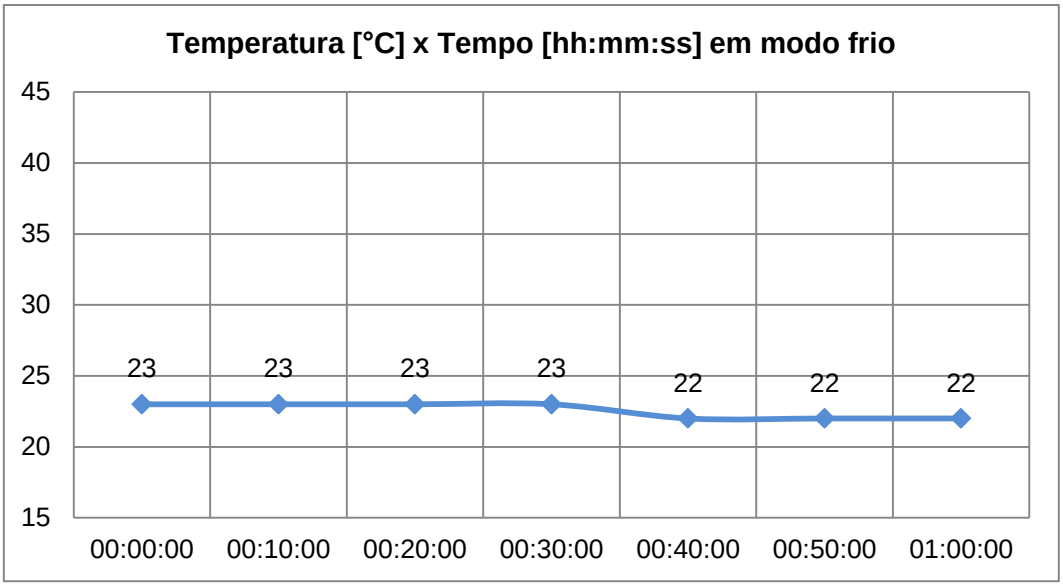


**Figura 30 - Gráfico da temperatura da estufa pelo tempo em modo de resfriamento.**

Como se pode perceber nos gráficos, a pastilha não foi eficiente para o resfriamento do ambiente, devido à baixa potência. Portanto, foi tomada a decisão de utilizar a pastilha apenas com o lado quente voltado para dentro, de forma a somente esquentar a estufa e/ou manter a mesma em uma temperatura constante, acima da temperatura ambiente. Dessa forma, não é necessário controle em ponte H, um rele liga e desliga o aquecimento conforme ilustrado no esquemático da seção 3.4.4.

### 4.1.2. Controle de Umidade

Quanto ao reservatório de água, a primeira opção foi colocá-lo ao lado da estufa. Uma foto dessa configuração pode ser vista na Figura 31. Essa montagem não apresentou bons resultados, pois tanto o pote plástico quanto a caixa de isopor foram furados para passagem da mangueira, e não foram encontrados bons métodos de vedação no reservatório de água.

A solução foi posicionar o pote plástico também na parte superior da estufa, com a mangueira direcionada para cima (saindo pela tampa do pote e entrando na caixa de isopor pela fenda no isopor). Dessa forma, nenhuma vedação é necessária. Para a fixação do sensor de nível de água, foi utilizada uma fita dupla-face da marca 3M que é bastante resistente. Fotos do detalhe da montagem definitiva do reservatório

de água podem ser vistas nas Figura 32 e Figura 33. Uma foto da nova configuração da estufa (com o reservatório de água em cima da tampa) será mostrada na Figura 40.

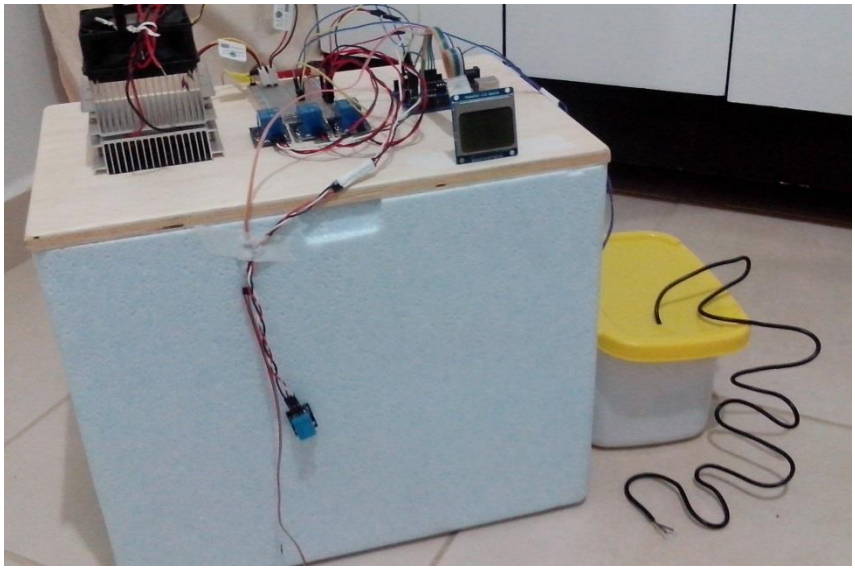

**Figura 31 - Configuração inicial da estufa.**

**Figura 32 - Bomba d'água posicionada e mangueira direcionada para cima.**

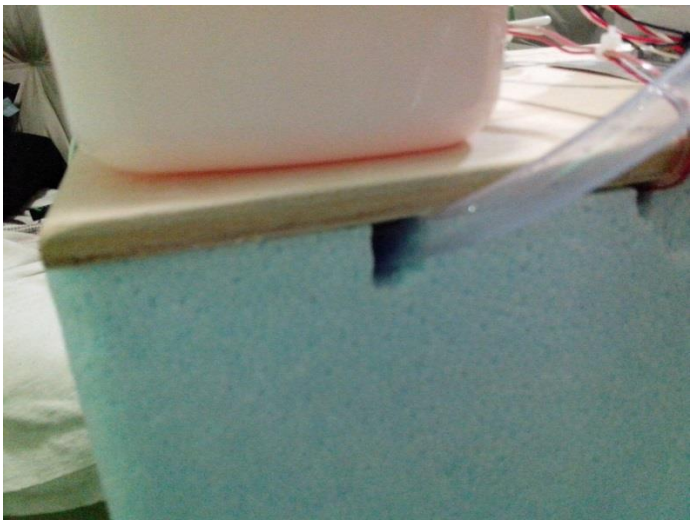

**Figura 33 - Detalhe da mangueira de água.**

### 4.1.3. Controle de Iluminação

Para conexão da lâmpada foi adquirido um soquete convencional de cerâmica. Dois fios são parafusados a ele e chegam a um conector KRE na placa de circuito impresso. A tensão desejada (110 ou 220 V) é ligada em outro conector igual.

Uma chapa de alumínio no formato de um “L” gira a lâmpada em 90º para que ela fique no sentido horizontal dentro da estufa. Um detalhe dessa montagem pode ser visto na Figura 34.

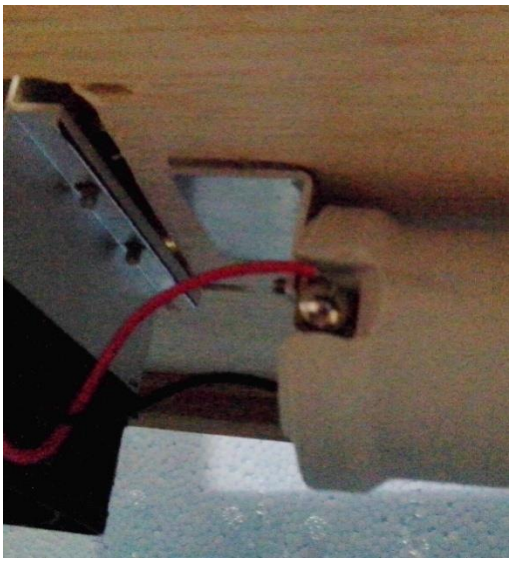

**Figura 34 - Detalhe da montagem da lâmpada.**

### 4.1.4. Circuito Impresso

Como qualquer projeto eletrônico, o primeiro protótipo da estufa foi construído sobre uma *protoboard*. O detalhe da primeira montagem pode ser visto na Figura 35.

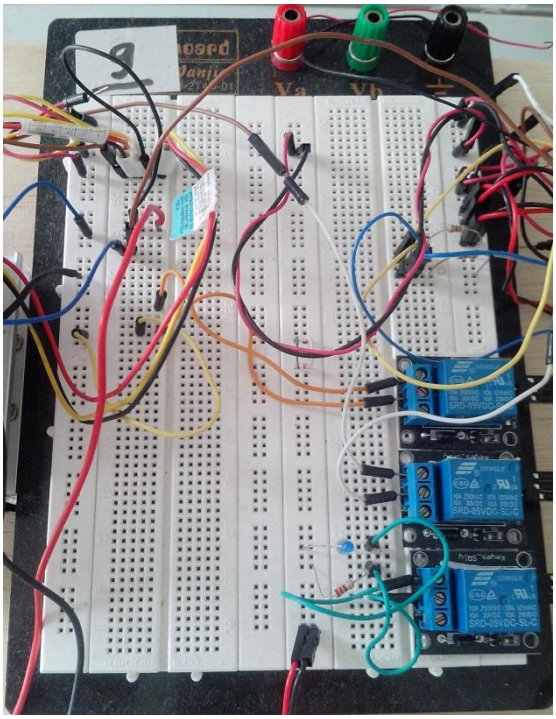

**Figura 35 - Montagem na *protoboard*.**

Com o desenvolvimento dos esquemáticos para documentação, a criação do desenho de uma placa de circuito impresso tornou-se mais simples: O próprio software Eagle faz a ligação entre os dois arquivos. Por se tratar de um projeto eletrônico pequeno, composto em sua maioria de conexões diretas, o formato escolhido para a placa de circuito impresso foi de um "*shield"* – um tipo de circuito que encaixa em todos os pinos do Arduino, realizando as conexões necessárias entre eles e os conectores de cada periférico. O desenho da placa pode ser visto na Figura 36. O detalhe do encaixe tipo "*shield*" pode ser visto nas Figura 39.

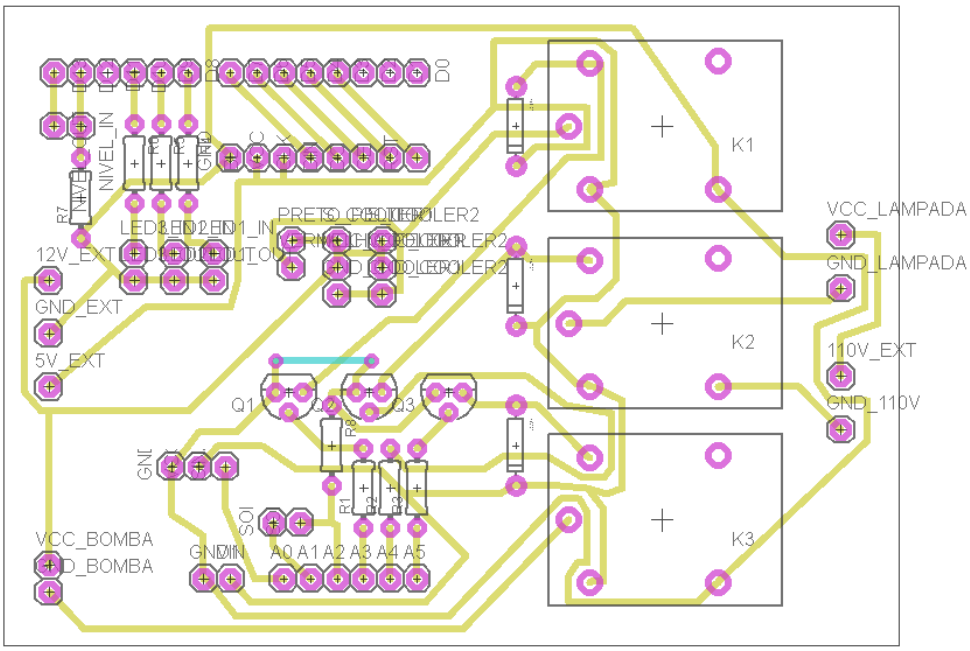


**Figura 36 - Desenho da placa de circuito impresso.**

Com o desenho pronto, a placa foi construída em uma máquina fresadora: uma estrutura que movimenta – com muita precisão – nos 3 eixos cartesianos, uma ferramenta que corrói o cobre da face inferior de uma placa, isolando as trilhas de conexão elétrica. A mesma máquina possui a função de fazer os furos para soldagem através da troca da ferramenta da extremidade. Uma foto do circuito feito nessa máquina pode ser visto na Figura 37.

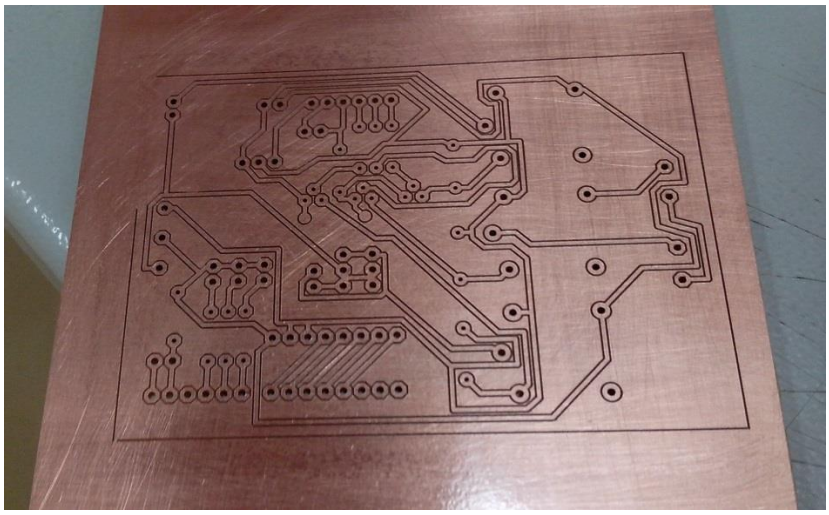

**Figura 37 - Circuito construído na máquina fresadora.**

A máquina fresadora deixa a placa com as conexões prontas e os furos feitos. Feito isso basta encaixar os componentes e soldá-los. Os componentes são: 2 conectores KRE de 2 vias (lâmpada e tomada), 3 reles (lâmpada, bomba d'água e Peltier/*coolers*), 3 diodos IN4148, 3 transistores BC547, 3 resistores 2,2 KΩ, 2 resistores de 10 KΩ e conectores macho simples (pinos de barra trilhada) para todos os outros periféricos (3 para o DHT11, 2 para o nível de água, 3 para umidade do solo, 2 para bomba d'água, 2 para Peltier, 6 para *coolers*, 8 para o *display*). Uma foto do circuito soldado pode ser vista na Figura 38. O detalhe do circuito tipo "*shield*" pode ser visto na Figura 39.

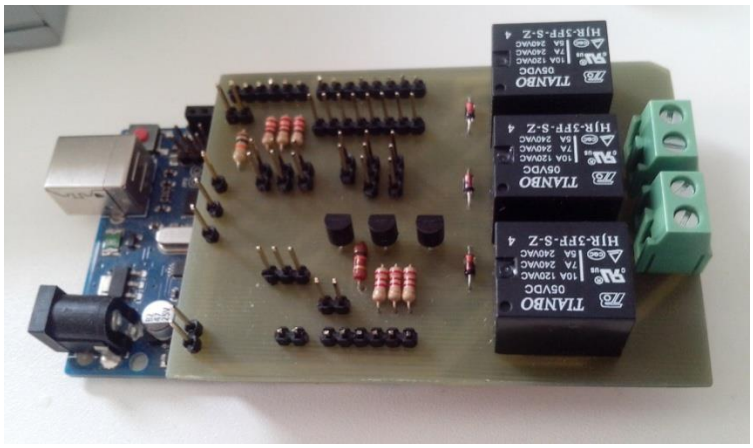

**Figura 38 - Placa de circuito impresso após soldagem.**

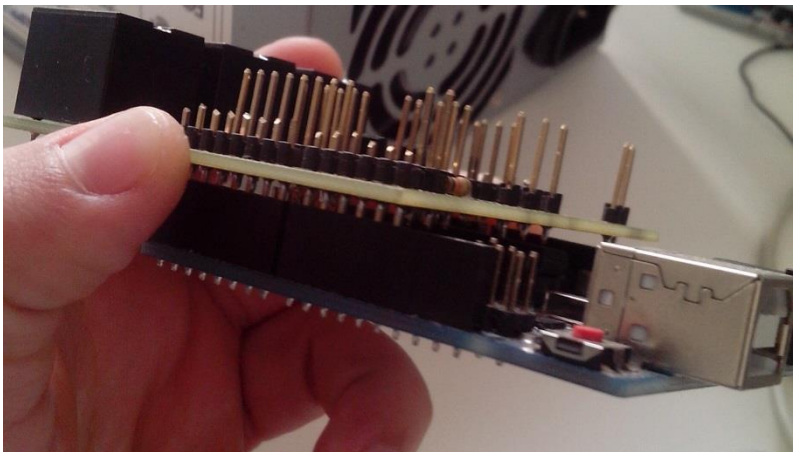

**Figura 39 - Detalhe do circuito tipo "*shield*".**

Uma foto dessa segunda montagem, com o reservatório de água na parte superior, e o circuito impresso no lugar da *protoboard*, em contraste com a Figura 31, pode ser vista na Figura 40.

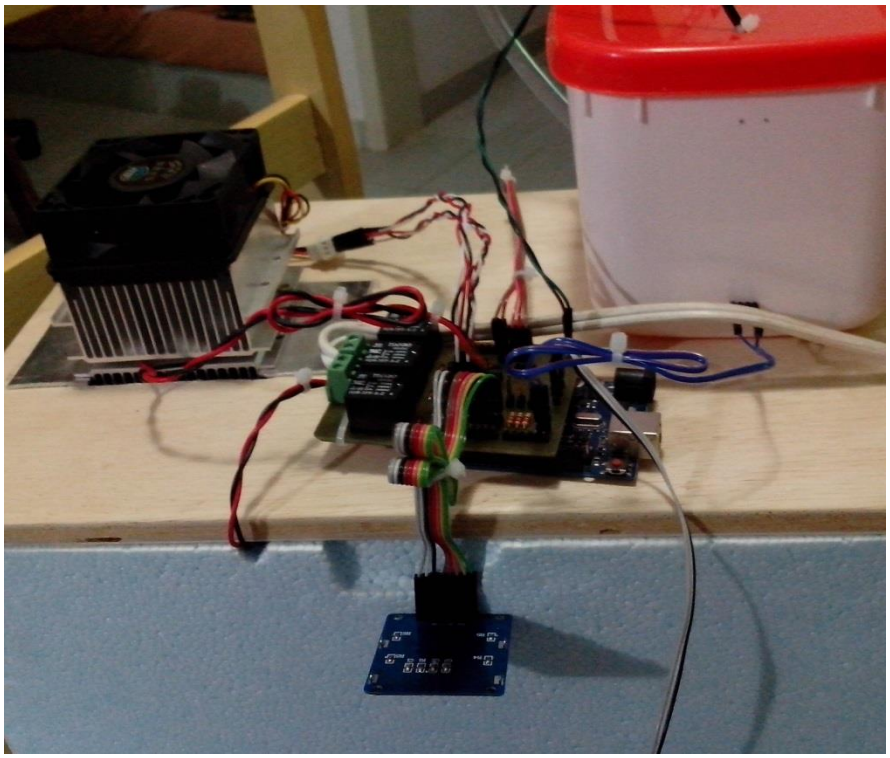

**Figura 40 - Segunda configuração da estufa.**

## 4.2. *Software* do Arduino

O *software* embarcado no Arduino conta com dois procedimentos: “setup()”, onde são feitas as definições de entrada e saída dos pinos e configuração da comunicação serial, e “loop()”, um laço infinito que permanece realizando a sequência de ações programadas.

No caso do programa desenvolvido, as ações que se repetem em cada *loop* são, em ordem:

1. Leitura da temperatura e umidade do ar no sensor DHT11;
2. Leitura do sensor de umidade do solo;
3. Leitura do sensor de nível de água;
4. Criação de uma *string* contendo todas as informações lidas;
5. Envio dessa *string* para o computador, via comunicação serial;
6. Impressão das informações no *display* Nokia.

Além disso, foi implementada uma interrupção serial, uma função que é executada cada vez que o Arduino recebe algum dado via serial (vindo da página web pelo computador). A interrupção, caracterizada pela função “serialEvent()”, padrão do Arduino, executa a seguinte rotina:

1. Enquanto dados estiverem chegando, concatena-os em uma *string* inicialmente vazia;
2. Compara a *string* montada com os comandos reconhecidos pelo Arduino:
   (a) Se a *string* for igual a “irrigar”, a bomba d’água é acionada.
   (b) Se a *string* for igual a “luz0”, a luz é apagada.
   (c) Se a *string* for igual a “luz1”, a luz é acesa.
   (d) Se a *string* for igual a “temperaturaX”, a temperatura desejada é “setada” para “X” graus Celsius.

A opção por utilizar interrupção foi devido à dificuldade de sincronismo entre o Arduino de forma manual. Muitos dados estavam sendo perdidos, pois o Arduino precisava estar posicionado em uma parte muito específica do código para verificar se haviam chegado dados.

Além da interrupção, o sincronismo também é dado por uma sequência de descargas nos canais seriais (Rx e Tx) feitas pelo computador. Mais detalhes podem ser vistos na seção 4.3.1.O código completo encontra-se no Apêndice A.

## 4.3. *Software* do Computador

O computador é responsável por atividades importantes, como a realização da comunicação serial com o Arduino, toda feita em Python via USB e a manutenção da interface *web*, realizada com a instalação de um servidor Apache [17] e programas com linguagens de programação *web*. Por fim, também foi necessário gerenciar as permissões de arquivos do sistema operacional para que tudo funcionasse conforme o desejado.

### 4.3.1. Comunicação Serial

A comunicação entre o computador e o Arduino é feita via USB através de um cabo serial-USB e um programa na linguagem Python. A linguagem Python foi escolhida pela existência de uma biblioteca chamada “pySerial” [12]. Essa biblioteca permite o tratamento da comunicação serial em alto nível, com funções como “serial.readline()” e “serial.println()”.

Para controlar a estufa via página *web*, usam-se arquivos texto que são escritos pela página *web* (permissão “www-data”, detalhes na seção 3.3.3) e lidos pelo programa em Python. Quando um botão *online* é apertado, o servidor altera um arquivo texto no seu diretório. O programa em Python permanece em um laço infinito lendo esses arquivos e verificando se houve alguma mudança em seus estados, para avisar o Arduino. A Tabela 2 explica melhor o funcionamento dos arquivos de controle.

**Tabela 2 - Funcionamento dos arquivos de controle.**

| ARQUIVO | AÇÃO PAGINA WEB | AÇÃO PYTHON |
|---|---|---|
| controle_iluminacao.txt | Escreve “0” nesse arquivo quando o estado da luz for “ON” e | Permanece verificando se o “nível lógico” escrito no arquivo |

| | | |
|---|---|---|
| | o botão da luz for clicado. Escreve “1” quando for a situação oposta. | muda, e envia ao Arduino a *string* “luz0” para apagar a lâmpada e “luz1” para acendê-la. |
| controle_irrigacao.txt | Escreve “1” nesse arquivo quando o botão da página *web* for clicado. | Permanece verificando o arquivo. Quando percebe o valor “1”, envia o comando “irrigar” ao Arduino e reseta o arquivo, escrevendo “0”. |
| controle_temperatura.txt | Escreve o valor da nova temperatura a ser atingida pela estufa no arquivo. | Permanece verificando por mudanças, quando ocorrem, envia ao Arduino a *string* “temperaturaX”, onde “X” é a nova temperatura que o Arduino deve buscar. |

Sempre que precisa avisar o Arduino de alguma alteração, o programa em Python segue uma sequência de comandos que se mostrou bastante eficaz para sincronizar a comunicação. Os comandos são do tipo “flush()”, ou seja, “descarga”. O programa em Python tem a habilidade de dar uma descarga nos canais seriais, descartando todos os dados que estão chegando ou saindo. Dessa forma, quando precisa enviar uma mensagem ao Arduino, primeiro o Python dá um “flush()” no canal de saída, “flushOutput()”. Logo depois, escreve no canal. Imediatamente, dá um novo “flush()”, agora no canal de entrada, “flushInput()” para aguardar a resposta do Arduino, que deve vir assim que ele interpretar o comando, durante uma interrupção serial[6].O código completo encontra-se no Apêndice B.

A comunicação no sentido contrário(do Arduino para o computador) envia, a cada *loop*, uma única *string* contendo todas as informações lidas dos sensores.

No caso da *webcam,* não foi necessário se preocupar com esta comunicação pois a mesma foi controlada diretamente pelo computador.

[6] Rotina que é executada toda vez que o Arduino recebe algum dado via serial, não importando o local do código que estava sendo executado no momento.

### 4.3.2. Programação *web*

O primeiro passo para criação de uma página *online* foi a instalação de um servidor para hospedá-la. Foi escolhido o Apache 2 [16]. O Apache é um servidor completo com diversas opções e de rápida instalação no computador (uma linha de comando via terminal instala tudo que é necessário).

O *software* responsável pela interface *web* entre o usuário e a estufa (arquivo "index.php" – vide Apêndice C) foi desenvolvido com uma combinação de HTML com PHP. Essas são linguagens de programação de fácil uso e com uma extensa documentação disponível. Optou-se por utilizá-las pela existência de diversos exemplos e tutoriais disponíveis na Internet, tornando-as fácil de aprender. Para configuração do estilo da página, foi utilizado CSS, além de imagens para os botões, e banners para o topo e fim da página. A imagem da interface completa encontra-se na Figura 26.

### 4.3.3. Permissões Arquivos

Para que o usuário pudesse de fato controlar a estufa através da página *web* foi necessário configurar corretamente as permissões de arquivos no sistema operacional do computador. Assim, para que cada botão funcionasse corretamente ao ser apertado pelo usuário era necessário que os arquivos textos de controle para a temperatura, irrigação e iluminação possuíssem permissão "www-data". Esta permissão é dada, por exemplo, com o seguinte comando:

```
sudo chown –R www-data:www-data controle_*.txt
```

Este comando permite que uma aplicação *web*, no caso através de um navegador, altere os dados contidos no arquivo texto, que é o mesmo lido pelo programa Python da comunicação serial com o Arduino, permitindo desta forma o controle da estufa através da página *web*.

Além das permissões “www-data” também foi necessário dar permissão “777” ao arquivo“/dev/video1” que controla a *webcam*, permitindo também tirar fotos através do botão da página *web*, com o seguinte comando:

```
sudo chmod 777 /dev/video1
```

Para facilitar a configuração do computador, foi criado um arquivo *shell script* com os comandos citados acima, que preparam todas as permissões para que o programa em Python funcione corretamente. O arquivo *shell script* é executado com o comando a seguir:

```
sudo sh config.sh
```

O arquivo “config.sh” encontra-se no ApêndiceD.

### 4.4. Caso de Teste

Para caso de teste da estufa, foi adquirido um vaso pequeno e foram plantados alguns grãos de feijão. Esse vegetal foi escolhido pela sua simplicidade, de forma que o objetivo desta parte do trabalho era validar o protótipo em termos de iluminação, irrigação, e, principalmente de monitoramento remoto, pela *webcam*. Concluiu-se que a escolha de um vegetal com necessidades muito específicas, que poderia acabar não se desenvolvendo totalmente devido às condições disponíveis no ambiente, não era interessante.

Duas imagens capturadas pela *webcam* após a plantação podem ser vistas na Figura 41e Figura 42. Elas mostram o vegetal em duas situações diferentes: logo após o plantio, e algumas semanas depois dos brotos aparecerem.

**Figura 41 - Imagem capturada pela *webcam* logo após o plantio.**

**Figura 42 - Imagem capturada pela *webcam* algumas semanas após o plantio.**

É válido reforçar novamente que se tratou de um teste de validação. A estufa pode ser totalmente adaptada a outras espécies de plantas, que precisem de mais iluminação, por exemplo, (instalação de mais lâmpadas), ou temperaturas mais quentes (instalação de uma pastilha de Peltier com maior potência).

# 5. Conclusões

## 5.1. Conclusões gerais

A primeira conclusão é que o projeto funcionou como projetado.

A pior característica ao final do desenvolvimento é a qualidade individual dos módulos adquiridos a baixo custo. Se houvesse um orçamento maior disponível, seria possível a criação de um protótipo profissional. Embora a aparência amadora do *hardware*, ao final das modificações o *software* não apresentou nenhum mau funcionamento.

## 5.2. Conclusões sobre o *hardware*

Como o desenvolvimento do projeto levou 1,5 anos, houve tempo suficiente para existir mais de um protótipo de montagem. Com isso, várias características foram melhoradas com o passar do tempo.

Entre a primeira e segunda versão, é interessante ressaltar a questão da vedação no reservatório de água, pois foi um problema que demorou a ser solucionado (fixá-lo na parte superior), além de, e principalmente, o circuito impresso.

A criação do circuito na *protoboard*, seguido da documentação feita em *software* próprio, a construção da placa na máquina fresadora, a soldagem e os testes, foram, em conjunto, a experiência mais interessante que teve das possibilidades existentes para uma engenheira de computação no mercado.

A maior dificuldade foi a falta de ferramentas em casa, como furadeira, chaves, etc., pois algumas coisas foram presas com fita e não ficaram tão boas.

## 5.3. Conclusões sobre o *software*

Os maiores aprendizados em relação ao *software* do computador foram a utilização da linguagem Python – que simplifica muito o trabalho, a questão das

permissões dos arquivos em Linux para o acesso do servidor *web* e do terminal simultaneamente (como no caso da *webcam*); e também a criação da sequência de "descargas" nos canais Tx e Rx da comunicação serial, pois resolveu vários problemas de comunicação que existiram.

No caso do *software* do Arduino, o primeiro protótipo não utilizava interrupção serial. Isso causava a perda de comandos e consequente mau funcionamento.

Com a união dessas melhorias, a comunicação ficou bastante confiável. No início do projeto, quando o sistema permanecia ligado por mais de 10 minutos, a comunicação perdia o sincronismo e não funcionava. Ao final do projeto isso não aconteceu mais.

A página *web* não apresentou grandes problemas para ser desenvolvida.

## 5.4. Trabalhos Futuros

Existem três principais ideias de trabalhos futuros para esse projeto: A aquisição de uma pastilha de Peltier com potência suficiente para refrigerar o recipiente, para que a faixa de temperatura mantido pela estufa aumente consideravelmente; A adaptação dos atuadores para que seja possível abrigar espécies específicas de plantas, que, por exemplo, precisem de mais iluminação do que a estufa consegue fornecer atualmente e também a construção de uma "caixinha" ou *case* que envolva o Arduino e o circuito impresso, dando um aspecto mais limpo e profissional ao projeto.

Com base em algumas falhas de funcionamento detectadas em alguns módulos na fase final do projeto, uma boa melhoria também seria a aquisição e instalação de sensores e atuadores mais profissionais, pois aquilo que é adquirido a baixo custo pode ter sua qualidade comprometida em algum momento.

# REFERÊNCIAS BIBLIOGRÁFICAS


[1] “Eagle”. Disponível em http://www.cadsoftusa.com/download-eagle/freeware/. Acesso em 29/10/2014.

[2] “Sistemas Operacionais”. Disponível em http://www2.ic.uff.br/~aconci/SistemasOperacionais.html. Acesso em 10/09/2014.

[3] Ubuntu 12.04. Disponível em http://www.ubuntu.com/download. Acesso em 01/05/2014.

[4]“Microcontroladores”. Disponível em http://tecnologia.hsw.uol.com.br/microcontroladores1.htm. Acesso em 10/09/2014.

[5] Microcontrolador ATMega328. Disponível em http://www.nkcelectronics.com/arduino-ready-avr-atmega328-microcontrol328.html. Acesso em 08/09/2014.

[6] Página oficial do Arduino UNO. Disponível em http://arduino.cc/en/Main/arduinoBoardUno. Acesso em 26/08/2013.

[7] *Datasheet* do sensor DHT11. Disponível em http://www.micro4you.com/files/sensor/DHT11.pdf. Acesso em 26/08/2013.

[8] “O transistor bipolar como chave”. Disponível em http://www.ebah.com.br/content/ABAAAfLpMAD/transistor-bipolar-como-chave. Acesso em 05/06/2014.

[9] “Efeito de luzes coloridas no desenvolvimento de plantas”. Disponível em http://www.ehow.com.br/efeito-luzes-coloridas-crescimento-plantas-sobre_76499/. Acesso em 10/09/2014.

[10] *DealExtreme.* Disponível em http://dx.com/p/6971-brushless-mute-air-pump-black-dc-6-12v-177852. Acesso em 26/11/2013.

[11] “Efeito Peltier”. Disponível em http://efisica.if.usp.br/eletricidade/basico/termo/efeito_peltier/. Acesso em 29/10/2014.

[12] *DealExtreme.* Disponível em http://dx.com/p/tec1-12706-semiconductor-thermoelectric-cooler-peltier-white-157283. Acesso em 26/11/2013.

[13] *DealExtreme.* Disponível em http://dx.com/p/cooler-master-heatsink-fan-for-intel-p4-and-celeron-d-725r-gp-5112. Acesso em 26/11/2013.

[14] Biblioteca “pySerial”. Disponível em http://pyserial.sourceforge.net/. Acesso em 20/11/2013.

[15] Tutorial “fswebcam”. Disponível em: http://www.slblabs.com/2012/09/26/rpi-Webcam-stream/. Acesso em 31/11/2013.

[16] Google Sketchup. Disponível em http://www.sketchup.com/pt-BR/download. Acesso em 01/05/2014.

[17] Apache 2. Disponível em http://httpd.apache.org/download.cgi. Acesso em 01/05/2014.

# Apêndice A

Código embarcado no microcontrolador ATMega328 (plataforma Arduino).

```
#include <Adafruit_GFX.h>
#include <dht11.h>
#include <Adafruit_PCD8544.h>
#include <string.h>

/*
 * @file estufaV1.ino
 * @author beatriz
 */

/*
 * pinagem
 *sensor dht11            A0
 *sensor nivel agua       (out)13  (in)12
 *sensor umidade solo     (out)A1  (in)A2
 *display    (rst)3  (ce)4  (dc)5  (din)6  (clk)7
*reles      (cooler/peltier)A3  (luz)A4  (bombaAgua)A5
*/

Adafruit_PCD8544 display = Adafruit_PCD8544(7, 6, 5, 4, 3);
dht11 DHT11;

#define  tempPin      A0
#define  nivelPinOut  13
#define  nivelPinIn   12
#define  soloPinOut   A1
#define  soloPinIn    A2
#define  coolerPin    A3
#define  luzPin       A4
#define  bombaPin     A5

charok_no[3], luz[3];
/* valores escolhidos pelo usuario */
inttemperatura, umidade, solo;
/* valores lidos pelo arduino */
intsensorTemperatura, sensorUmidade, sensorSolo, sensorNivel;
```

```
StringcriaString(int temperatura, int umidade, int solo, intnivel,
char* luz);
voidimprimeTela(int temperatura, int umidade, int solo, intnivel,
char* luz);
void aquecer();
void irrigar();
voidapagarluz();
void acenderluz();

String inputString = "";
booleanstringComplete = false;

void setup() {

inputString.reserve(200);

  /* configuracao dos pinos */
pinMode(tempPin, INPUT);
pinMode(nivelPinOut, OUTPUT);
pinMode(nivelPinIn, INPUT);
pinMode(soloPinOut, OUTPUT);
pinMode(soloPinIn, INPUT);
pinMode(coolerPin, OUTPUT);
pinMode(luzPin, OUTPUT);
pinMode(bombaPin, OUTPUT);

/* liga os sensores de nivel e umidade do solo */
digitalWrite(nivelPinOut, HIGH);
digitalWrite(soloPinOut, HIGH);

  /* configura o lcd display */
display.begin();
display.setContrast(40);
display.clearDisplay();

  /* inicia comunicacao serial */
Serial.begin(9600);

strcpy(luz, "NO");
digitalWrite(luzPin, LOW);
```

```
digitalWrite(coolerPin, LOW);

  /* inicializa valores de controle */
temperatura = 25;
solo = 50;
}

void loop() {

display.clearDisplay();                    /* limpa a tela */

  /* SENSORES */

DHT11.read(tempPin);                          /* leitura da temperatura
e umidade do ar */
sensorTemperatura = DHT11.temperature;
sensorUmidade = DHT11.humidity;

sensorSolo = analogRead(soloPinIn);           /* leitura da umidade do
solo */

floatporcSoloFloat = sensorSolo/1024.0 * 100.0;
intporcSolo = porcSoloFloat;

sensorNivel = digitalRead(nivelPinIn);         /* leitura do nivel de
agua */

  /* envia uma string para a rasp com todas as informacoes */
Serial.println(criaString(sensorTemperatura,            sensorUmidade,
sensorSolo, sensorNivel));

imprimeTela(sensorTemperatura, sensorUmidade, sensorSolo, sensorNivel,
luz);

}

String criaString(inttemperatura, intumidade, int solo, intnivel) {

  char nivelStr[3];
  if (nivel == 1) {
strcpy(nivelStr, "OK");
```

```
  }
  else {
strcpy(nivelStr, "NO");
  }

  float porcSoloFloat = solo/1024.0 * 100.0;
intporcSolo = porcSoloFloat;

  String temp = String(temperatura);
  String umid = String(umidade);
  String umids = String(porcSolo);

  String tudo = "";
  String t = "TEMPERATURA: ";
String u = "UMIDADE: ";
String s = "UMIDADESOLO: ";
String n = "NIVEL: ";
  String div = "|";

tudo.concat(t);
tudo.concat(temp);
tudo.concat(div);
tudo.concat(u);
tudo.concat(umid);
tudo.concat(div);
tudo.concat(s);
tudo.concat(umids);
tudo.concat(div);
tudo.concat(n);
tudo.concat(nivelStr);
tudo.concat(div);

return tudo;
}

voidimprimeTela(int  temperatura,  int  umidade,  int  solo,  intnivel,
char* luz) {

floatporcSoloFloat = solo/1024.0 * 100.0;
intporcSolo = porcSoloFloat;
```

```
display.setTextSize(1);
display.setTextColor(BLACK);
display.setCursor(30,0);
display.println("v1.0");

display.print("Temp:    ");
display.print(temperatura);
display.println("oC");

display.print("Umid:     ");
display.print(umidade);
display.println("%");

display.print("UmidSl:   ");
display.print(porcSolo);
display.println("%");

display.print("Nivel:     ");
  if (nivel == 1) {
display.println("OK");
  }
  else {
display.println("NO");
  }

display.print("Luz:       ");
display.println(luz);

display.display();
  delay(500);
}

/*
 * @brief liga a peltier com o lado quente para dentro. o mesmo rele
que liga o cooler, liga a peltier
*/
void aquecer() {

digitalWrite(coolerPin, HIGH);
}
```

```
/*
 * @brief liga a bomba d'agua por 2 segundos e desliga
*/
void irrigar() {

digitalWrite(bombaPin, HIGH);
  delay(2000);
digitalWrite(bombaPin, LOW);
}

/*
 * @brief apaga a luz
 */
void apagarluz() {
digitalWrite(luzPin, LOW);
strcpy(luz, "");
strcpy(luz, "OFF");
}

/*
 * @brief acende a luz
*/
void acenderluz() {
digitalWrite(luzPin, HIGH);
strcpy(luz, "");
strcpy(luz, "ON");
}

/*
 * @briefinterrupcao serial
*/
void serialEvent() {

  while (Serial.available()) {
charinChar = (char)Serial.read();
inputString += inChar;
    if (inChar == '\n') {
stringComplete = true;
    }
  }
```

```
  if (inputString.equals("irrigar")) {

Serial.println("COMANDO RECEBIDO PELA SERIAL: LIGAR BOMBA");
irrigar();

  }        else        if        ((inputString.equals("luz0")        ||
inputString.equals("luz1"))) {

if (inputString.charAt(3) == '0') {
Serial.println("COMANDO RECEBIDO PELA SERIAL: APAGAR LUZ");
apagarluz();
}else {
Serial.println("COMANDO RECEBIDO PELA SERIAL: ACENDER LUZ");
acenderluz();
      }

  } else if (inputString.charAt(0) == 't' &&inputString.charAt(1) ==
'e' &&inputString.charAt(2) == 'm' &&inputString.charAt(3) == 'p') {

      String valor_temp = inputString.substring(11);

intvalor_temp_int = valor_temp.toInt();

temperatura = valor_temp_int;

Serial.print("COMANDO RECEBIDO PELA SERIAL: TEMPERATURA = ");
Serial.print(valor_temp_int);
Serial.println(" oC");

if (sensorTemperatura< (temperatura)) {
aquecer();
}else {
digitalWrite(coolerPin, LOW);
}
}

inputString = "";
}
```

## Apêndice B

Código em Python que permanece em execução no computador.

```
import serial
import time
fromdatetimeimportdatetime

temp_old = 0
ilum_old = 0
sec_old = 0
ilum = 0

while 1:
      # INICIA COMUNICACAO COM O ARDUINO
      ser = serial.Serial('/dev/ttyACM0',9600, timeout=1)
      time.sleep(0.5)

      # LEITURA DE INFORMACOES DO ARDUINO
      arq0 = open('/var/www/arquivo.txt','w+')
      arq0.seek(0)
      lido = ser.readline()
      if (lido.startswith("TEMPERATURA")):
            arq0.write(lido)
      arq0.seek(0)
      print arq0.readline()
      arq0.close()

      # CONTROLE DE TEMPERATURA
arq1 = open('/var/www/controle_temperatura.txt','r')
arq1.seek(0)
      temp=arq1.read()
if(temp != temp_old):
            # ENVIA COMANDO AO ARDUINO
            ser.flushOutput()
            ser.flushInput()
ser.write("temperatura"+temp)
            ser.flush()
            time.sleep(0.1)
            # ACK
```

```
            printser.readline()
            # ATUALIZA TEMP_OLD
            temp_old=temp
arq1.close()

      # CONTROLE DE IRRIGACAO
      arq2 = open('/var/www/controle_irrigacao.txt','r+')
      arq2.seek(0)
      if(arq2.read()=="1"):
            # ENVIA COMANDO AO ARDUINO
            ser.flushOutput()
            ser.flushInput()
            ser.write("irrigar")
            ser.flush()
            time.sleep(0.1)
            # ACK
            printser.readline()
            # RESETA O ARQUIVO
            arq2.seek(0)
            arq2.write("0")
      arq2.close()


      # CONTROLE DE ILUMINACAO
arq3 = open('/var/www/controle_iluminacao.txt','r')
arq3.seek(0)
ilum=arq3.read()
if(int(ilum)!=int(ilum_old)):
            # ENVIA COMANDO AO ARDUINO
            ser.flushOutput()
            ser.flushInput()
            ser.write("luz"+ilum)
            ser.flush()
            time.sleep(0.1)
            # ACK
            printser.readline()
            # ATUALIZA ILUM_OLD
            ilum_old = ilum
arq3.close()
```

# Apêndice C

Código “index.php” responsável pela página *web*.

```
<?php
$page = $_SERVER['PHP_SELF'];
$sec = '3';
$T=file_get_contents("controle_temperatura.txt");
$I=file_get_contents("controle_iluminacao.txt");
?>
<html>
      <head>
      <meta           http-equiv="refresh"           content="<?phpecho
$sec?>;URL='<?phpecho $page?>'">
      <styletype="text/css">
            @importurl("styles.css");
//          @importurl("onOff.css");
      </style>
<?php
            if(isset($_POST["botao_foto"])){
                  echoshell_exec('fswebcam  -r  960x720  -d  /dev/video1
foto1.jpg');
                  echo "Tirando a foto...";
      }
if(isset($_POST["botao_temp_inc"])){
                  if($T>=30){
                        echo "Valor maximo= 30 oC";
                  }
                  else{
      echo "Aumentando temperatura...";
                        $T++;
                        echo $T;
                        file_put_contents("controle_temperatura.txt",
$T);
                  }
      }
if(isset($_POST["botao_temp_dec"])){
if($T<=20){
echo "Valor minimo= 20 oC";
}
```

```
else{
      echo "Diminuindo temperatura...";
                        $T--;
      echo $T;
      file_put_contents("controle_temperatura.txt", $T);
                  }
      }
if(isset($_POST["botao_irrigar"])){
echo "Irrigacao ativada...";
file_put_contents("controle_irrigacao.txt", "1");
      }
if(!empty($_POST["onoffswitch"])){
echo "On/Off apertado...";
//                          echoshell_exec('sh irrigar.sh');
      }
            if(isset($_POST["botao_luz"])){

                  if($I == 0){
                        echo "Acendendo a luz...";
                        $I=1;
                        file_put_contents("controle_iluminacao.txt",
$I);
                  }
                  else{
echo "Apagando a luz...";
                        $I=0;
file_put_contents("controle_iluminacao.txt", $I);
                  }
            }


       ?>
      </head>
      <body id="corpo">
      <divalign="center" id="b1">
      <imgsrc="banner1.jpg" align="middle">
      </div>
      <h5     style="font-family:arial;text-align:center;color:black;"
/h5>
      <?php
      $str = file_get_contents("arquivo.txt");
      list($temperatura,   $umidade,   $umidadeSolo,   $niveldagua)   =
```

```
explode("|", $str);
      list($temp, $ntemp)=explode(":",$temperatura);
      list($umid, $numid)=explode(":",$umidade);
list($umidS, $numidS)=explode(":",$umidadeSolo);
list($nivel, $nnivel)=explode(":",$niveldagua);
      if($I==0)
            $nilum="OFF";
      else
            $nilum="ON";
      $ilum="LUZ";
/*        if($nnivel<100)
               $enivel="CRITICO";
else
            $enivel="OK";*/

      echo '<tablewidth="30%" align="center" id="tabela" style="font-
family:arial;color:black;font-size:x-large;">';
      echo '<thead><tr>';
      echo '</tr></thead>';
      echo '<tbody>';
echo '<tr>';
echo '<td><b>' . "TEMPERATURA" . '</b></td>';
echo '<td><b>' . $ntemp ." oC" .'</b></td>';
echo '</tr>';

echo '<tr>';
echo '<td><b>' . "UMIDADE AR" . '</b></td>';
echo '<td><b>' . $numid ." %" . '</b></td>';
echo '</tr>';

echo '<tr>';
echo '<td><b>' . "UMIDADE SOLO" . '</b></td>';
echo '<td><b>' . $numidS . " %" . '</b></td>';
echo '</tr>';

echo '<tr>';
echo '<td><b>' . "ILUMINACAO" . '</b></td>';
echo '<td><b>' . $nilum . '</b></td>';
echo '</tr>';

echo '<tr>';
```

```
echo '<td><b>' . "NIVEL DAGUA" . '</b></td>';
echo '<td><b>' . $nnivel . '</b></td>';
echo '</tr>';

      echo '</tbody></table>'
      ?>

      <h1 id="painel" style="font-family:arial;color:black;">Painel de
controle</h1>

      <divclass="caixa_painel"></div>

      <divalign="center" id="foto">
            <imgsrc="foto1.jpg"       align="middle"       width="300"
height"200">
      </div>

<formalign="center"    id="botao"    name="botao_foto"    method="post"
action="index.php">
<input     type="image"     src="img_camera.jpg"     name="botao_foto"
id="botao_maquina"  value="Fotografar!"  style="width:  100px;  height:
100px; font-size: 10px;"/>
      </form>

<formalign="center"   id="Tinc"   name="botao_temp_inc"   method="post"
action="index.php">
<input   type="image"   src="img_temp_inc.png"   name="botao_temp_inc"
id="Tinc"  value="+"  style="width:  100px;  height:  20px;  font-size:
15px;"/>
</form>

      <div id="Techo">
      <h2     style="font-family:arial;text-align:center;color:black;"
/h2>
            <?phpecho "$T"; ?>
      </div>

<formalign="center"   id="Tdec"   name="botao_temp_dec"   method="post"
action="index.php">
<input   type="image"   src="img_temp_dec.png"   name="botao_temp_dec"
id="Tdec"  value="-"  style="width:  100px;  height:  20px;  font-size:
```

```
15px;"/>
</form>

<formalign="center"  id="irrigar"  name="botao_irrigar"  method="post"
action="index.php">
<input    type="image"    src="img_regador.jpg"    name="botao_irrigar"
id="irrigar" value="Irrigar" style="width: 100px; height: 100px; font-
size: 50px;"/>
</form>

      <formalign="center"   id="luz"   name="botao_luz"   method="post"
action="index.php">
            <input   type="image"   src="img_luz.jpg"   name="botao_luz"
id="luz" value="Iluminar" style="width=100px; height=100px;">
      </form>

<divalign="center" id="b2">
<imgsrc="img_banner2.jpg" align="middle">
      </div>
      </body>
</html>
```

## Apêndice D

Código em *Shell Script* para configuração do computador.

```
chownwww-data:www-data controle_*.txt
chmod 777 /dev/video0
python controle.py
```