PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ
ESCOLA POLITÉCNICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

ANDRÉ MARCELO GUARINGUE
CELSO MARTINS INÁCIO JÚNIOR
GUILHERME ALCEU XAVIER DA SILVA

RELATÓRIO FINAL DE PROJETO INTEGRADOR
PROJETO AUTO COFFEE

CURITIBA
2015

ANDRÉ MARCELO GUARINGUE
CELSO MARTINS INÁCIO JÚNIOR
GUILHERME ALCEU DA SILVA XAVIER

RELATÓRIO FINAL DE PROJETO INTEGRADOR
PROJETO AUTO COFFEE

Relatório de Projeto apresentado ao Curso de Engenharia de Computação da Pontifícia Universidade Católica do Paraná, como requisito parcial para a disciplina de Resolução de Problemas em Engenharia de Computação.

Orientador: Prof. Afonso Ferreira Miguel

CURITIBA
2015

RESUMO

Este relatório tem por finalidade apresentar os conceitos empregados na solução de problemas concernentes ao desenvolvimento da cafeteira automatizada *Auto Coffee.* A teoria paralela ao projeto, objetivos, problemas enfrentados, tratamentos, resultados, conclusões e suas consequências ao ambiente são aqui apresentadas.

Palavras-chave: Café. Automatizado. Máquina. Internet.

ABSTRACT

This report's finality is to show concepts employed in its respective problems' solution, which Auto Coffee's development brought on. Project's parallel theories, objectives, faced problems, treatment, results, conclusions and their environmental consequences are presented here.

Key-words: Coffe. Auto. Machine. Internet.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

Wi-fi	*Wireless fidelity* - fidelidade sem fio.
iOS	*iPhone Operational System* - Sistema operacional móvel da *apple*
IP	*Internet protocol* - protocolo de *internet*
IDE	*Integrated-development Environment* - Ambiente de desenvolvimento integrado
PWM	*Pulse-Width Modulation* - Modulação por largura de pulso
PCI	Placa de circuito impresso
.cpp	Extensão de arquivos contendo diretivas de C++
.h	Extensão de declarações em linguagens como C++
PUCPR	Pontifícia Universidade Católica do Paraná
App	Aplicativo (para dispositivos móveis)

SUMÁRIO

1. INTRODUÇÃO

Este projeto tem por objetivo atender a necessidade de seus usuários em relação ao tempo e praticidade na hora de tomar seu café, pois através de um aplicativo, o usuário poderá solicitar sua bebida, mesmo que não esteja no mesmo ambiente em que a cafeteira, como por exemplo, solicitar o café do seu quarto, e ao levantar, seu café já estará pronto, ou até mesmo antes de sair da faculdade, e assim que chegar em casa, sua bebida estará pronta para consumo.

Além de toda praticidade, o usuário poderá solicitar a hora exata em que sua bebida ficará pronta, e ainda, escolher o nível de café e açúcar que deseja, podendo assim selecionar uma bebida de amarga a doce, de fraca a forte.

O procedimento se inicia a partir do momento em que o usuário solicita um café através do aplicativo, com um módulo *Wi-fi* integrado no *arduino*, a cafeteira receberá o pedido e dará inicio ao processo de preparação para o café.

A cafeteira utiliza como principais ferramentas *servo*-motores para fazer todos o processo relacionado ao copo, bem como ao *mixer*. Além de utilizar diversos mecanismos para fazer o controle de água, do café e do açúcar.

1.1.HISTÓRICO DO PROJETO

Tendo como idealizadores os alunos Celso Junior e Guilherme Alceu, bem como o docente Afonso Ferreira Miguel’ o projeto surgiu através da necessidade de uma cafeteira, onde pudesse ser solicitado o café do local em que o usuário estivesse, e ainda, escolher a quantidade de produtos, como açúcar e café, e a hora e que o café ficará pronto, tendo em vista que nas cafeteiras convencionais o usuário precisa ficar certo tempo esperando sua bebida ficar pronta.

1.2.OBJETIVOS

Neste tópico abordaremos sobre os objetivos gerais, que fala sobre uma visão mais ampla dos objetivos, e os objetivos específicos, que explica os objetivos mais detalhados.

1.2.1.Objetivo Geral

Solicitar o café através de um aplicativo de celular. Fazer com que o café esteja pronto na hora solicitada. Controlar a quantidade de café e açúcar. Redução do tempo de espera em relação as demais máquinas.

1.2.2.Objetivos Específicos

Os objetivos específicos do trabalho são:

a) Solicitar o café através de um aplicativo *Android* ou *iOS*: Será possível selecionar opções como a hora em que o produto ficará pronto, a quantidade de açúcar e a quantidade de café desejada;
b) Criação de uma máquina com controle de temperatura: Através do *arduino* será possível manter a temperatura do café entre uma margem específica.
c) Criação de uma máquina que controle o fluxo de café: Através de duas válvulas solenoides será possível controlar o fluxo do café.

2. ESTADO DA ARTE

Com base em pesquisas levantadas, pode-se fazer relação com as seguintes cafeteiras já existentes:

➢ *Vita Evolution* Café[1]

O funcionamento desta máquina se dá a partir do momento em que o usuário seleciona a bebida. Diferente das máquinas convencionais, que utilizam moeda, seu uso é com um cartão chamado *Smart Card* (cartão recarregável que dispensa o uso de moedas).

A máquina possui um sistema em que libera o copo automaticamente, e dispõe de um *mixer* para que as bebidas fiquem espumosas.

Algumas bebidas disponíveis são: Café expresso, café carioca, café com leite, *cappuccino*, *mocaccino*, leite, chocolate, chá ou um sabor de sopa.

Imagem 1

➢ COLIBRI I5[2]

Esta máquina não necessita de um meio de pagamento para o seu funcionamento, seu processo se inicia quando o usuário seleciona a bebida desejada. Ela também libera o copo automaticamente.

(Imagem 2)

Estes são os modelos mais comuns de uma imensa lista desse tipo de cafeteira. Embora semelhantes, nenhuma tem todas as características da *Auto Coffee*, pois todas necessitam um contato físico com a cafeteira, através de um botão, para selecionar a bebida desejada, o qual difere da cafeteira aqui apresentada, onde a bebida será selecionada através do um aplicativo *android* ou *iOS*, sem a necessidade de estar na mesma localidade física.

3. Referencial teórico

Alguns conceitos e materiais preliminares foram pesquisados em busca de aplicação prática ao projeto. São eles:

3.1. Conceituação

3.1.1. Endereço de IP[3]

Endereço IP é um número de 32 bits (IPv4) escrita em base decimal cujos dígitos representam respectivamente: redes específica da internet, e na sequência o host de tal rede. É usado para identificar um dispositivo, de forma genérica, e representa um meio de comunicação quando se diz respeito à internet. Geralmente, é endereçado por um domínio (convertido a um nome específico, precedido da sigla para "*World Wide Web*" por um DNS - *Domain Name System*).

3.1.2. *Ethernet* [4] / *Mac Address* [5]

Ethernet: Essa tecnologia desenvolvida pela *intel* é um meio de transmissão/ recebimento de informações, dependente de instalação física, conexão elétrica e conexão lógica entre dispositivos.

Endereça dados e os transmite, mais recentemente identificando erros durante o tráfego dos dados (tecnologia "*fast ethernet*"). O *shield* aqui usado é um exemplo de *half duplex*, pois não transmite e recebe dados simultaneamente. Afinal, apenas um processo se fez necessário por vez.

O *Mac Address*, por sua vez, é um conjunto de 48 *bits* em formato hexadecimal que se conecta a um endereço gravando a informação em *hardware* - seja memória ROM, placa de rede, etc…

Obs: Tal qual o endereço de IP, o *Mac Address* é padronizado pela *IEEE - Institute of Eletrical and Eletronics Engeneers*, separando a primeira metade de *bits* para identificação do fabricante.

3.1.3. *Internet* das coisas[6]

Cada vez mais dispositivos portáteis - que não *notebooks* e *smartphones* - têm funcionado em conjunto com a rede mundial de acesso eletrônico, a *internet*.

Desde carros (conforme projeto conjunto entre *intel* e *ford*), oferecendo reconhecimento facial no intuito de oferecer a segurança à prova de furtos e também o conforto em receber recomendações de músicas e ajustes de inclinação condizentes ao perfil do consumidor; até elevadores que emitem relatórios de predição de problemas de cunho técnico.

O fato é: num futuro não muito distante, aquele conceito da "*internet* das coisas", proposto em 1999 por Kevin Ashton do *Massachusetts Institute of Technology*, está a cada dia mais próximo da vivência humana, presente em cada interação. Fato esse que levou o grupo a emitir um simples pedido de café pela *internet*, caracterizando o tremendo potencial que a rede mundial tem a oferecer com seu envio de dados (especialmente ao se considerar que tais dados indicam até mesmo o horário de interesse, previamente definido por um instrumento altamente difundido como um *smartphone*).

3.1.4. Redes de computadores [7]

Esta que geralmente é definida de forma errônea como uma série de computadores interligados para *apenas* trocar informações é na verdade uma área de estudo que busca compartilhar através de dois ou mais dispositivos (vulgos "nós") uma mesma base de recursos ("*hardware*" e troca de mensagens) através de protocolos.

Todos eles dependem de endereçamento, meio e protocolo, sendo a *internet* (rede aqui explorada) apenas uma das formas de rede. Estas podem trocar informações através de diversos métodos, tais como satélites, fios de cobre e ondas de rádio.

3.1.5. Sistemas embarcados[8]

Sistemas embarcados são dispositivos (inclusive os analógicos) que, apesar de não serem computadores pessoais, nem estiverem diretamente ligados a tais; possuem memória, processador, por vezes interface e dispositivos de armazenamento (como memória *flash*).

Geralmente seu processador é mais simples/barato e seus microcontroladores e *chips* desempenham função única, isentas de multitarefas.

Sistemas embarcados possuem vasta aplicação que variam desde bonecos que emitem som, até carros que emitem informações e sistemas de medição de nível d'água.

3.1.6. Impacto Ambiental[9]

Impacto ambiental é definido segundo a resolução 001 de Conama de 1986 como "*qualquer alteração das propriedades físicas, químicas e biológicas do meio ambiente, causada por qualquer forma de matéria ou energia resultante das atividades humanas que, direta ou indiretamente, afetam a saúde, a segurança e o bem-estar da população; as atividades sociais e econômicas; a biota; as condições estéticas e sanitárias do meio ambiente; e a qualidade dos recursos ambientais*". Nesse ponto, a principal fonte de impacto ambiental que compõe o trabalho desenvolvido diz respeito ao descarte de materiais, que, em conjunto do conceito de "sustentabilidade" a seguir definido, é explorado na continuidade.

3.1.7. Sustentabilidade[10]

Sustentabilidade é um conceito de muita extensão, mas, em base, representa o equilíbrio na exploração e sua consequente "reposição", no que concerne aos recursos naturais, estes esgotáveis.

3.2. Embasamento teórico

Dentro da viabilidade do orçamento e conforme pesquisa preliminar, fez-se necessário o uso de válvulas para passagem do material a ser consumido, motores

para conversão de energia elétrica em mecânica (e consequente movimentação), dispositivos de envio de dados comentados em itens anteriores. Estes estendem-se à necessidade de alimentação de energia e chaveamento. As escolhas para essas funções indispensáveis ao projeto são descritas na continuidade, em especial no item 5.

3.3. Diagramas

Abaixo, detalhes sobre os diagramas concernentes ao projeto.

3.3.1. Diagrama de funcionamento

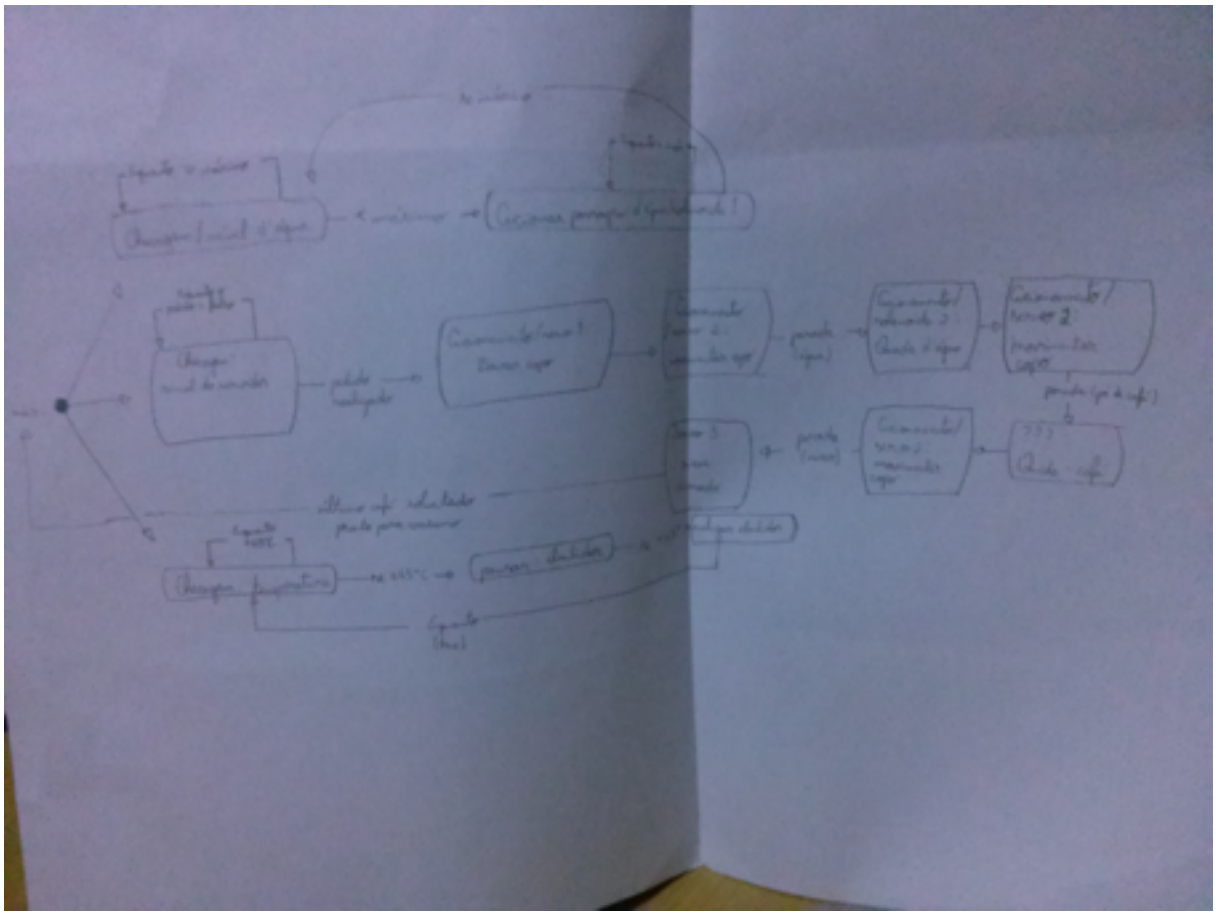


Imagem 3 - diagrama

Obs.: Lembrando que os delays entre as etapas aqui descritas foram ajustados no *arduino* programaticamente.

3.3.2. Máquina de estados

A máquina de estados funciona em conformidade ao diagrama acima, sendo implementado no *arduino* conforme anexos 8.1 e 8.2, discutidos no item 5, “o projeto”.

3.4. Aplicação da pré-definida sustentabilidade

Conceitos explorados no item 7 são aqui introduzidos.

3.4.1. Lei do chumbo[11]

Apelidado de “lei do chumbo”, essa diretiva de origem europeia (datando de 2006) — que não é uma lei ainda —, RoHS (*Restriction of Certain Hazardous Substances*, livremente traduzido como: Restrição de Certas Substâncias Perigosas). Ela envolve, ainda, os elementos cádmio, mercúrio, o cromo hexavalente, alguns éteres difenil-polibromados e bifólios polibromados.

As seis substâncias aqui tratadas, afinal, põe em risco a vida de cada ser cujo *hábitat* é composto por acúmulo de materiais descartados e compostos por elas. Além disso, a RoHS vem sendo levemente aplicada fora da União Europeia (nos Estados Unidos, por exemplo, o estado da Califórnia passou a aderi-la, lembrando que este compõe grande parte do lixo eletrônico por conter o vale do silício).

Em função da alta difusão de tais materiais na indústria em momentos anteriores à sua proibição, essa diretiva teve de vir acompanhada de um apoio para o que já existia dos materiais - o WEEE, discutido na sequência.

Vale lembrar que justamente por suas aplicações, mais e mais pesquisadores têm se empenhado em descobrir maneiras alternativas de produção de determinados produtos, tais como a simples solda que não só é composto por estanho, como por alta concentração de chumbo. Aquele substituído por estanho e cobre não tem os mesmos resultados, por exemplo.

3.4.2. Descarte de materiais eletrônicos (WEEE)[12]

"*Waste Electrical and Electronic Equipment recycling*" (em tradução literal, "Reciclagem de equipamentos elétricos e eletrônicos em descarte") veio para auxiliar o já citado RoHS com relação ao que existia previamente à lei ambiental.

As 2 milhões de toneladas de equipamentos eletrônicos anualmente rejeitados no Reino Unido, por exemplo, são tratados para que a saúde daqueles que vivem em nosso meio não tenham mais ameaças à vida do que no cenário atual. Quase toda a reciclagem se faz em cima de objetos possuidores dos seis itens citados na lei do chumbo.

4. METODOLOGIA

Para a realização deste trabalho o grupo subdividiu atividades semanais para cada integrante desempenhar individualmente. Durante os momentos em sala de aula os integrantes do grupo discutiam o que tinham conseguido realizar durante a semana para então planejar as atividades da semana seguinte.

Algumas atividades envolveram a aquisição de matérias e componentes, muitos destes comprados via *internet*. Desta forma, após a realização de algum pedido e enquanto o produto estivesse a caminho, eram realizadas pesquisas e estudos de exemplos utilizando o material, que após a entrega eram testados na prática.

Para a utilização do módulo wifi ESP8266, testamos algumas bibliotecas já prontas que não funcionaram como gostaríamos e não atendiam as nossas necessidades. Frente a este problema o grupo decidiu criar uma espécie de biblioteca própria em C++ para o projeto em, enviando em seu núcleo comandos AT via comunicação serial. Para o desenvolvimento do método de envio de comandos os vídeos de exemplo disponibilizados pelo professor foram de grande ajuda. Como o código *arduino* ficou relativamente grande, o grupo optou por utilizar a *IDE* de desenvolvimento Visual Studio 2013 por ser um software mais robusto de desenvolvimento quando comparado com a IDE padrão do Arduino.

Para o projeto da placa de circuito impresso no formato de *Shield* de Arduino o grupo decidiu por utilizar o software *Fritzing* devido a sua facilidade de montar um esquemático para depois desenvolver o design da placa já com a *pingem* do *arduino* correta, tendo em vista que o software já foi desenvolvido com o foco de permitir a criação de projetos de *Shields*.

De modo geral, a equipe utilizou de exemplos de projetos, tutoriais e vídeo-aulas encontradas na internet como base para desenvolver suas soluções.

5. O projeto

O projeto inicial visava atender à necessidade de uma pessoa atarefada através da possibilidade de marcar rapidamente pela internet uma hora (via *smartphone*) para preparo de café automatizado, em conformidade à típica pressa do ritmo de vida do homem contemporâneo.

5.1. Projeto mecânico

Uma breve discussão acerca do projeto mecânico.

5.1.1. Pré-projeto

Inicialmente, a prototipagem feita na ferramenta *solidworks* abrangeu apenas um recipiente triplo - para conter água, pó de café e açúcar, respectivamente.

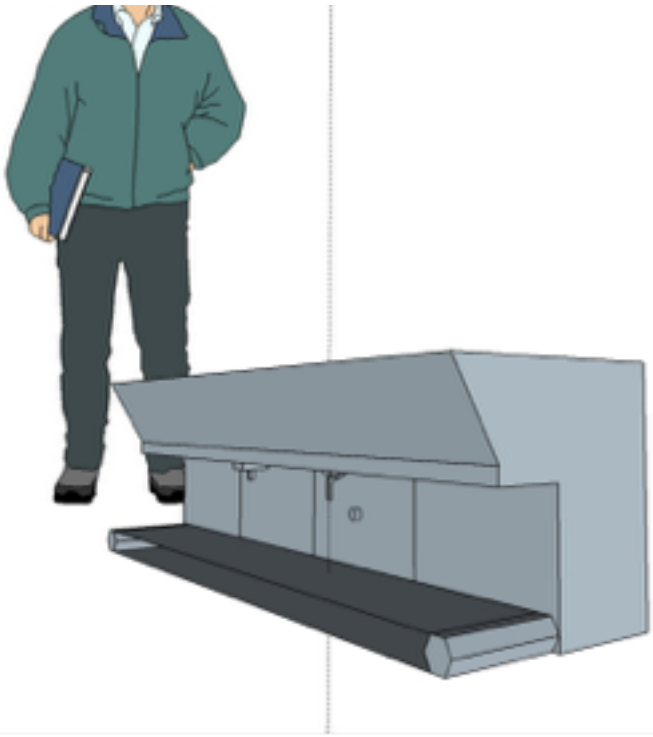


Imagem 4

5.1.2. Problemas e soluções empregadas

No decorrer do projeto fez-se necessária a substituição da esteira de rolagem pelo uso de *servo*-motores para retirada e locomoção de copo plástico, de uma fonte

diferente (vulgo “porta-copos”). Tais motores atrasaram a aplicação da parte mecânica devido à área de atuação deles, pouco definida e, portanto, implicando em pouca noção do espaço mínimo e máximo concebido.

Na sequência, foram estabelecidos recipientes separados e provisórios para prototipagem, conforme as fotos da próxima seção, em disparidade ao conjunto de recipientes inicial. Sejam os recipientes improvisados em suporte de madeira, de passagem d’água; os de pó de café liberados pela rotação de uma broca, ou a parede que continha o servo-motor, todo componente inicial do projeto mecânico foi substituído.

O último problema observado está relacionado ao porta-copos. A angulação de abertura exigia uma demanda de força maior que a suportada pelo *servo*. Tal angulação não poderia ser mudada e, comparada à distância total (“raio” da fórmula de momento angular), a retirada da mola mostrou-se mais conveniente ao propósito de solucionar a abertura desse componente mecânico.

5.1.3. Materiais das soluções empregadas

Imagem 5: recipientes d’água, cujas válvulas e sensores foram vedados com *sugru* e ebulidor foi deixado “solto” sem encostar nos demais componentes.

Imagem 6 - Porta-copos e braço mecânico do servo-motor em sua parede improvisada.

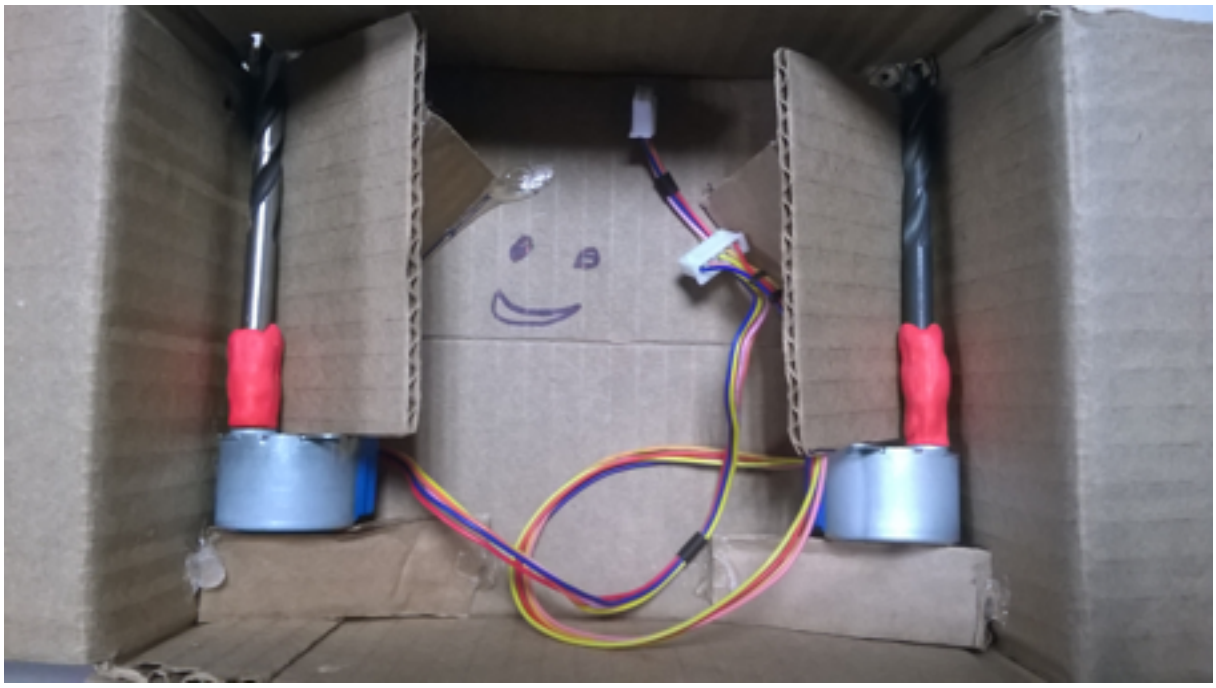


Imagem 7 - Liberação dos pós de açúcar e café por motor de passo, preso à broca que não escapa por estar grudada ao material conhecido como *sugru*.

5.2. Projeto eletroeletrônico

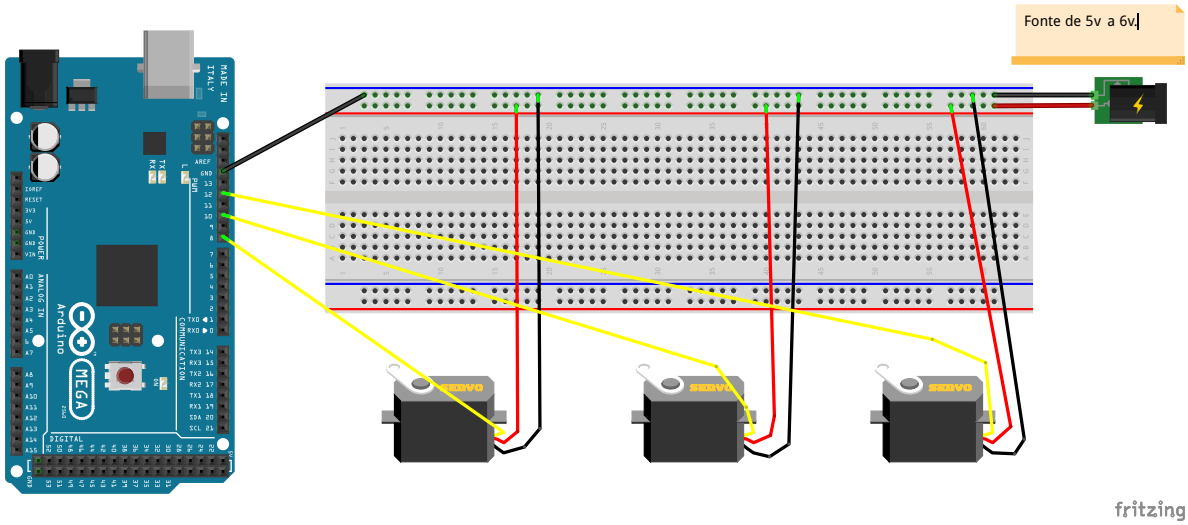
Discussão dos detalhes envolvidos nos esquemas elétricos.

5.2.1. Pré-projeto

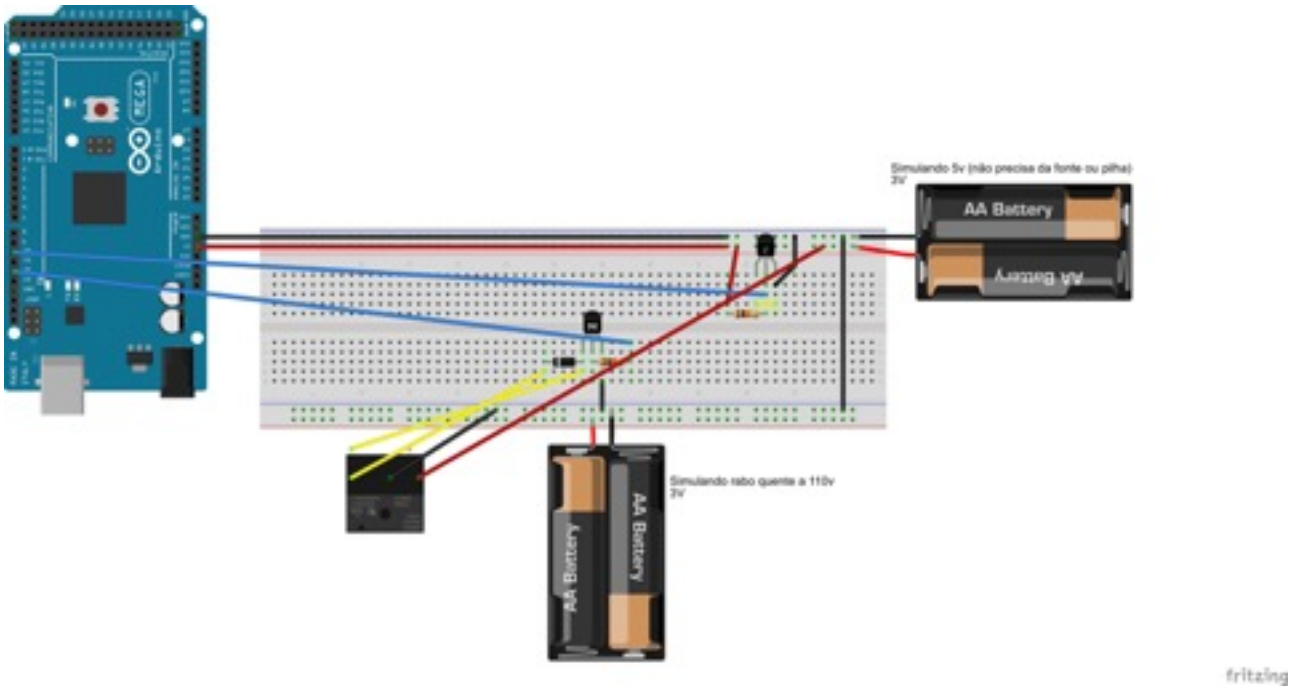
O pré-projeto incluiu a busca por materiais adequados ao *dispenser* dos materiais contidos nos recipientes, a busca por formas de movimentação de copos, de aquecimento da água e mistura dos ingredientes. Nesse quesito, apenas o sensor de fluxo d''água e o motor de passo da antiga esteira foram substituídos (por medição via “boia" e *servo*-motores, respectivamente). No final, foi adicionado um motor de passo ao *dispenser* de pó de café, que não foi incluso num momento primário.

5.2.2. Diagramas elétricos/eletrônicos empregados e seus materiais

Os servo-motores possuem uma simples conexão direta à fonte e passagem de dados PWM, conforme a imagem 8:

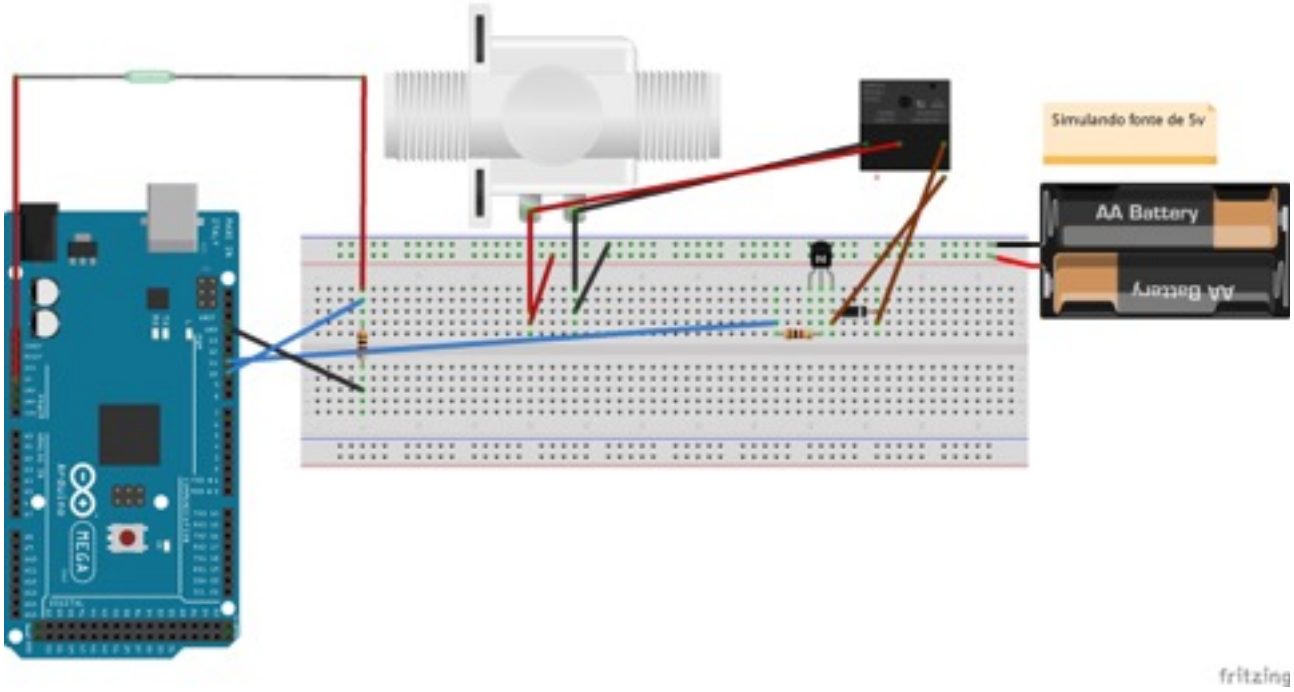


O comando de *arduino* responsável por acionamento ou não do aquecimento via ebulidor se deu através da conexão do mesmo com um relé 12V, controlado por um pino PWM definido programaticamente. O esquema da imagem 9 abaixo pode ser observado nos anexos também, em conjunto com o funcionamento das válvulas!



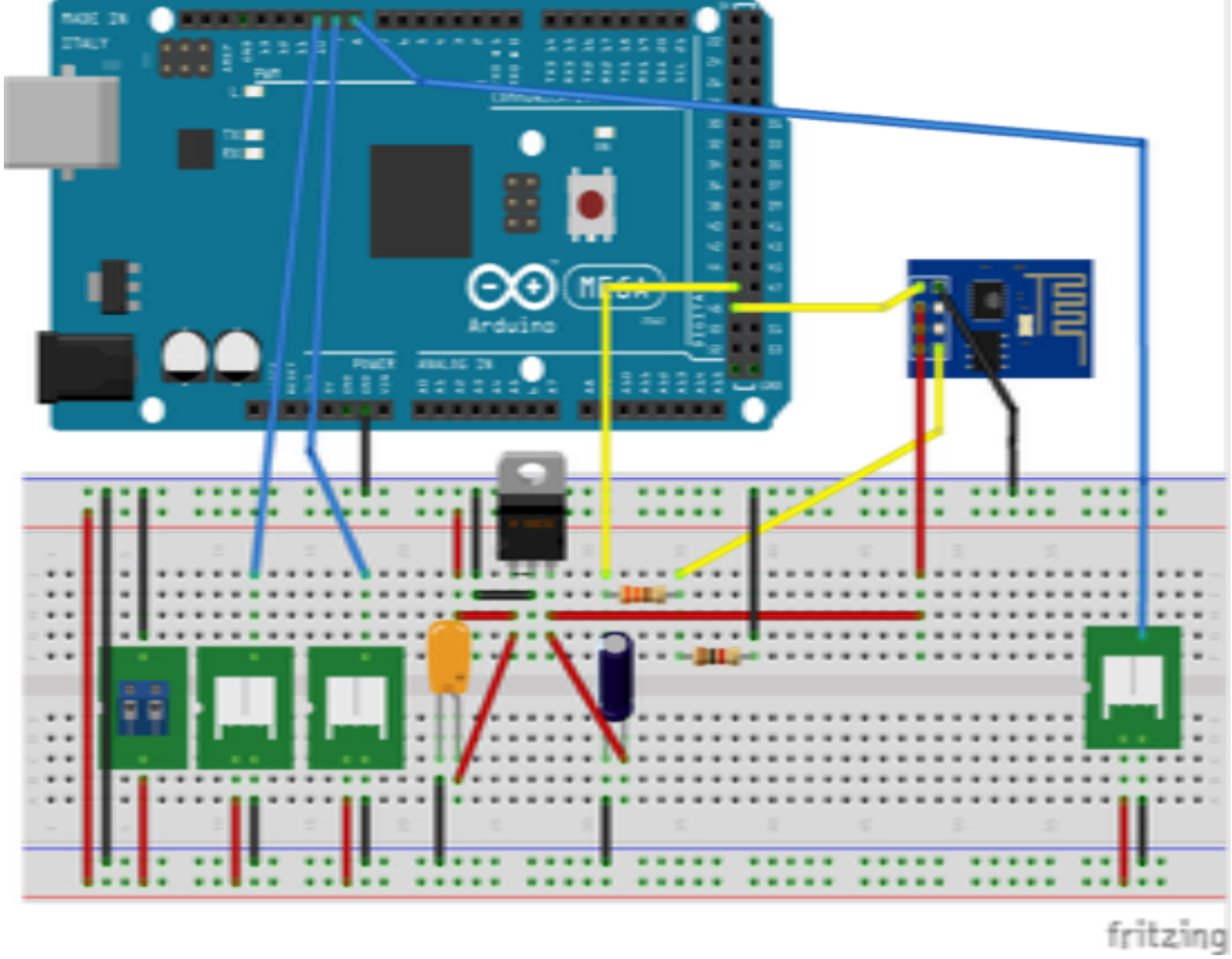
Esse ebulidor funciona em temperaturas pré-determinadas, retiradas do sensor à prova d'água DS18B20 (parecido com um transistor no esquema acima).

Em seguida, observamos na imagem 10 a válvula solenoide controlada pelo conjunto do transistor, diodo e relé, que acionam ou não a passagem de água programaticamente em conexão à lógica da boia, apresentada na subseção "*software*":



E finalmente, aqui contemplamos o esquemático do ESP, igualmente controlado por *arduino* conforme o esquemático a seguir - que foi convertido para a placa de circuito impresso do anexo F.

Obs.: Funcionamento com tensão regulada e com divisor resistivo para comunicação serial com *arduino*, conforme imagem 11.



5.3. Projeto de software

5.3.1. Problemas iniciais

Ao passo que se pesquisavam os produtos, novas modelagens de programação em *arduino* eram buscadas para aplicação ao objeto de estudo - a cafeteira.

Alguns dos problemas deparados foram: fórmulas aplicadas à biblioteca *OneWire*, máquina de estados para servo-motores e conexão à internet.

5.3.2. Modelos de software e soluções

Os modelos de software foram separados basicamente em três: um para passagem de dados do/para o ESP, um para acionamento de motores e outro para liberação dos produtos e aquecimento. Subdivisões foram incluídas aqui.

Começando pelos *servo*-motores, a solução emprega em formato de máquina de estados o funcionamento de três unidades (código no anexo A). O primeiro servo é responsável por liberar e agarrar o copo (denotado pelo código como *myServo3*).

O segundo *servo* serve para passear entre as posições de água, café e açúcar (denotado pela variável *myServo*) e, finalmente, o último (*myServo2*) mantém no *mixer* até que o processo seja repetido. É importante notar que os *delays* giram

em torno de pelo menos 5 segundos para que suas funções sejam desempenhadas em tempo condizente (valores a serem testados e alterados*).

No anexo B, passamos para controle e aquecimento de água. Este último é feito quando o sensor DS18B20 acusa determinadas faixas de temperatura, atribuindo *high* na tensão de saída do ebulidor. A saída em graus *celsius* se deu através de uma série de fórmulas que a biblioteca *Open Source* do *arduino* forneceu, envolvendo 9 *bytes* de saída em um vetor que compõe a informação numérica de saída.

Essa atribuição de nível alto/baixo na tensão é aplicada igualmente na passagem da solenoide de entrada de água no reservatório, por meio do nível máximo que o sensor de nível indica e, no caso da saída da água, será diretamente acionada durante um *delay* pré-definido no momento requisitado pelo ESP, de forma a encher a maior parte do copo.

Aproveitando o assunto, o ESP é implementado no anexo C (arquivo *.h* / *.cpp*) de forma a gerar um link que guarda suas solicitações (horário do dia a ser preparado o café e quantidades de açúcar e café), que quando acessado emite as informações necessárias ao início dos procedimentos do *arduino*, no código da sequência.

O motor de passo do anexo seguinte simplesmente se propõe a, uma velocidade determinada por código, liberar os pós de açúcar e café.

E, por último, um acesso alternativo ao servidor de requisição, encontra-se o código de implementação *iOS*, para uso em dispositivos móveis, conforme o objetivo inicial de praticidade aqui proposto.

6. RESULTADOS

Como resultado dos testes experimentas e pesquisas realizadas ao longo do desenvolvimento do projeto, a equipe produziu uma placa de circuito impresso no formato de *Shield* de a*rduino* para controle de *servo*-motores e regulação de tensão para o módulo *wi-fi ESP8266*, uma formar de puxar a alavanca do porta copos permitindo que através de um servo motor seja possível o realizar a liberação de copos, o desenvolvimento de uma classe em C++ para operação do módulo de internet sem fio como servidor, recebendo dados via método GET, leitura de temperatura de água usando um sensor a prova d'água e acionamento de válvula solenoide para controle de passagem de água de acordo com o nível lido por uma bóia.

- Principais problemas: indicação do horário para o café estar pronto, no caso do *Android;* movimentação do porta-copos via *servo;* queda dos pós e integração dos itens entre todos os integrantes.
- O que não foi feito: Integração de itens devido aos projetos mecânicos inconsistentes uns com os outros. Liberação a velocidade condizente (considerando a limitação do motor de passo) no que diz respeito aos pós de café e açúcar, programação *android* para horário selecionado.

7. IMPACTO AMBIENTAL

Como anteriormente resumido, "impacto ambiental" é a alteração no meio ambiente ou em algum de seus componentes por determinada ação ou atividade humana. O objetivo de se estudar os impactos ambientais é, principalmente, o de avaliar as consequências destas ações para que possa haver a prevenção da alteração da qualidade do ambiente após a execução dessas ações.

Relacionado ao impacto ambiental, estão os Resíduos eletrônicos, que é o termo utilizado para qualificar equipamentos eletroeletrônicos descartados ou obsoletos. Atualmente no paraná, mais precisamente em Curitiba, contamos com duas empresas recicladoras de eletroeletrônicos, sendo elas: Mega Reciclagem de Materiais Ltda e Bulbox - Triturador e “descontaminador" de Lâmpadas Fluorescentes. (Fonte: CEMPRE[13]).

Uma das medidas tomadas que colaborou com a diminuição de contaminações por equipamentos eletrônicos foi a criação da diretiva “Lei do Chumbo”, conhecida como “Lei sem Chumbo” (RoHS - *Restriction of Certain Hazardous Substances, Restrição de Certas Substâncias Perigosas*). Esta diretiva está em vigor a partir do dia 1º de Julho de 2006 e consiste na proibição do uso de materiais perigosos na confecção de alguns produtos, como: Cádmio (Cd), Mercúrio (Hg), Cromo Hexavalente (Cr(VI)), Bifenilos Polobromados (PBBs), Éteres Definel-Polibromados (PBDEs) e Chumbo (Pb).

O projeto *Auto Coffee* utiliza materiais recicláveis e não recicláveis (como por exemplo a solda e a pilha, no segundo caso). O maior problema é que a solda tradicional é composta de 63% de estanho (Sn) e 37% de chumbo (Pb), o que torna descartável qualquer produto que o utilize. Por outro lado, o projeto é composto por uma grande quantidade de produtos que podem ser reutilizados, sem precisar passar pelo processo de reciclagem, como é o caso do porta-copos, motores, *arduino* e etc.

Dentre os materiais utilizados, todos estão de acordo com a lei do chumbo, e grade parte pode ser reutilizado. Alguns itens podem ser descartados nos lixos recicláveis comum, mas devem estar devidamente separados por categoria (metal, plástico, etc), outros itens devem ser descartados nos Centros de triagem de reciclagem, onde terá um destino correto.

Hoje o Paraná conta com diversos centros de triagem e reciclagem, principalmente na capital e regiões metropolitanas.

Além do centro de triagem de reciclagem e da coleta regular do lixo reciclável, podemos também contar com cooperativas de catadores, pois estes também dão o destino certo para os materiais, seja para o centro de triagem ou para o centro de reciclagem comum.

Veja abaixo uma tabela de descartes dos materiais utilizados:

Materiais	Reciclável	Reutilizável	Correto Descarte
Arduíno e Shields	Sim	Sim	Centro de triagem de reciclagem
PCI	Sim	Não	Centro de triagem de reciclagem
Motores	Sim	Não	Centro de triagem de reciclagem
Porta-Copos	Sim	Não	Lixo reciclável comum
Canaletas	Sim	Não	Lixo reciclável comum
Válvulas	Sim	Não	Centro de triagem de reciclagem
Pilha	Sim	Não	Centro de triagem de reciclagem
Mixer	Sim	Não	Parte no centro de triagem de recliclagem e parte no Lixo reciclável comum
Componentes Eletrônicos	Sim	Não	Centro de triagem de reciclagem
Broca	Sim	Sim	Lixo reciclável comum

8. CONSIDERAÇÕES FINAIS

Na produção deste trabalho, pudemos colocar na prática alguns dos conhecimentos adquiridos nos semestres iniciais do curso e participar do processo de criação de um projeto, desde a concepção de sua ideia, planejamento, produção e documentação. Pudemos perceber a importância de cada etapa e verificar como o trabalho em conjunto e o esforço de todos os membros do grupo é importante para se obter um bom resultado final.

O projeto do *Auto Coffee* teve início a partir de uma conversa com o professor orientador da turma onde o grupo apresentou a ideia de uma máquina para preparação de chá. Após o diálogo, foi então decida pela máquina de café que permitiria a realização de pedidos remotos a fim de poupar tempo do utilizador proporcionando que ele possa chegar em casa e obter sua bebida já pronta.

O trabalho então se sucedeu pela divisão de atividades entre os membros do grupo, onde pudemos perceber que o comprometimento individual semanal foi de extrema importância para as produções individuais necessárias para o produto final. Um fator que o grupo percebeu ser de extrema importância foi o gerenciamento do tempo, obrigatório para uma boa organização, pois como os membros da equipe possuem compromissos diários externos, o tempo livre para dedicação ao projeto foi de certa forma limitado.

Algumas alterações tiveram que ser feitas e uma das funcionalidades, como por exemplo a escolha do horário para a bebida ser preparada foi removida no *app* para *android*.

Ainda sobre este mesmo tema, para futuro, apresentamos algumas ideias de implementações que seriam interessantes de serem feitas a fim de estender a funcionalidade do produto: Permitir a escolha de mais de um tipo de café; inserção de leite (em pó ou líquido); possibilidade de adoçar a bebida com adoçante; preparação de uma quantidade maior (mais copos ou uma jarra); implementação do monitoramento da quantidade de produto (café, açúcar, etc ...) disponível com aviso para o utilizador quando algum deles estiver no fim para que seja reabastecido; geração de relatórios de pedidos realizados ao longo de um período de tempo.

Isso considerando uma melhor integração das partes que compõe o todo, uma vez que estavam prontas/funcionais, mas não interagindo umas com as outras.

REFERÊNCIAS

1 - Disponível em: <http://www.cafeautomatic.com.br/pt-br/maquinas-vending/vita-evolution-cafe>

2 - Disponível em: <http://sathicafe.com.br/site/colibri-c4/>

3 - WIKIPÉDIA. **Endereço IP**. Baseado em <http://pt.wikipedia.org/wiki/Endereço_IP> Acesso em: 4 jun. 2015.

4 - CIANET. **Tecnologia Ethernet**. Baseado em <http://www.cianet.ind.br/pt/produtos/tecnologias/tecnologia-ethernet/> Acesso em: 4 jun. 2015.

5 - WIKIPÉDIA. **Endereço MAC**. Baseado em <http://pt.wikipedia.org/wiki/Endereço_MAC> Acesso em: 4 jun. 2015.

6 - TECHTUDO. **Internet das coisas: entenda o conceito e o que muda com tecnologia**. Baseado em <http://www.techtudo.com.br/noticias/noticia/2014/08/internet-das-coisas-entenda-o-conceito-e-o-que-muda-com-tecnologia.html> Acesso em: 4 jun. 2015.

7 - WIKIBOOKS. **Redes de computadores**. Baseado em <http://pt.wikibooks.org/wiki/Redes_de_computadores/Introdução> Acesso em: 5 jun. 2015.

8 - HARDWARE. **Entendendo sistemas embarcados**. Baseado em <http://www.hardware.com.br/artigos/entendendo-sistemas-embarcados/> Acesso em: 4 jun. 2015.

9 - MUNDO EDUCAÇÃO. **Impactos ambientais**. Baseado em <http://www.mundoeducacao.com/biologia/impactos-ambientais.htm> Acesso em: 5 jun. 2015.

10 - ATITUDES SUSTENTÁVEIS. **Sustentabilidade**. Baseado em <http://www.atitudessustentaveis.com.br/sustentabilidade/sustentabilidade/> Acesso em: 5 jun. 2015.

11 - WIKIPÉDIA. ***Rohs***. Baseado em <http://pt.wikipedia.org/wiki/Rohs> Acesso em: 5 jun. 2015.

12 - HSE. **Waste Electrical**. Baseado em <http://www.hse.gov.uk/waste/waste-electrical.htm> Acesso em: 5 jun. 2015.

13 - Disponível em <http://www.cempre.org.br> Acesso em: 15 jun. 2015

ANEXO A – SOURCE (*SERVO*-MOTORES):

```
#include <Servo.h>

Servo myservo;
Servo myservo2;
Servo myservo3;

int pos3 = 180;    // Posição inicial do servo
int pos2 = 1;
int pos = 1;    // Posição inicial

void setup()
{
  myservo.attach(8);   // Pino Branco (laranja)
  myservo.write(pos);
  digitalWrite(8, LOW);

  myservo2.attach(12);  // Pino utilizado (branco/laranja)
  myservo2.write(pos2);
  digitalWrite(12, LOW);

  myservo3.attach(10);  // Pino utilizado (branco/laranja)
  myservo3.write(pos3);
  digitalWrite(10, LOW);

  Serial.begin(9600);
}

void loop () {
  Serial.println ("\n\nAguardando pedido...\n\nDigite 1 e aperte ENTER para iniciar o
processo.\n\n");
```

```
while (Serial.available() == 0);
int val = Serial.read();


  if (val == '49'){
    digitalWrite(8, HIGH);
    digitalWrite(10, HIGH);
    digitalWrite(12, HIGH);
  }
    else {
    digitalWrite(8, LOW);
    digitalWrite(10, LOW);
    digitalWrite(12, LOW);
}
  if (pos3 = 180){
    myservo3.write(pos3);
    delay (2400);
    Serial.println ("\nCopo sendo liberado.\n");
  }

  for(pos3 = 170; pos3 >= 110; pos3 -= 1) // pegar copo
{
  myservo3.write(pos3);
  delay(100);
}

for(pos3 = 120; pos3<=180; pos3 += 1)
{
  myservo3.write(pos3);
  delay(50);
}
```

```
 for(pos = 1; pos <= 22; pos += 1) // Vai de 1º a 22º
 {
   myservo.write(pos);
   delay(200);
 }

 if(pos = 22)     // ao chegar na posição de 22º, para por 5 segundos para pegar o
copo
 {

   myservo.write(pos);
   delay(2000);                    // Tempo para pegar o copo
 }

 for (pos = 22;pos <= 66; pos+=1) // Vai da posição de 22º para posição 66º
 {
   myservo.write(pos);
   delay (200);
  }

  if (pos = 66) // ao chegar na posição de 66º, para por 5 segundos para pegar a
água
  {
    Serial.println("\nPegando a água.\n");
    myservo.write(pos);
    delay(2000);  // Tempo para pegar a Água
  }

  for (pos = 66; pos <= 110; pos+=1) // vai da posição de 66º´para a posição de 110º
  {
    myservo.write(pos);
    delay (200);
```

```
  }

  if (pos = 110) // Ao chegar na posição de 110º, para por 5 seguntos para pegar o
Café
  {
    Serial.println("\nPegando café.\n");
    myservo.write(pos);
    delay (2000); // Tempo para pegar o Café
  }

  for (pos = 110; pos <= 150; pos+=1) // Vai da posição de 110º para a posição de
150º
  {
    myservo.write(pos);
    delay(200);
  }

  if (pos = 150) // Ao chegar na posição de 150º, para por 5 segundos para pegar o
açúcar
  {
    Serial.println("\nPegando o Açucar\n");
    myservo.write(pos);
    delay(2000); //Tempo para pegar o acúcar
  }

  for (pos = 150; pos <= 179; pos+=1) // Vai da posição de 150º para a posição de
179º
  {
  myservo.write(pos);
  delay(200);
  }
```

```
  if (pos = 179) // Ao chegar na posição de 179º, para por 10 segundos para o Mixer mistura os produtos
  {
    myservo.write(pos);
    delay(5000); // Tempo para fazer a mistura dos produtos
  }



   for(pos2 = 1; pos2 <= 85; pos2 += 1) //Vai da posição de 1º até 85º (descer o Mixer)
 {
   myservo2.write(pos2);
   delay(100);
 }

 if (pos2 = 85)       //Quando a posição chegar até 85º, fica parado por 12 segundos para o Mixer realizar as misturas
 {
   Serial.println("\nMisturando os produtos\n");
   myservo2.write(pos2);
 delay (5000);
 }

 for(pos2 = 85; pos2>=1; pos2 -= 1)     // Vai para a posição de 0º e aguarda o sinal para realizar um novo processo
 {
   myservo2.write(pos2);
   delay(100);
 }
```

```
      if (pos = 179)  //Vai para a posição de 1º e aguarda o sinal para reiniciar o
processo.
  {
  pos = 1;
  myservo.write(pos);
  delay(100);
  }
}
```

ANEXO B - AQUECIMENTO/PASSAGEM D'ÁGUA:

```
#include <OneWire.h>
OneWire  ds(10);  // pino 10

int RaboQuente = 9;
int pinNivelAgua = 11;
int pinFluxoIn = 12;

void setup(void) {
  Serial.begin(9600);
  pinMode(RaboQuente,INPUT);
  pinMode(pinNivelAgua,INPUT);
  pinMode(p'nFluxoIn,OUTPUT);
}

void loop(void) {
  byte i;
  byte present = 0;
  byte type_s;
  byte data[12];
  byte addr[8];
  float celsius;

  if ( !ds.search(addr)) {
    ds.reset_search();
    delay(250);
    return;
  }

  for( i = 0; i < 8; i++) {
    Serial.write(' ');
  }
```

```
’  if (OneWire::crc8(addr, 7) != addr[7]) {
      return;
  }

  ds.reset();
  ds.select(addr);
  ds.write(0x44, 1);
  delay(1000);

  present = ds.reset();
  ds.select(addr);
  ds.write(0xBE);

  for ( i = 0; i < 9; i++) {    //precisa de 9 bytes
    data[i] = ds.read();

  }

  int16_t raw = (data[1] << 8) | data[0];
  if (type_s) {
    raw = raw << 3;
    if (data[7] == 0x10) {
      raw = (raw & 0xFFF0) + 12 - data[6];
    }
  }
else {
    byte cfg = (data[4] & 0x60);
    if (cfg == 0x00) raw = raw & ~7;
    else if (cfg == 0x20) raw = raw & ~3;
    else if (cfg == 0x40) raw = raw & ~1;
  }
  celsius = (float)raw / 16.0;
```

```
  Serial.print("Temperature = ");
  Serial.print(celsius);
  Serial.print(" ºC ");
  Serial.println("\n\n");

  if(celsius < 55 && celsius > 45){
    digitalWrite(RaboQuente, HIGH);
    Serial.println("Liguei Rabo Quente");
  }

  else{
    digitalWrite(RaboQuente, LOW);
    Serial.println("Desliguei Rabo Quente");
  }

  if(digitalRead(pinNivelAgua)==HIGH){
    digitalWrite(pinFluxoIn,LOW);
    Serial.println("Valvula superior fechada");
  }
   else{
    digitalWrite(pinFluxoIn,HIGH);
    Serial.println("Valvula superior aberta");
  }
}
```

ANEXO C - SOURCE (ESP):

.h :

```
#ifndef ESP8266Server_h
#define ESP8266Server_h
#define ESP8266_DEBUG true
class Dado{

private:
        String nome, dado;
        int tamanho;
public:
        void setNomeTamanho(String n, int t);
        void setDado(String d);
        String getNome();
        String getDado();
        int getTamanho();
};

class ESP8266Server{
private:
        AltSoftSerial esp8266;
        String ultimoBuffer;
        String ip;
        Dado dados[10];
        int qtdDados = 0;
        byte reset();
        boolean enviarComando(String comando, int timeout, String ok);
public:
        void iniciar(int velocidade);
        void iniciarComRedePropria();
        void iniciarConexaoRoteador(String SSID, String senha);
        void iniciarServidorWeb();
        void descobrirIp();
```

```
        String getIp();
        void adicionarDadoValidar(String nome, int tamanho);
        int getQtdDados();
        String getDado(String nome);
        boolean verificarRequisicao();
};

#endif
```

.cpp :

```
//#include <Arduino.h>
#include <AltSoftSerial.h>
#include "ESP8266Server.h"
void ESP8266Server::iniciar(int velocidade){
        Serial.begin(velocidade);
        esp8266.begin(velocidade);
        if (ESP8266_DEBUG)
                Serial.println("Serial iniciada");
}

byte ESP8266Server::reset(){
        boolean result;
        do{
                result = enviarComando("AT+RST\r\n", 2500, "ready\r\n");
        } while (!result);
}

void ESP8266Server::iniciarComRedePropria(){
        this->reset();
        enviarComando("AT+CWMODE=2\r\n", 1000, "");
```

```
        this->descobrirIp();
}



void ESP8266Server::iniciarConexaoRoteador(String SSID, String senha){
        boolean result;
        this->reset();
        enviarComando("AT+CWMODE=1\r\n", 1000, "");
        do{
                result = enviarComando("AT+CWJAP=\"" + SSID + "\",\"" + senha + "\"\r
\n", 15000, "OK\r\n");
        } while (!result);
        this->descobrirIp();
}

void ESP8266Server::descobrirIp(){
        boolean result;
        do{
                result'= enviarComando("AT+CIFSR\r\n", 1000, "OK\r\n");
        } while (!result);
        ip = ultimoBuffer.substring(14, ultimoBuffer.indexOf("\\r\\n\\r\\nOK\\r\\n"));
}

String ESP8266Server::getIp(){
        return ip;
}

void ESP8266Server::iniciarServidorWeb(){
        boolean result;
```

```
	do{
		result = enviarComando("AT+CIPMUX=1\r\n", 1000, "OK\r\n");
	} while (!result);

	do{
		result = enviarComando("AT+CIPSERVER=1,80\r\n", 1000, "OK\r\n");
	} while (!result);
}



void ESP8266Server::adicionarDadoValidar(String nome, int tamanho){
	dados[qtdDados].setNomeTamanho(nome, tamanho);
	qtdDados++;
}

boolean ESP8266Server::verificarRequisicao(){
	boolean result;
	boolean dadoEncontrado = false;
	String requisicao = "";
	int index;
	if (esp8266.available()){
		if (esp8266.find("+IPD,")){
			delay(1000);
			int connectionId = esp8266.read() - 48;
// subtract 48 because the read() function returns the ASCII decim'l value and 0 (the first decimal number) starts at 48
			//Carrega a primeira linha da requisição http em uma string
			while (esp8266.available()){
				char c = esp8266.read();
				requisicao += c;
				if (requisicao.length() >= 8 &&
```

```
requisicao.substring(requisicao.length() - 8).equals("HTTP/1.1")){
                                        break;
                                }
                        }
                        if (ESP8266_DEBUG)
 Serial.println("Requisicao separada para analise: " + requisicao);
                        //Valida se a string possui algum dos dados configurados para
serem capturados
                        for (int i = 0; i < qtdDados; i++){
                                index = requisicao.indexOf(dados[i].getNome());
                                //Existe o dado
                                if (index > 0){
                                        dadoEncontrado = true;
                                        while (!requisicao.substring(index, index +
1).equals("="))
                                                index++;
                                        //Serial.println("Substring de " + String(index + 1) +
" ate " + String(index + 1 + dados[i].getTamanho()));
                                        dados[i].setDado(requisicao.substring(index + 1,
index + 1 + dados[i].getTamanho()));
                                        if (ESP8266_DEBUG)
                                                Serial.println("Encontrado " +
dados[i].getNome() + ": " + dados[i].getDado());
                                        }
                                else{
                                        dados[i].setDado("");
                                }
                        }
                        String cipSend;
                        String webpage;
                        //------------INICIO
```

```
                webpage = "<html><head><title>AutoCoffee</title></
head><body>";
                cipSend = "AT+CIPSEND=";
                cipSend += connectionId;
                cipSend += “,”;
                cipSend += webpage.length();
                cipSend += "\r\n";
                do{
                        result = enviarComando(cipSend, 1500, "\n>");
                } while (!result);
                do{
                        result = enviarComando(webpage, 1500, "SEND OK\r\n");
                } while (!result);
                //------------CONTEUDO
                if (dadoEncontrado)
                        webpage = "<p>Pedido recebido com sucesso!</p>";
                else webpage = "<p>AutoCoffee <b><span style=
\"color:#0F0;\">OnLine</span></b></p>";
                cipSend = "AT+CIPSEND=";
                cipSend += connectionId;
                cipSend += ",";
                cipSend += webpage.length();
                cipSend += "\r\n";
                do{
                        result = enviarComando(cipSend, 1500, "\n>");
                } while (!result);
                do{
                        result = enviarComando(webpage, 1500, "SEND OK\r\n");
                } while (!result);
                //------------FIM
                webpage = "</body></html>";
                cipSend = "AT+CIPSEND=";
```

```
                    cipSend'+= connectionId;
                    cipSend += ",";
                    cipSend += webpage.length();
                    cipSend += "\r\n";
                    do{
                        result = enviarComando(cipSend, 1500, "\n>");
                    } while (!result);
                    do{
                        result = enviarComando(webpage, 1500, "SEND OK\r\n");
                    } while (!result);
                    //Fechando conexao
                    String closeCommand = "AT+CIPCLOSE=";
                    closeCommand += connectionId;
                    closeCommand += "\r\n";
                    enviarComando(closeCommand, 5000, "OK\r\n");
                    }
                }
        return dadoEncontrado;
}

int ESP8266Server::getQtdDados(){
        return qtdDados;
}


String ESP8266Server::getDado(String nome){
        for (int i = 0; i < qtdDados; i++){
                if (dados[i].getNome().equals(nome))
                        return dados[i].getDado();
                }
                return "";
}
```

```
boolean ESP8266Server::enviarComando(String comando, int timeout, String ok){
        String buffer = "";
        int tamanhoOk = ok.length();
        unsigned long inicio = millis();
        unsigned long fim = 0;
        boolean isOK = false;
        esp8266.print(comando);
        while ((inicio + timeout) > millis()){
                if (esp8266.available()){
                        buffer += (char)esp8266.read();
                        fim = millis();
                        if (tamanhoOk > 0 && buffer.length() >= tamanhoOk){
                                if (buffer.substring(buffer.length() -
tamanhoOk).equals(ok)){
                                        isOK = true;
                                        break;
                                }’                      }
                }
        }
        buffer.replace("\r", "\\r");
        buffer.replace("\n", "\\n");
        ultimoBuffer = buffer;

        if (ESP8266_DEBUG)
 Serial.println("("+String(fim-inicio)+"/"+timeout+")"+buffer);
        return isOK;
}

void Dado::setNomeTamanho(String n, int t){
```

```
        nome = n;
        tamanho = t;
}

String Dado::getNome(){'   return nome;
}

String Dado::getDado(){
        return dado;
}

int Dado::getTamanho(){
        return tamanho;
}

void Dado::setDado(String d){
        dado = d;
}
```

.ino :

```
//#include <Arduino.h>
#include <AltSoftSerial.h>
#include "ESP8266Server.h"

//Para Arduino UNO, conectar RX do ESP na porta 9 e o TX na porta 8. N□o usar PWM do 10.
//Para Arduino MEGA, conectar RX do ESP na porta 46 e o TX na porta 48. N□o usar PWM do 44 e 45.
ESP8266Server wifi;
void setup(){
        wifi.iniciar(9600);
        //wifi.iniciarComRedePropria();
```

```
        wifi.iniciarConexaoRoteador("","");
        wifi.iniciarServidorWeb();
        Serial.println("\nServidor iniciado com sucesso!\nDigite o seguinte IP no navegador: "
+ wifi.getIp() + "\n");
        wifi.adicionarDadoValidar("qtdCafe", 1);
        wifi.adicionarDadoValidar("qtdAcucar", 1);
}

void loop(){
        if (wifi.verificarRequisicao()){
                String qtdCafe = wifi.getDado("qtdCafe");
                String qtdAcucar = wifi.getDado("qtdAcucar");
                Serial.print("Preparar bebida com quantidade de cafe " + qtdCafe);
                Serial.println(" e quantidade de acucar " + qtdAcucar);
        }
}
```

ANEXO D - MOTOR DE PASSO:

```
#include <Stepper.h>
#define STEPPER_VOLTA_COMPLETA 2048
//32 Fixo, pinos na sequencia 1-3-2-4
Stepper motor(32, 8, 10, 9, 11);

void setup(){
  //Configura velocidade - Max ~ 600
  motor.setSpeed(500);
}

void loop(){
  motor.step(STEPPER_VOLTA_COMPLETA);
  delay(1000);

  motor.step(-STEPPER_VOLTA_COMPLETA);
  delay(1000);
}
```

ANEXO E - APLICATIVO iOS:

```
import UIKit
class ViewController: UIViewController {

    var tempo: UInt32!
    var endereco:String!
    var verificador:Int = 0
    var validos:[String] = ["192.168.4.1", "autocoffee.hol.es", "10.95.56.68"]
    var timer:NSTimer!
    var caminho: String!
    @IBOutlet weak var cafe: UISegmentedControl!
    @IBOutlet weak var acucar: UISegmentedControl!
    @IBOutlet weak var ip: UITextField!
    @IBOutlet weak var myDatePicker: UIDatePicker!

    override func viewDidLoad() {
        super.viewDidLoad()
        self.view.backgroundColor = UIColor(patternImage: UIImage(named:
"background.jpg")!)
       timer = NSTimer.scheduledTimerWithTimeInterval(1.0, target: self, selector:
Selector("loop"), userInfo: nil, repeats: true)
    }

    func loop(){
        let currentDate = NSDate()
        myDatePicker.minimumDate = currentDate
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
   }
```

```
    @IBAction func Confirmavalores(sender: AnyObject) {
        endereco = ip.text
        var dt1: NSDate = myDatePicker.date
        var dt2: NSDate = NSDate()
        var num = Int(dt1.timeIntervalSinceDate(dt2)-20) //leva 17 segundos a mais do que simplesmente hh:mm:00, ou seja: formato hh:mm:17
        if(num<1){
            num = 1
        }
        println(num)

        for(var i=0; i<self.validos.count; i++){
            if(self.ip.text == self.validos[i]){
                verificador = 1
            }
        }
        if(verificador == 0){
            var alert = UIAlertController(title: "Erro", message: "Esse endereço de IP não é um servidor válido!", preferredStyle: UIAlertControllerStyle.Alert)
            alert.addAction(UIAlertAction(title: "Tente novamente", style: UIAlertActionStyle.Default, handler: nil))
            self.presentViewController(alert, animated: true, completion: nil)
        }
        else if(verificador == 1){
            var alert = UIAlertController(title: "Aguarde", message: "Seu pedido foi processado. Aguarde o horário selecionado!", preferredStyle: UIAlertControllerStyle.Alert)
            alert.addAction(UIAlertAction(title: "Beleza!", style: UIAlertActionStyle.Default, handler: {
                (Action: UIAlertAction!) -> Void in
                let url = NSURL(string: self.caminho!)
```

```
            //let request = NSURLRequest(URL: url!)
            UIApplication.sharedApplication().openURL(url!)
        }))
        self.presentViewController(alert, animated: true, completion: nil)
        caminho = "http://" + ip.text + "/?qtdCafe=" +
(String((cafe.selectedSegmentIndex)+1)) + "&qtdAcucar=" +
(String((acucar.selectedSegmentIndex)))
        println(caminho)
    }
    else{ verificador = 0 }
  }
}
```

ANEXO F - PCI (ESP):

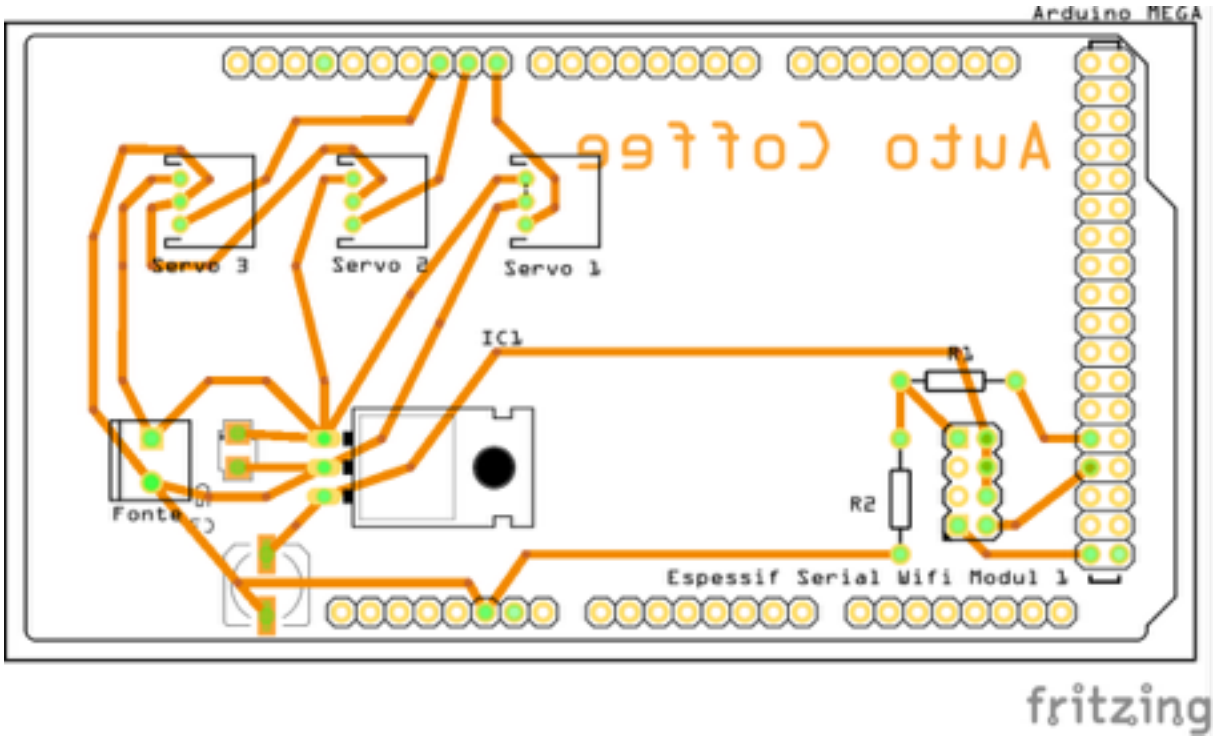


Imagem 12

ANEXO G - VISÃO PROTOBOARD:

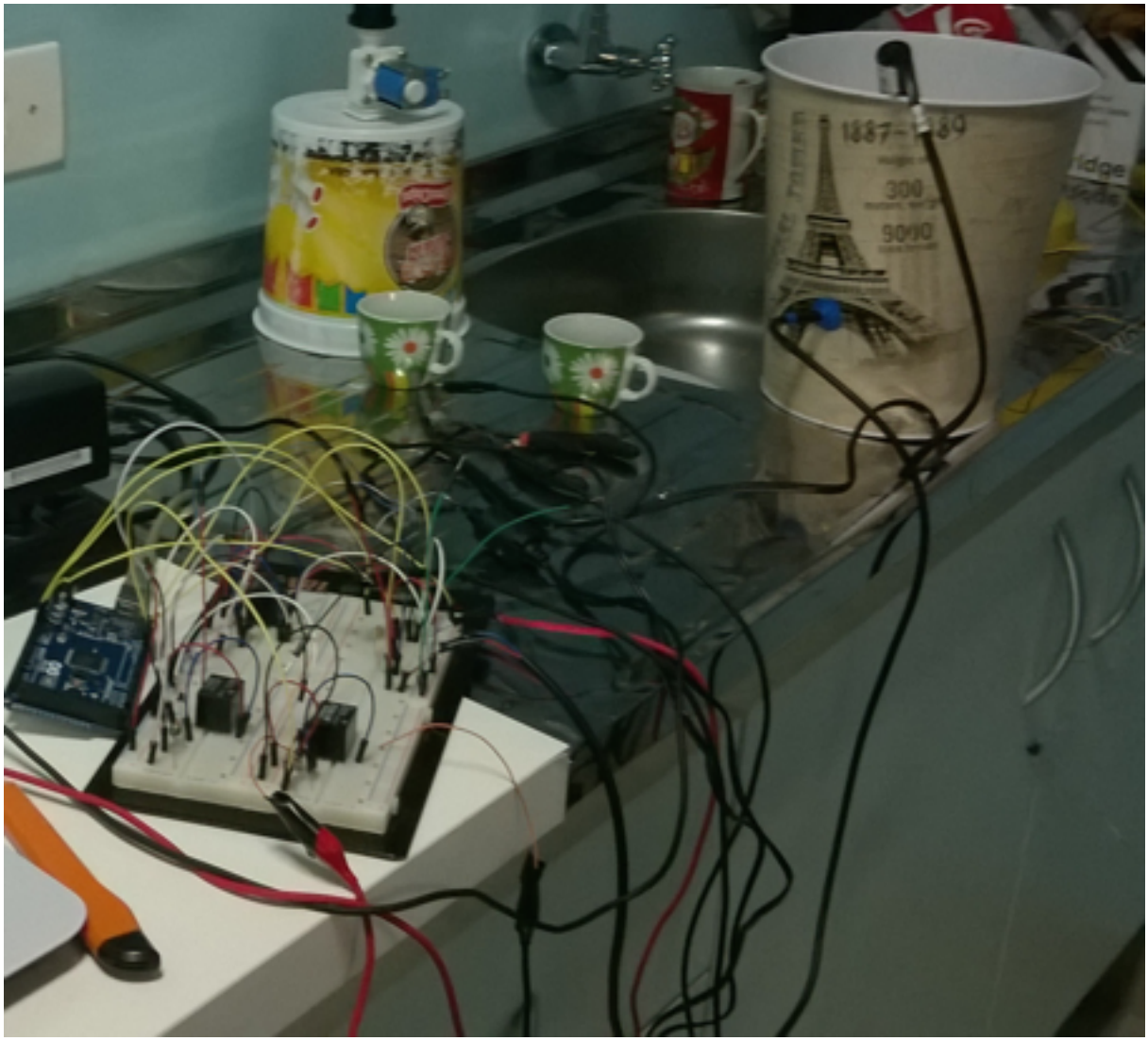


Imagem 13