CENTRO DE INSTRUÇÃO
ALMIRANTE GRAÇA ARANHA - CIAGA
ESCOLA DE FORMAÇÃO DE OFICIAIS DA
MARINHA MERCANTE – EFOMM

IMPLEMENTAÇÃO DE UM SERVIDOR WEB MICROCONTROLADO PARA AUTOMAÇÃO REMOTA DE PROCESSOS.

Por: César Luís ARNHOLD

Orientador
Prof. André Mourilhe

Rio de Janeiro
2011

CENTRO DE INSTRUÇÃO
ALMIRANTE GRAÇA ARANHA - CIAGA
ESCOLA DE FORMAÇÃO DE OFICIAIS DA
MARINHA MERCANTE – EFOMM

IMPLEMENTAÇÃO DE UM SERVIDOR WEB MICROCONTROLADO PARA AUTOMAÇÃO REMOTA DE PROCESSOS.

Apresentação de monografia ao Centro de Instrução Almirante Graça Aranha como condição prévia para a conclusão do Curso de Bacharel em Ciências Náuticas do Curso de Formação de Oficiais de Máquinas (FOMQ) da Marinha Mercante.

Por: César Luís **Arnhold**

CENTRO DE INSTRUÇÃO

ALMIRANTE GRAÇA ARANHA - CIAGA

ESCOLA DE FORMAÇÃO DE OFICIAIS DA

MARINHA MERCANTE – EFOMM

AVALIAÇÃO

PROFESSOR ORIENTADOR (trabalho escrito):_____

NOTA - _____

BANCA EXAMINADORA (apresentação oral):

_____

Prof. (nome e titulação)

_____

Prof. (nome e titulação)

_____

Prof. (nome e titulação)

_____

NOTA: _____

DATA: _____

NOTA FINAL: _____

AGRADECIMENTOS

Agradeço a todos os professores, amigos e família que de alguma forma me apoiaram na elaboração desta monografia.

DEDICATÓRIA

Dedico esta monografia a meus pais Ivo e Clarice que sempre acreditaram nos meus sonhos e se sacrificaram para que eu chegasse até aqui.

RESUMO

Este projeto tem como objetivo desenvolver um ambiente WEB centralizado de informações sobre dispositivos microcontrolados de baixa potência e baixo consumo. O projeto será demonstrado em dois ambientes distintos, o primeiro simulando um ambiente residencial/industrial comum, no qual haverá um dispositivo mestre que terá por objetivo a manutenção e controle dos dispositivos atuadores e dos demais dispositivos escravos. Com isso a intenção será de gerenciar os ambientes controlando seus dispositivos remotamente.

O principal propósito do projeto é o de fazer o cliente interagir com o ambiente utilizando os dispositivos atuadores via web, dentre alguns exemplos possíveis seriam eles: ligar luzes, bombas, acionar equipamentos, entre outras funcionalidades que estejam ligadas na rede elétrica ou barramento principal e implementadas com atuadores e sensores.

Palavras-chave: WEB, microcontrolado, remotamente, atuadores, sensores.

ABSTRACT

This project aims to develop a centralized web environment of information on micro-controlled devices with low power and low consumption. The project will be shown in two different environments, the first simulating a common residential/industrial environment, where there will be a master device that will aim to maintain and control the actuator device and the other slave devices. The intention will be to manage the devices remotely by controlling their environments.

The main purpose of the project is to make the customer interact with the environment using actuators devices via the Web, among them some possible examples would be: turn on lights, pumps, fire equipment, among other features that are connected to the power grid and implemented with actuators and sensors.

Keywords: WEB microcontroller remotely, actuators, sensors.

ÍNDICE DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

O papel da automação vai além do aumento da eficiência e qualidade de vida no âmbito residencial e profissional, mas também pode ser focada a ambientes corporativos, ela está intimamente ligada ao uso eficaz da energia, sendo importante para a economia e o meio ambiente.

O objetivo do presente projeto visa disponibilizar a possibilidade de interagir com as novas tecnologias de automação que poderão ser utilizadas tanto para automação da praça de máquinas de navios como também em residências, empresas, dentre outros. Em todos os casos possibilitará ao cliente atuar com acesso total nos dispositivos elétricos que estiverem com os atuadores e sensores ligados a central, através de uma conexão remota (via WEB), também poderão automatizar funções rotineiras e cotidianas, pois terá acesso a sensores e atuadores através do ambiente WEB, provendo serviços como segurança remota. Isso é obtido através de um projeto único que envolve infra-estrutura, dispositivos e software de controle, cuja meta é garantir ao usuário a possibilidade de acesso e de controle do ambiente automatizado, dentro ou fora da mesma, via web.

A duas figuras seguintes ilustram as possibilidades possíveis para esse tipo de automação.

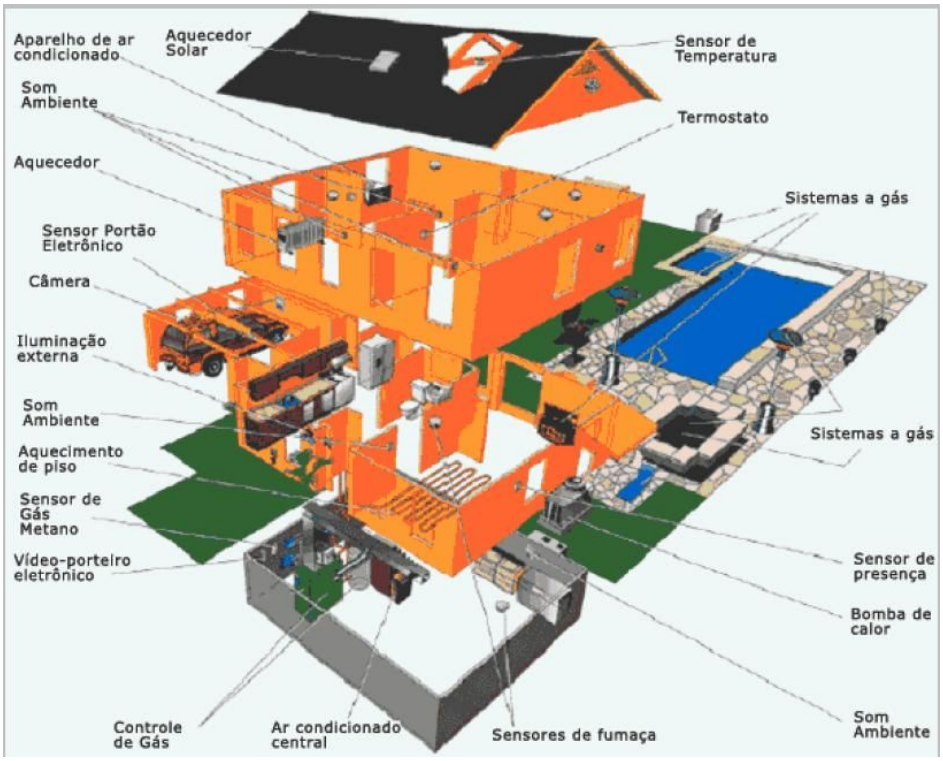

Figura 1 - Possíveis formas de automação residencial.

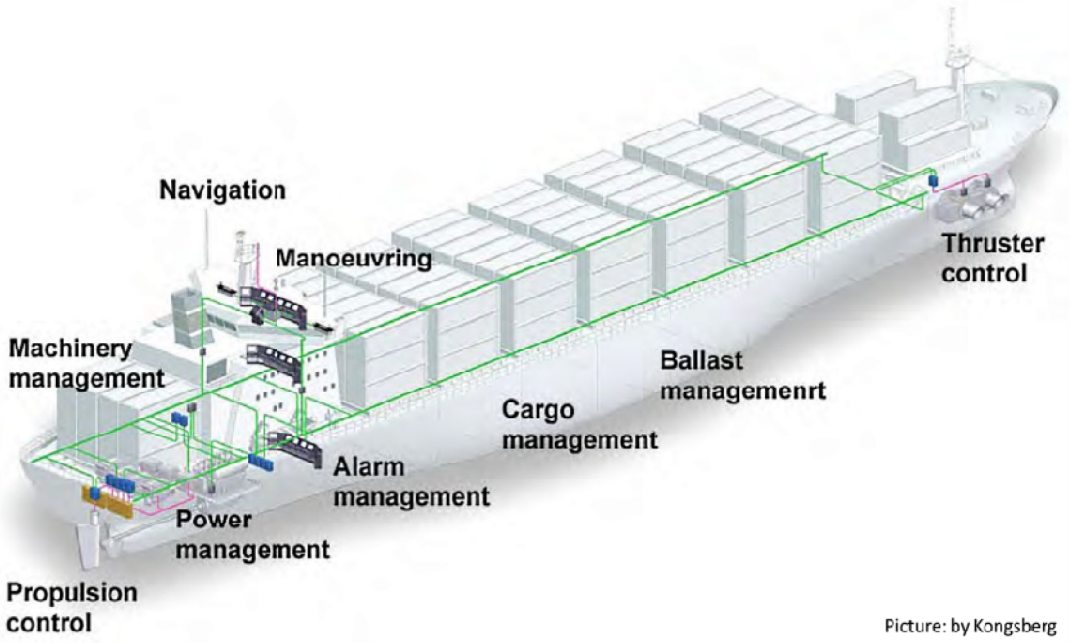

Figura 2 - Possíveis formas de automação em navios.

Para elaboração do presente projeto foi implementado um módulo central de controle, realizado numa plataforma de desenvolvimento *opensource*, o Arduino. Ainda encontra-se em desenvolvimento um segundo módulo que fará comunicação por RF (434MHz) com o módulo central. A idéia é acrescentar mais sensores e uma Interface Homem Máquina (IHM) que possua maior mobilidade e possa ser instalada em locais de difícil acesso por cabeamento. O servidor web faz todo o gerenciamento das requisições vindas da rede local e aciona os atuadores como também envia os dados de todos os sensores.

Destaca-se também a utilização de diversos sensores e atuadores no circuito final. Para a aquisição, montagem e testes um dos principais fatores na escolha dos componentes foi a relação entre custo e benefício. Os sensores de temperatura e pressão são largamente utilizados a bordo para o monitoramento de óleo combustível, lubrificante, sistema hidrofórico, alarmes, ar condicionado e numa infinidade de aplicações. No presente projeto, utiliza-se um sensor de pressão para a medição precisa do nível de um tanque e uma válvula solenóide e bomba para o seu controle. Sensores de umidade podem ser utilizados em câmaras frigoríficas e controle da qualidade do ar condicionado entre outros. O monitoramento da tensão e corrente do barramento principal é de fundamental importância para o correto funcionamento de todas as máquinas, e isso também foi obtido com os sensores adequados. Finalmente, o sensor de luminosidade pode ser utilizado para ajustar automaticamente a intensidade luminosa de determinados ambientes e detectar se a caldeira de bordo encontra-se acionada.

2 DETALHAMENTO DO PROJETO

A seguir, encontra-se o Diagrama de Blocos (Figura 2) geral do projeto, contendo os dois módulos a serem desenvolvidos, além de uma descrição detalhada do funcionamento tanto de hardware quanto de software de cada módulo utilizado no projeto.

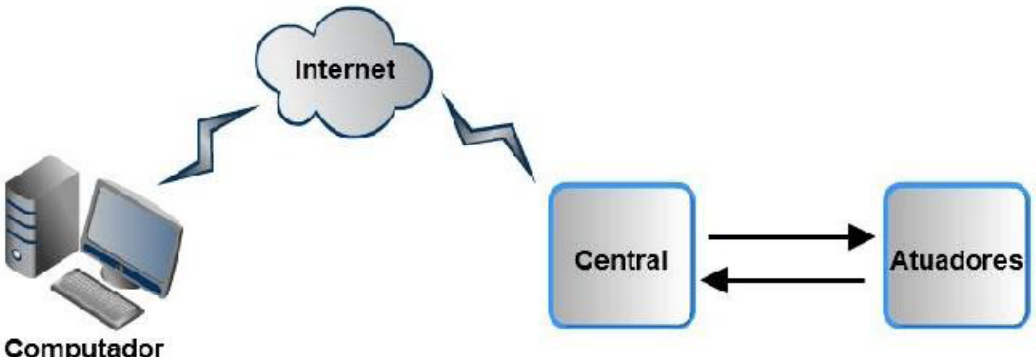

Figura 3 - Diagrama de blocos geral do projeto.

2.1 Central

Central é o módulo responsável pelo gerenciamento e encaminhamento dos dados recebidos a partir de outros módulos. A Central é composta por três dispositivos de hardware sendo eles: Arduino Mega (ATmega1280), Ethernet Shield e futuramente o módulo RFM22, os quais serão detalhados a seguir.

2.1.1 Arduino

Arduino é um computador físico baseado numa simples plataforma de hardware livre, projetada com um microcontrolador (µC) da ATMEL©, com suporte de entrada/saída e uma linguagem de programação padrão, que é essencialmente C/C++.

Uma placa Arduino é composta por um controlador, algumas linhas de E/S digital e analógica, além de uma interface serial ou USB utilizada para a programação e interação em tempo real.

Sua placa consiste em um microcontrolador da família AVR da ATMEL©, de 8 bits, com componentes complementares para facilitar a programação e incorporação para outros circuitos. Um importante aspecto é a maneira padrão que os conectores são expostos, permitindo a CPU ser interligada a outros módulos expansivos, conhecidos como *shields*.

A descrição abaixo resume algumas características do Arduino Mega.

- Microcontroller …….....ATmega1280
- Operating Voltage ……...5V
- Input Voltage (recommended) ……7-12 V
- Input Voltage (limits) ……..6-20 V
- Digital I/O Pins ……54 (of which 14 provide PWM output)
- Analog Input Pins ……16
- DC Current per I/O Pin ……40 mA
- DC Current for 3.3V Pin …….50 mA
- Flash Memory ……..128 KB (of which 4 KB used by boot loader)
- SRAM ….....8 KB
- EEPROM .....4 KB
- Clock Speed .....16 MHz

O Arduino vem gravado um *bootloader* que permite que você faça o *upload* do novo código para ele sem a utilização de um programador de hardware externo. Ele se comunica utilizando o protocolo STK500. O Arduino IDE é o compilador utilizado para o upload do novo código.

Os projetos e esquemas de hardwares são distribuídos sob a licença Creative Commons Attribution Share-Alike 2.5, e estão disponíveis em sua página oficial. Arquivos de layout e produção para algumas versões também estão hospedadas. O código fonte para o IDE e a biblioteca de funções da placa são disponibilizadas sob a licença GPLv2 e hospedadas pelo projeto Google Code. Na figura 4 encontra-se detalhado o módulo Arduino.

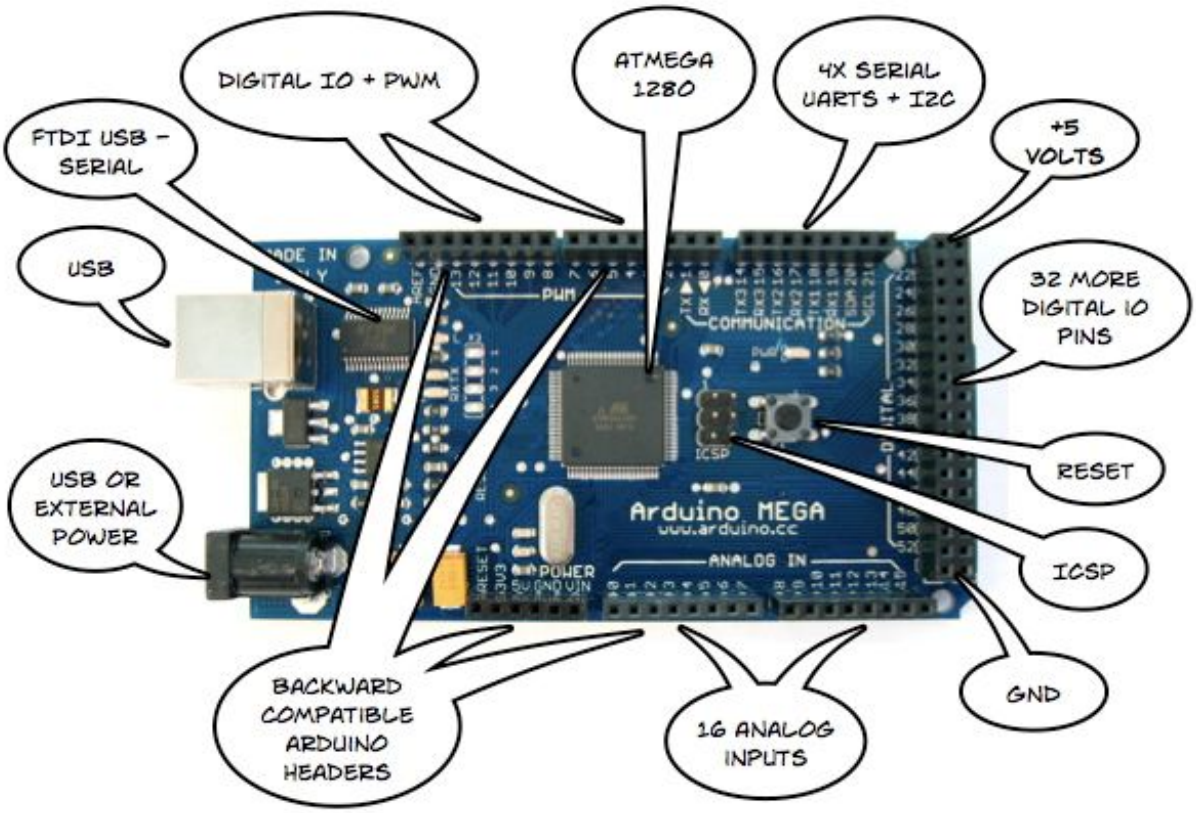

Figura 4 – Detalhamento do Arduino Mega.

2.1.2 Ethernet Shield

O Arduino Ethernet Shield V1.1 é compatível com os padrões das placas Arduino. O ENC28J60 é um controlador Ethernet *stand-alone* com um protocolo de comunicação por Serial Peripheral Interface (SPI). Ele é projetado para servir como uma rede Ethernet com interface para qualquer controlador equipado com SPI. O ENC28J60 satisfaz todas as especificações IEEE 802.3, e possui duas camadas sendo uma a camada PHY (Physical Layer) e a outra MAC (Medium Access Layer) e uma tomada RJ45 padrão.

Na figura 5 encontram-se os parâmetros e valores do ENC28J60. Na figura 6 encontra-se o módulo Ethernet Shield.

Parameter Name	Value
MAC	Yes
PHY	Yes
TX/RX RAM Buffer(bytes)	8192
Interrupt Pin	1
LEDs	2
Op. Voltage (V)	3.3
Temp. Range Min. (°C)	-40
Temp. Range Max. (°C)	85
Max. Speed (MHz)	25
Interface	SPI
Pre-Programmed MAC Address	No
Security Engines	No
Standalone Ethernet Controller	10Base-T

Figura 5 – Pârametros e valores do ENC28J60.

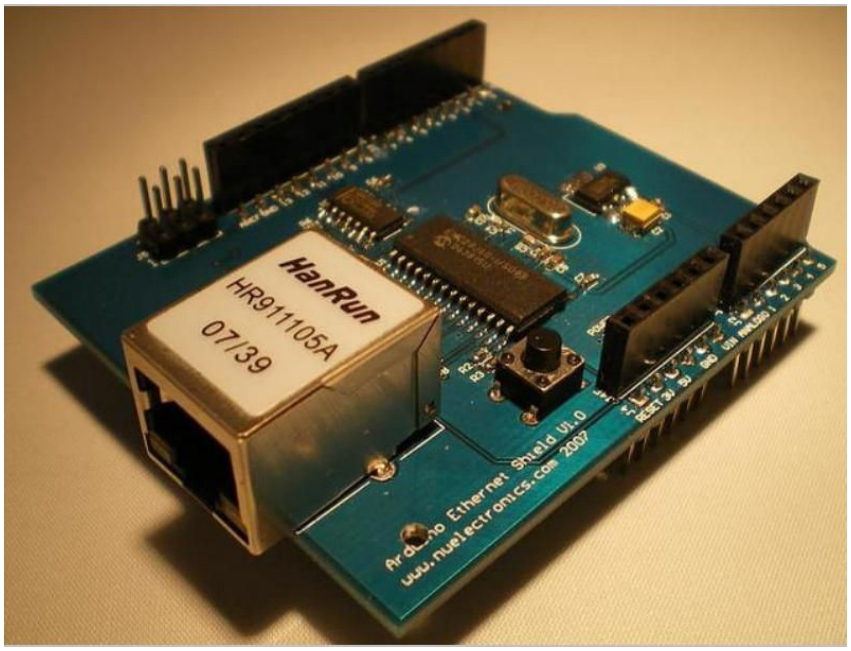

Figura 6 - Ethernet Shield

O software Ethernet Shield é no formato da biblioteca Arduino. A biblioteca é implementada com base em arduino.cc que contém uma pilha TCP/IP open-source para ATMEGA88 e ENC28J60. Na figura 7 se encontra o diagrama de blocos do ENC28J60.

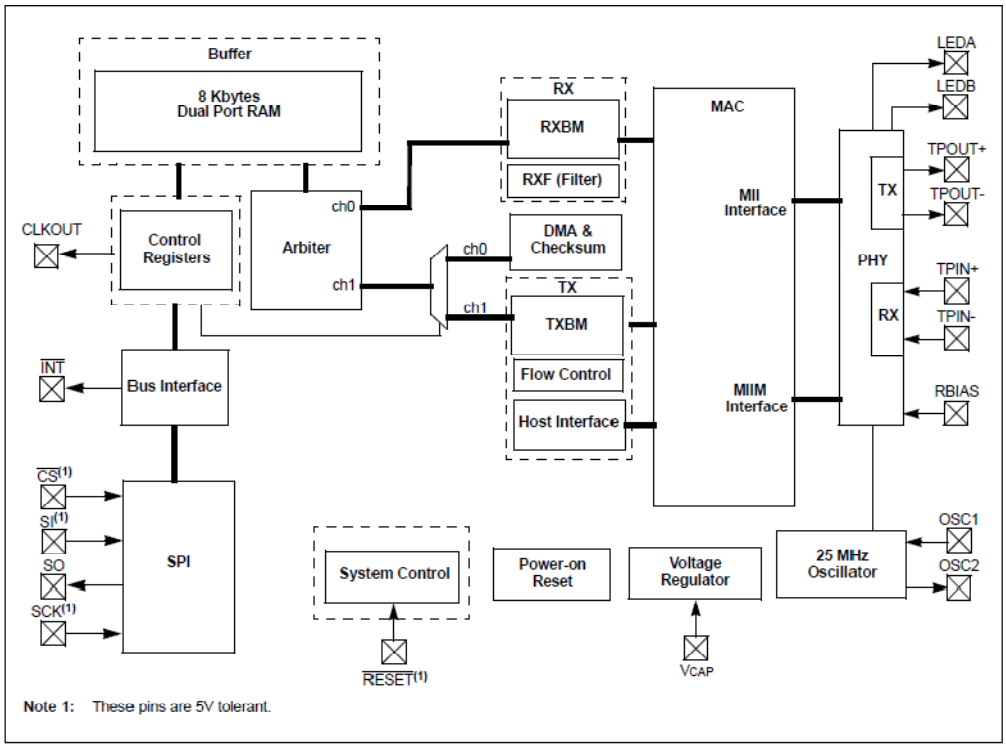

Figura 7 - Diagrama em blocos do ENC28J60.

O TCP/IP é um protocolo padrão para estabelecer uma conexão. Para isso, uma série de pacotes deve ser trocada entre os dois lados para estabelecer a conexão e assim os pacotes de dados podem ser trocados. As especificações do protocolo TCP/IP não serão discutidas no presente trabalho por fugirem do escopo.

Para o Arduino com ATMEGA168, um microcontrolador AVR 8-bit com 1K SRAM, é impossível implementar uma pilha TCP completa. Além disso, a página web para microcontroladores de 8 bits, é normalmente usada para controlar uma lâmpada ou ler um sensor de temperatura. Portanto, em vez de implementar um protocolo TCP completo, um único pacote de dados protocolo TCP é utilizado. Todo conteúdo web, incluindo todas as tags HTML, deve estar em um pacote. O comprimento do pacote é limitado pelo tamanho da SRAM, atualmente 1600 bytes de memória RAM são usados para buffer de pacotes de rede. Isso é suficiente para páginas simples como a que iremos implementar.

Como mencionado anteriormente um único pacote de dados do protocolo TCP é utilizado, o WEB Server está implementado diretamente na memória do ATMEGA1280, por questões de espaço em memória o tamanho das páginas serão bem reduzidos.

2.1.3 Dispositivo RF

O módulo RFM22 é altamente versátil, é uma solução de rádio de baixa potência que é relativamente fácil de configurar e integrar em qualquer projeto, que exija uma comunicação sem fio RF. Sua comunicação com o µC é feita usando o protocolo SPI.

Na fase atual de desenvolvimento, o módulo RFM22 ainda não foi implementado fisicamente no hardware. Neste momento, pode-se considerar então como uma extensão ainda em testes e que será implementada futuramente para a comunicação com outros sensores, atuadores e interfaces localizadas em locais que seria de difícil acesso usando o método convencional com fios e conexões.

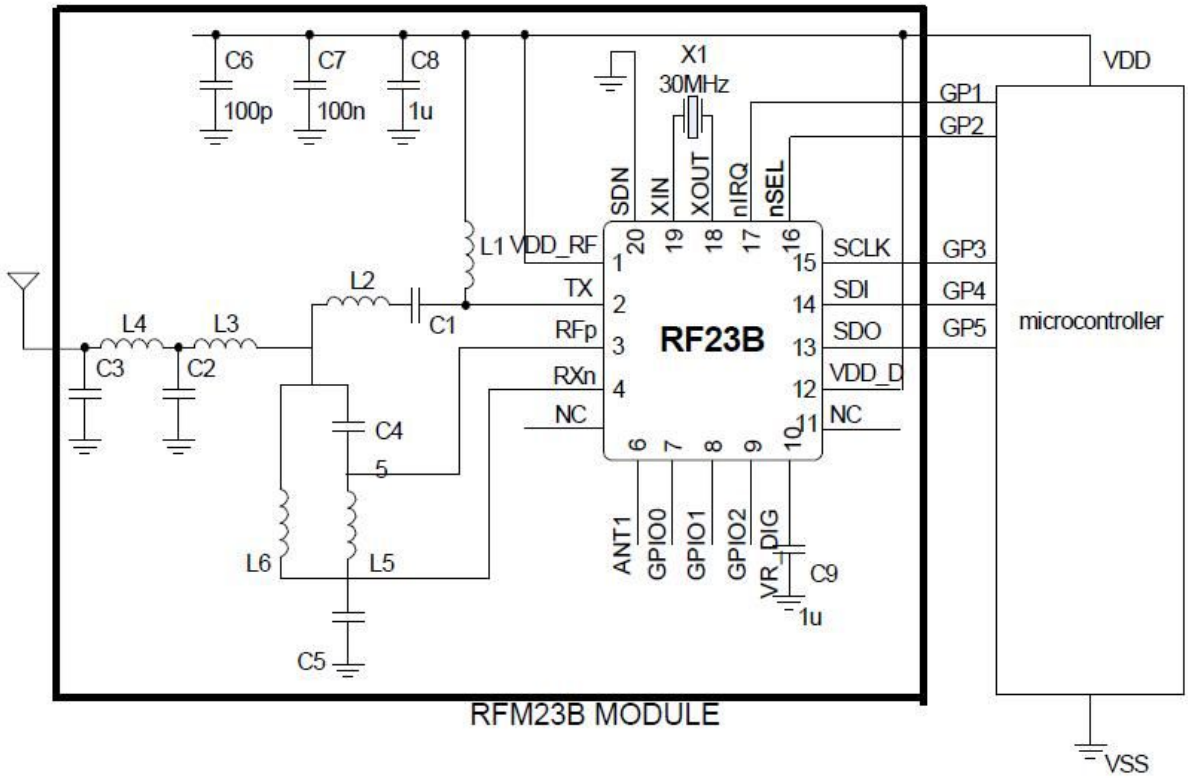

Figura 8 - Diagrama do módulo RFM22/23

2.1.4 Implementação

A central foi desenvolvida basicamente em duas partes: servidor web e tratamento das informações provenientes dos sensores e atuadores. Todas essas implementações foram desenvolvidas em linguagem C++ e HTML/CSS.

Resumindo, a central é responsável pelo tratamento das requisições web e também pela interpretação das respostas vindas dos sensores e atuadores, para isso ela é composta por um módulo Arduino, um Ethernet Shield e futuramente um dispositivo RF.

2.2 Hardware - Sensores e Atuadores

A seguir, serão detalhadas as especificações e características dos componentes que fazem a comunicação com o mundo físico, ou seja, os sensores e atuadores.

2.2.1 Sensores

Sensores são dispositivos usados para detectar, medir ou gravar fenômenos físicos tais como calor, radiação etc, e que responde transmitindo informação, iniciando mudanças ou operando controles.

A seguir estão listadas as características de todos os sensores utilizados no projeto. Para maiores informações, o *datasheet* do respectivo componente deverá ser consultado.

2.2.1.1 Sensor digital de temperatura – DS18S20

Este sensor utiliza o protocolo 1-Wire para comunicação com o µC. Como cada sensor possui seu próprio endereço, é possível a conexão de mais de um sensor na mesma linha de dados.

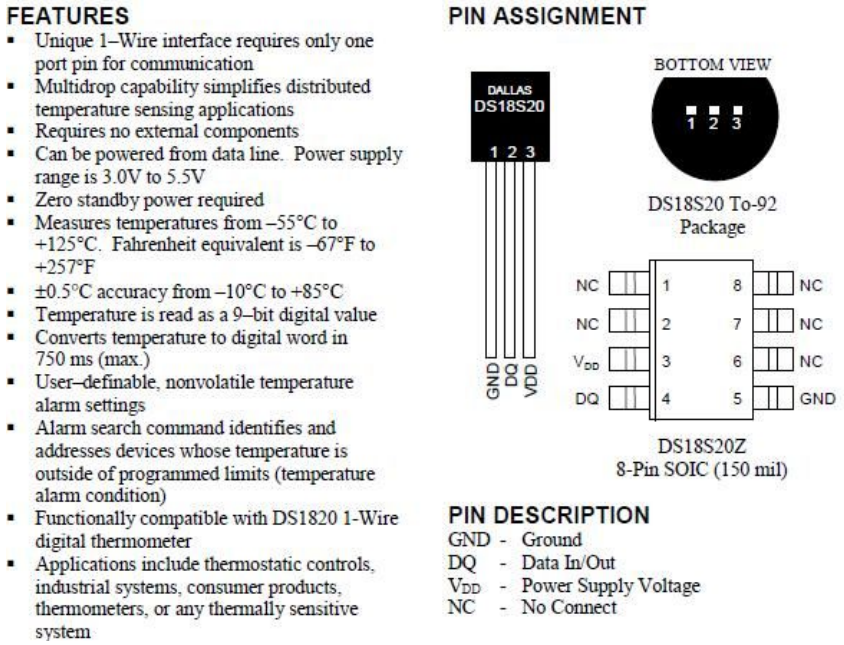

Figura 9 - Característica técnicas do sensor DS18S20

2.2.1.2 Sensor digital de Umidade e Temperatura – DHT22

Este sensor utiliza um protocolo proprietário (específico) para a comunicação com o µC, não sendo compatível com o 1-Wire anteriormente citado.

Model	DHT22
Power supply	3.3-6V DC
Output signal	digital signal via single-bus
Sensing element	Polymer capacitor
Operating range	humidity 0-100%RH; temperature -40~80Celsius
Accuracy	humidity +-2%RH(Max +-5%RH); temperature <+-0.5Celsius
Resolution or sensitivity	humidity 0.1%RH; temperature 0.1Celsius
Repeatability	humidity +-1%RH; temperature +-0.2Celsius
Humidity hysteresis	+-0.3%RH
Long-term Stability	+-0.5%RH/year
Sensing period	Average: 2s
Interchangeability	fully interchangeable
Dimensions	small size 14*18*5.5mm; big size 22*28*5mm

Figura 10 - Características técnicas do sensor DHT22

2.2.1.3 Sensor analógico de tensão

Para o sensoriamento da tensão de alimentação do circuito, foi utilizado um divisor resitivo de tensão e, sua saída conectada à entrada analógica do µC (arduino).

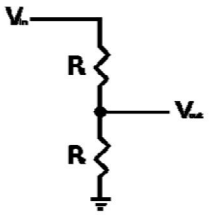

Figura 11 - Divisor resitivo para monitoramento da tensão de alimentação.

2.2.1.4 Sensor analógico de corrente de efeito Hall

Para o sensoriamente da corrente total do circuito, foi utilizado um sensor de efeito Hall.

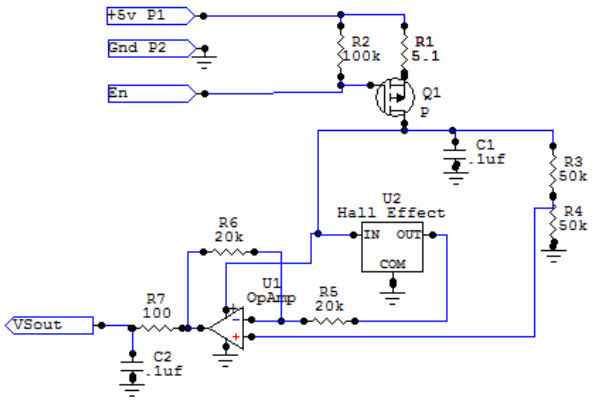

Figura 12 - Diagrama esquemático do sensor de corrente

Operating Voltage	4.5V to 5.5v
Power Supply Current	5 mA typical
Over Current Response Time	~10us
Bandwidth	50khz
DC Offset (with no applied field current)	75mv(MAX) 25mV(typical)
Sensitivity	23 mV/Amp typical
Output Center Voltage	½ Supply - ~25mV
Inductance	~30nh

Figura 13 - Características técnicas do sensor de corrente de efeito Hall.

2.2.1.5 Sensor analógico de luminosidade – LDR

LDR (do inglês Light Dependent Resistor) é um tipo de resistor cuja resistência varia conforme a intensidade de radiação eletromagnética do espectro visível que incide sobre ele. Sua resistência diminui quando a luz é muito alta, e quando a luz é baixa, a resistência no LDR aumenta.

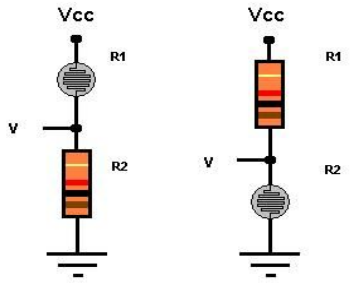

Figura 14 - Divisor resitivo para o LDR.

Figura 15 - Foto-resistor.

2.2.1.6 Sensor analógico de Pressão – MPX2010

Para monitoramento do nível do tanque, foi utilizado um sensor linear de pressão conectado diretamente na tubulação de saída do tanque.

OPERATING CHARACTERISTICS (V_S = 10 Vdc, T_A = 25°C unless otherwise noted, P1 > P2)

Characteristic	Symbol	Min	Typ	Max	Unit
Pressure Range(1)	P_{OP}	0	—	10	kPa
Supply Voltage(2)	V_S	—	10	16	Vdc
Supply Current	I_o	—	6.0	—	mAdc
Full Scale Span(3)	V_{FSS}	24	25	26	mV
Offset(4)	V_{off}	–1.0	—	1.0	mV
Sensitivity	ΔV/ΔP	—	2.5	—	mV/kPa
Linearity(5)	—	–1.0	—	1.0	$\%V_{FSS}$
Pressure Hysteresis(5) (0 to 10 kPa)	—	—	±0.1	—	$\%V_{FSS}$
Temperature Hysteresis(5) (–40°C to +125°C)	—	—	±0.5	—	$\%V_{FSS}$
Temperature Effect on Full Scale Span(5)	TCV_{FSS}	–1.0	—	1.0	$\%V_{FSS}$
Temperature Effect on Offset(5)	TCV_{off}	–1.0	—	1.0	mV
Input Impedance	Z_{in}	1000	—	2550	Ω
Output Impedance	Z_{out}	1400	—	3000	Ω
Response Time(6) (10% to 90%)	t_R	—	1.0	—	ms
Warm–Up	—	—	20	—	ms
Offset Stability(7)	—	—	±0.5	—	$\%V_{FSS}$

Figura 16 - Características técnicas do sensor de pressão.

O sinal proveniente do sensor possui uma amplitude muito baixa e assim não é possível conectar diretamente na entrada analógica do µC para obter a precisão desejada. Pelos cálculos e testes efetuados, uma variação de nível de 90 cm no tanque produz uma variação de aproximadamente 11 mV na saída do sensor. Como o conversor analógico/digital (AD) do arduino possui uma resolução de 10 bits e a referência analógica utilizada foi de 5 V, a máxima resolução obtida pelo µC é de aproximadamente 5 mV (5/1024 = 0,004882). Assim, fez-se necessária a construção de um módulo de amplificação desse sinal. Para tal, foi utilizado o amplificador operacional LM324 no modo de amplificador diferencial de tensão com alta impedância de entrada. Para satisfazer todas as características físicas e de software, o ganho total foi calculado em aproximadamente 180.

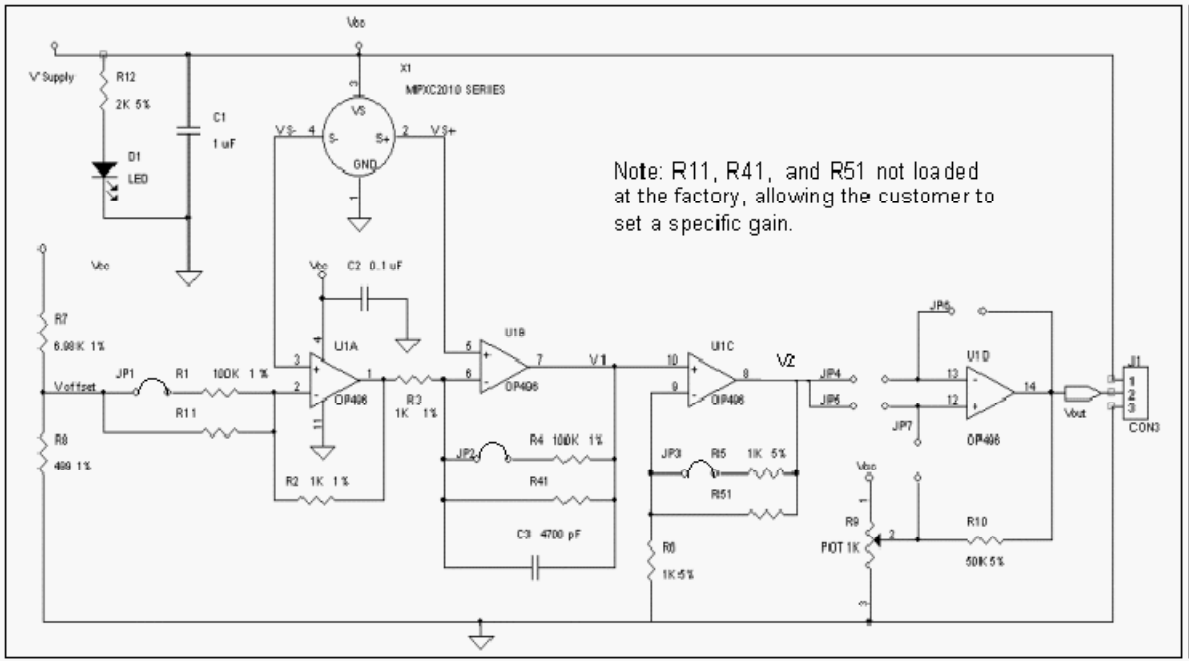

Figura 17 - Circuito de amplificação do sinal do sensor de nível.

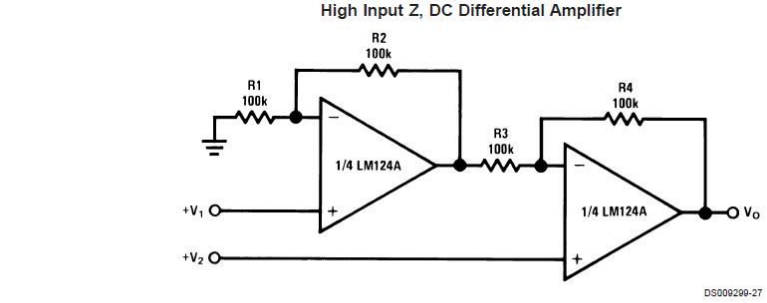

Figura 18 - LM324 como amplificador diferencial.

2.2.2 Atuadores

Atuador é um elemento que produz movimento, atendendo a comandos que podem ser manuais ou automáticos, ou seja, qualquer elemento que realize um comando recebido de outro dispositivo, com base em uma entrada ou critério a ser seguido.

Foram utilizados os seguintes atuadores.

2.2.2.1 Bomba de água

Para minimizar os custos nessa fase do projeto, foi utilizada uma bomba de água convencional de um limpador de pára-brisa de carro (12 V). A vazão estimada é de 6L/min.

Figura 19 - Bomba de água de limpador de pára-brisa.

2.2.2.2 Válvula Solenóide

Válvula solenóide convencional, muito utilizada em máquinas de lavar roupa. A bobina original foi trocada por uma de 12 V.

Figura 20 - Válvula solenóide.

2.2.2.3 Relés

Para acionamento da bomba de água e da válvula solenóide, foram utilizados dois relés, conforme esquema abaixo.

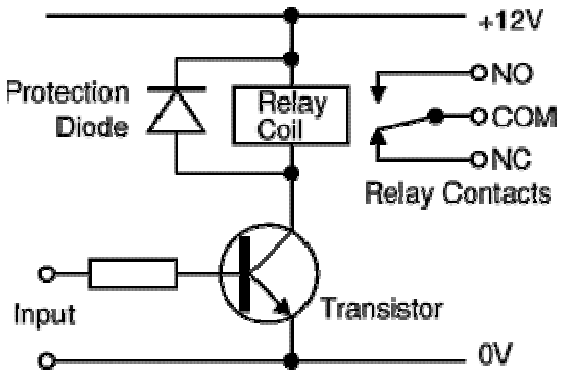

Figura 21 - Diagrama esquemático de acionamento dos relés.

2.2.2.4 Saídas diversas

A quantidade de dispositivos que podemos controlar depende exclusivamente da quantidade de pinos de entradas e saídas disponíveis no µC. Assim, esse número pode ser relativamente grande. Para efeito prático, foram acrescentadas mais quatro saídas, além da bomba e da válvula solenóide e, nestas saídas foram conectados alguns LEDs.

2.2.3 Implementação

A implementação do circuito de acionamento e do amplificador operacional foi feita através da criação de placas de circuito impresso onde estas têm a função de fazer a devida conexão com o circuito externo de maior potência, com o arduino que trabalha com baixa potência. Esse mecanismo foi desenvolvido para uma questão de isolamento e melhor acabamento dos dispositivos.

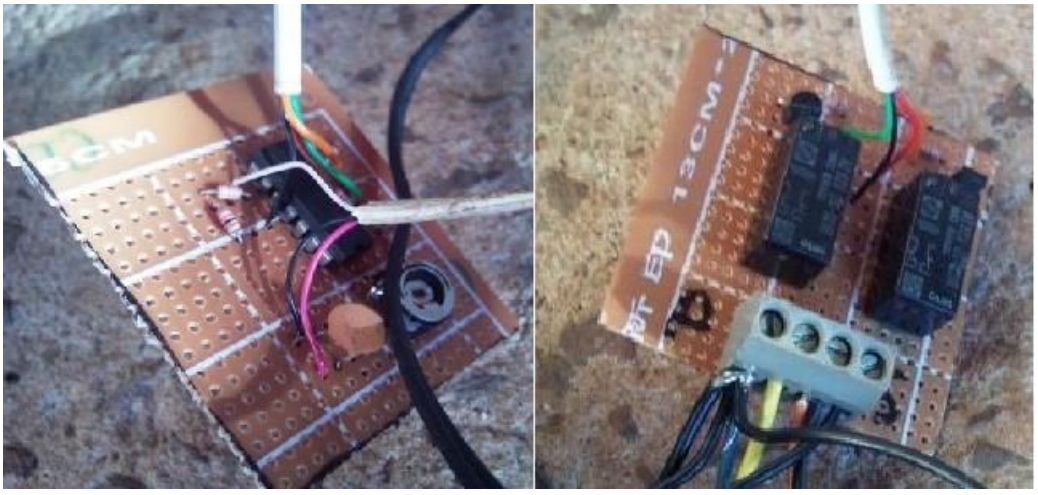

Figura 22 - Placas de circuito impresso com Amplificador Operacional e relés.

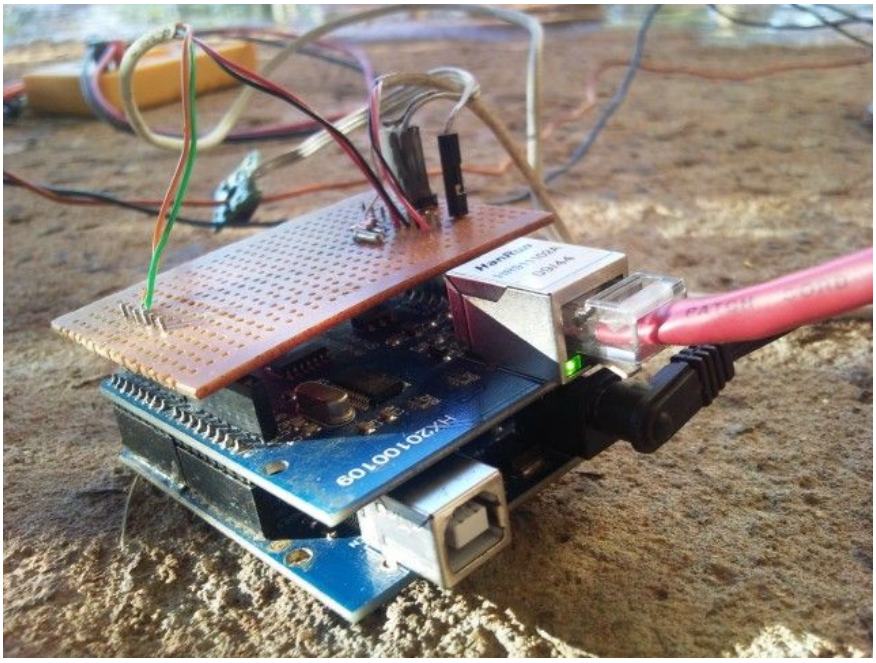

Figura 23 - Placa de circuito impresso para acoplamento no arduino.

2.3 Desenvolvimento do Software

O Arduino IDE é uma aplicação multiplataforma escrita em Java na qual é derivada dos projetos Processing e Wiring. É esquematizado para introduzir a programação a hobbistas e a pessoas não familiarizadas com o desenvolvimento de software. Inclui um editor de código com recursos de realce de sintaxe, parênteses correspondentes e identação automática, sendo capaz de compilar e carregar programas para a placa com um único clique. Com isso não há a necessidade de editar *makefiles* ou rodar programas em ambientes de linha de comando.

Tendo uma biblioteca chamada "Wiring", ele possui a capacidade de programar em C/C++. Isto permite criar com facilidade muitas operações de entrada e saída, tendo que definir apenas duas funções para fazer um programa funcional:

`setup()` – Inserida no inicio, na qual pode ser usada para inicializar configuração, e
`loop()` – Chamada para repetir um bloco de comandos ou esperar até que seja desligada.

Para incrementar todas as funções deste projeto, também foram utilizadas bibliotecas de código aberto para facilitar a programação e comunicação com todas as partes do hardware. O código fonte completo está no final desse trabalho, em anexo.

3 TESTES

3.1 Testes do modulo central

O teste com a central engloba todo o funcionamento da placa Arduino juntamente com o Ethernet Shield. A programação de acionamento e gerenciamento foi feita em linguagem C, pois o processador do Arduino suporta este tipo de linguagem, e para a interface com o usuário foi desenvolvida uma pagina em HTML esta onde o cliente pode encontrar seus dispositivos de atuação online.

Os testes realizados foram os seguintes com suas descrições e suas respostas.

3.1.1 Teste de compilação e gravação no microcontrolador (arduino)

Configurou-se uma COM virtual para que pudesse ser feita a programação no µC ATmega1280, utilizando o compilador próprio do hardware, tendo como satisfatório o resultado obtido. A figura abaixo mostra que a compilação para o µC foi executada com sucesso.

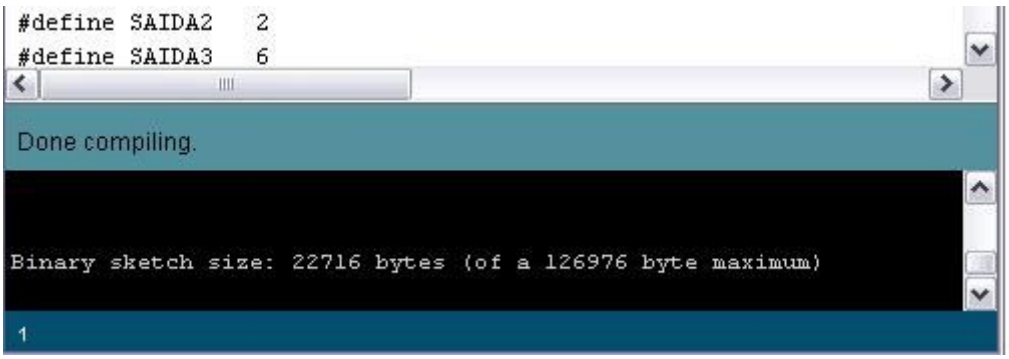

Figura 24 - Compilação e gravação no microcontrolador.

3.1.2 Teste de validação do servidor

Após a codificação do servidor web, em linguagem C++/HTML, a maneira encontrada para validá-lo foi utilizar o prompt de comando do Windows, e nele utilizou-se o comando “ping”, com isso verifica-se a resposta obtida pelo servidor. A figura abaixo é o trecho de código implementado onde pode-se verificar o *MAC address* e o IP que foram setados no dispositivo para que ele pertença a uma rede local.

```
// ethernet interface mac address - must be unique on your network
static byte mymac[] = { 0x74,0x69,0x69,0x2D,0x30,0x31 };
static byte myip[] = { 192,168,0,25 };

static BufferFiller bfill;  // used as cursor while filling the buffer
byte Ethernet::buffer[1600];   // tcp/ip send and receive buffer
```

Figura 25 - Parte do código/MAC e IP

3.1.3 Teste da página WEB

O teste que foi aplicado na página WEB foi feito visando uma possível falha em todos os links disponíveis para o controle e monitoramento. Procurou-se testar a resposta da página em cada um deles. Foram testados também os botões disponíveis na página web para confirmar a eficácia da resposta e a confiança na informação enviada. Em ambos os testes a resposta obtida foi satisfatória.

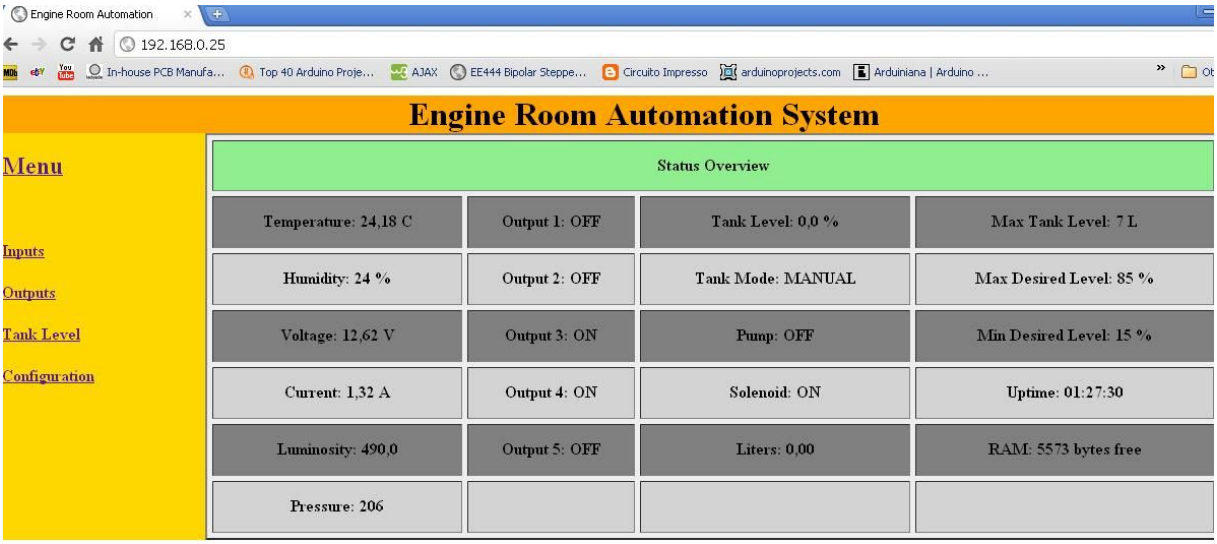

Figura 26 - Página Web Principal

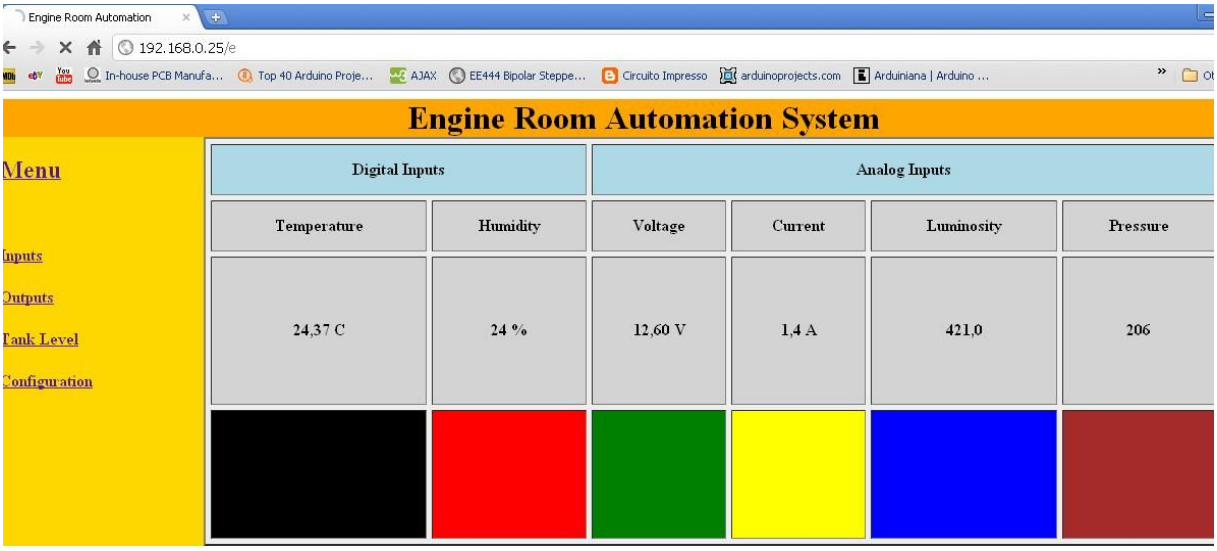

Figura 27 - Página Web - Entradas

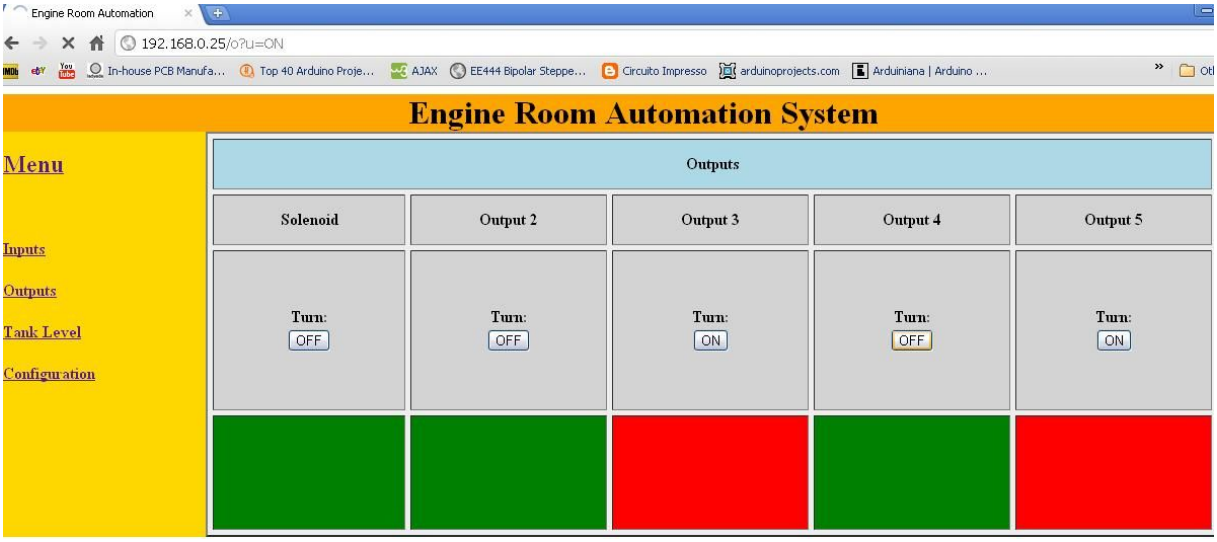

Figura 28 - Página Web - Saídas

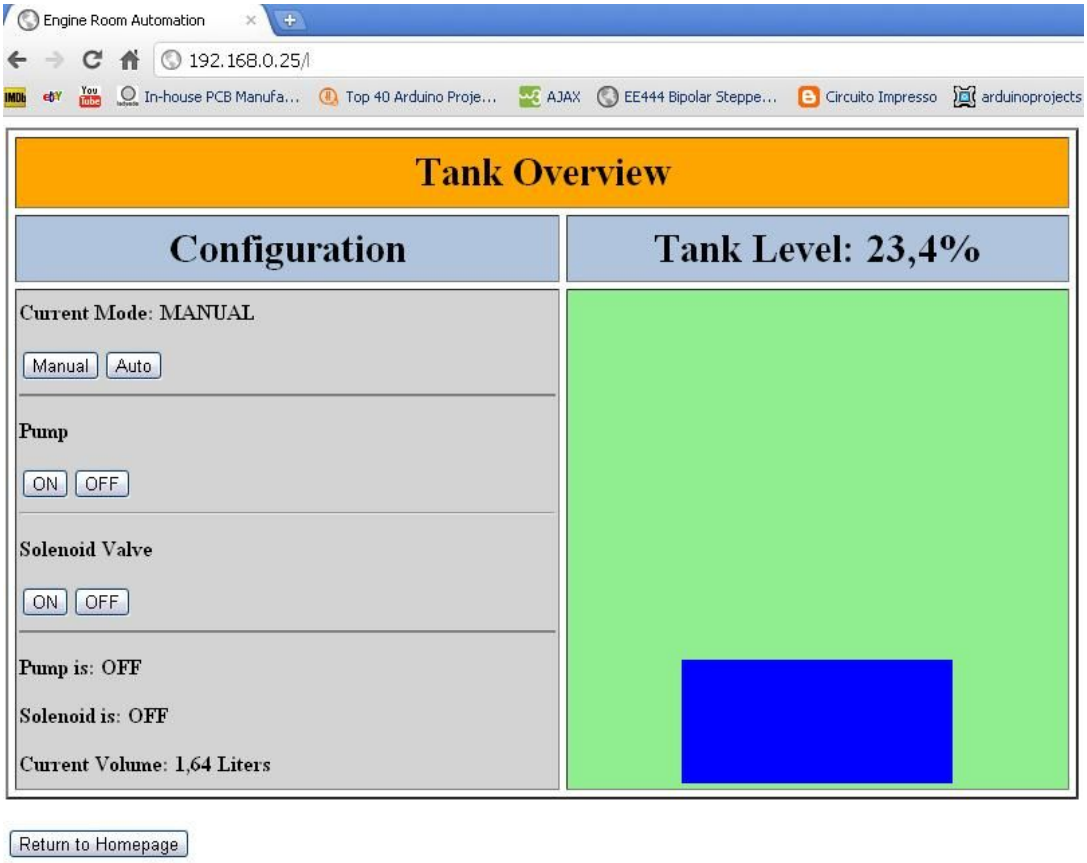

Figura 29 - Página Web - Tanque no modo Manual

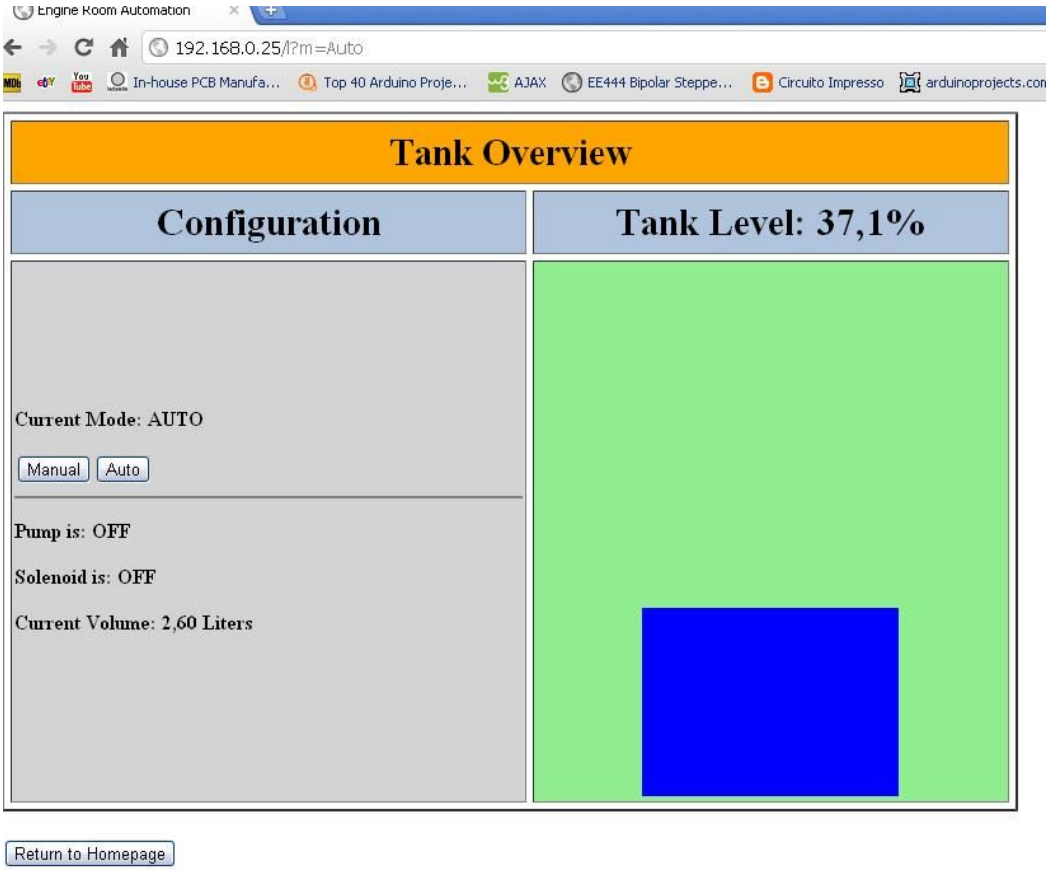

Figura 30 - Página Web - Tanque no modo Automático

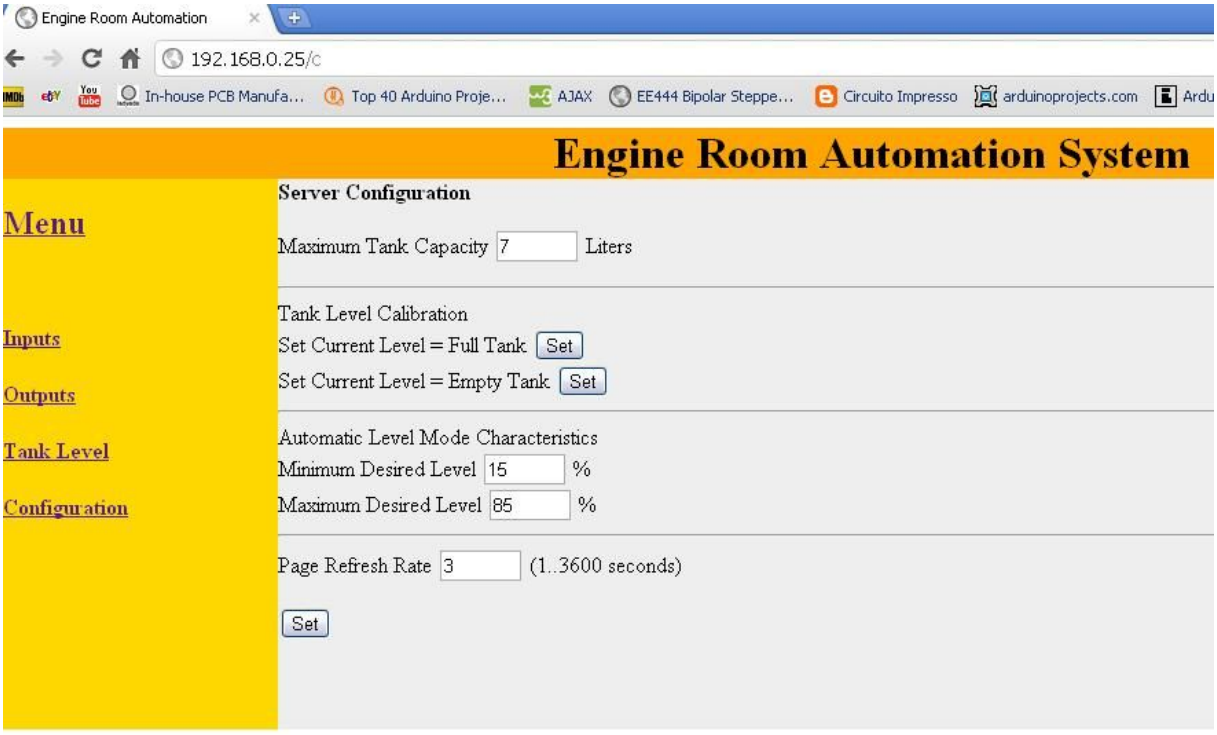

Figura 31 - Página Web - Configurações

3.1.4 Teste dos dispositivos em conjunto

O teste final da central foi feito com todos os módulos unidos, são eles: Arduino, Ethernet Shield, sensores e atuadores, e todas as codificações como pagina HTML, Servidor WEB e programação do processador ATMEGA. O mini servidor web é acessado por um computador externo dentro da rede local e com isso o cliente pode interagir nos atuadores que estavam codificados no Arduino. O resultado obtido com a integração destas foi o esperado, pois todas se comunicaram com perfeição.

Figura 32 - Projeto final montado em fase de testes.

3.2 Teste Geral do projeto

Para finalizar os testes foram integrados todos os módulos, e a partir de um computador externo, conseguiu-se simular todas as funcionalidades propostas no escopo do projeto, tais como:

- → Acessar a pagina WEB de outro computador ou celular.
- → Envio de requisições do servidor para os atuadores.
- → Processamento da informação com consistência.
- → Servidor ativo.
- → Resposta precisa dos atuadores.
- → Tempo de resposta de todos os sensores (temperatura, pressão, umidade, tensão, corrente, luminosidade).
- → Velocidade de envio das informações entre os módulos.
- → Confiabilidade dos *hardwares* utilizados.
- → Inclusão da Central em uma rede local.
- → Mobilidade dos atuadores.

4 CONCLUSAO

Este documento apresentou detalhadamente o estudo sobre automação remota via web. Demonstraram-se as etapas que foram desenvolvidas uma a uma com as devidas especificações, diagrama de blocos, circuitos e pinagens, também foram esclarecidos os tipos de testes de validação executados em cada parte do projeto, sempre visando o bom funcionamento de cada módulo e buscando um produto final de qualidade e consistência.

Após pesquisas aprofundadas em hardwares que poderiam suprir as especificações do projeto, decidiu-se adotar a plataforma Arduino, dentro desta definindo novos módulos: Central, Arduino Ethernet Shield, e para a comunicação com o módulo remoto móvel será usado o dispositivo RF (RFM22).

Essa plataforma traz grandes benefícios nessa fase do projeto, pois um grande diferencial é que seus conectores possuem uma forma padrão, permitindo o Arduino ser interligado a outros módulos expansivos, conhecidos como *Shields*.

Após a escolha do *hardware*, foi iniciada toda a parte de desenvolvimento do protótipo, onde conseguimos validar e testar grande parte dos itens propostos no documento.

Na etapa de conclusão do projeto, houve alguns imprevistos com a integração dos atuadores, que por serem indutivos geravam interferências na linha de alimentação do circuito e influenciavam na leitura de alguns sensores analógicos. Pequenas modificações nas conexões da alimentação resolveram esses detalhes.

Outro imprevisto que é muito importante citar, é a pouca memória disponível no processador da Central (Arduino), que por sua vez é responsável por criar o pacote TCP/IP a ser enviado. Esse pacote fica limitado ao tamanho da memória disponível (8KB), ou seja, há um limite de informações que podem ser enviados de uma única vez. Isto limita a quantidade de atuadores e sensores que podem ser controlados pela central, visto que todas essas informações são inclusas nesses pacotes.

Algumas melhorias futuras seriam a escolha de um processador que tivesse uma capacidade maior de memória ou até a inclusão de uma memória externa, para solucionar o problema.

Assim, conclui-se que este projeto atingiu os objetivos propostos e declarados no escopo, claro com algumas alterações necessárias e previstas. Salienta-se também que todo o desenvolvimento serviu como uma grande experiência e aprendizado, onde se aprendeu a lidar com prazos, riscos funcionais do projeto, medidas de contingência e o mais importante,

entender que planejar um projeto desse porte requer muito estudo e conhecimento das tecnologias existentes e, além disso, gera uma imensa responsabilidade para os desenvolvedores.

Face ao exposto documento termina-se este projeto com a sensação e o sentimento de dever cumprido.

5 REFERÊNCIAS BIBLIOGRÁFICAS

[1] Associação Brasileira de Automação Residencial. Disponível em:
< http://www.aureside.org.br/default.asp>
Acesso em: 18 de Abril de 2011.

[2] Arduino Home Page. Disponível em:
< http://arduino.cc/ >
Acesso em: 16 de Agosto de 2010.

[3] Arduino Language Reference. Disponível em:
< http://arduino.cc/en/Reference/HomePage >
Acesso em: 16 de Agosto de 2010.

[4] RF22 library for Arduino. Disponível em:
< http://www.open.com.au/mikem/arduino/RF22/ >
Acesso em: 20 de Novembro de 2010.

[5] SPI Ethernet Library. Disponível em:
<http://www.mikroe.com/download/eng/documents/compilers/mikroc/pro/dspic/help/spi_ethernet_library.htm?TB_iframe=true&height=740&width=970 >
Acesso em: 13 de Dezembro de 2010.

[6] EtherCard library. Disponível em:
< http://jeelabs.net/projects/cafe/wiki/EtherCard >
Acesso em: 10 de Março de 2011.

[7] Project Nanode. Disponível em:
< http://wiki.hackspace.org.uk/wiki/Project:Nanode >
Acesso em: 10 de Março de 2010.

[8] HTTP/TCP with an atmega88 microcontroller (AVR web server). Disponível em:
< http://tuxgraphics.org/electronics/200611/embedded-webserver.shtml >
Acesso em: 05 de Novembro de 2010.

[9] Cpluplus.com. Disponível em:
< http://www.cplusplus.com/ >
Acesso em: 20 de Agosto de 2010.

[10] JavaScript Tutorial. Disponível em:
< http://www.w3schools.com/js/default.asp >
Acesso em: 01 de Julho de 2011.

[11] CSS Tutorial. Disponível em:
< http://www.w3schools.com/css/default.asp >
Acesso em: 02 de Julho de 2011.

[12] HTML Tutorial. Disponível em:
< http://www.w3schools.com/html/default.asp >
Acesso em: 03 de Julho de 2011.

[13] Using DS18(B)20 temperature sensor. Disponível em:
< http://sheepdogguides.com/arduino/ar3ne1tt.htm >
Acesso em: 07 de Maio de 2011.

[14] Basic Environment Monitoring using Arduino and Pachube. Disponível em:
< http://blog.thiseldo.co.uk/?p=344 >
Acesso em: 02 de Fevereiro de 2011.

[15] DIY projects: AVR microcontroller electronics. Disponível em:
< http://tuxgraphics.org/electronics/ >
Acesso em: 02 de Fevereiro de 2011.

[16] Pachube. Disponível em:
< https://pachube.com/ >
Acesso em: 10 de Outubro de 2010.

6 APÊNDICE

6.1 Código-fonte do servidor WEB

6.1.1 Monografia.pde

```
// The EtherCard library is based on Guido Socher's driver, licensed as
GPL2.
//######################  CPU  ####################
#define __AVR_ATmega1280__

#include <EtherCard.h>
#include <avr/eeprom.h>

//#############################################
//CONFIGURAÇÃO DOS PINOS
//SAIDAS
#define SOLENOID 4
#define PUMP     3
#define SAIDA1   A4
#define SAIDA2   2
#define SAIDA3   6
#define SAIDA4   8
#define SAIDA5   9

//ENTRADAS
#define AMPSENSOR       A0
#define VIN             A1
#define TANKLEVELSENSOR A2
#define LUMEN           A3
#define ONE_WIRE_BUS    7     //DS18S20 sensor
#define DHT22_PIN       5     //DHT22 sensor

//RESERVADO PARA COMUNICAÇÃO COM ETHERNET SHIELD (SPI)
//10, 11, 12, 13 para ATmega328 ( arduino duemilanove)
//50, 51, 52, 53 para ATmega1280 ( arduino mega)
//#############################################

#define ON 1
#define OFF 0
#define MANUAL 1
#define AUTO 0

//DIGITAL SENSORS: COMMENT OUT IF NOT USED
#define SENSOR_DS18S20
#define SENSOR_DHT22

#define DEBUG   1   // set to 1 to display free RAM on web page
#define SERIAL  0   // set to 1 to show incoming requests on serial port

#define CONFIG_EEPROM_ADDR ((byte*) 0x10)

// configuration, as stored in EEPROM
struct Config {
  boolean levelmode;
  word calibraMax;
  word calibraMin;
  byte minlevel;
  byte maxlevel;
  word capacity;
```

```
  word refresh;
  byte valid; // keep this as last byte
} config;

//VARIAVEIS GLOBAIS SENSORES
struct Var {
  word levelpercent;
  byte levelpercentdec;
  word litros;
  byte litrosdec;
  byte litroscent;
  boolean pumpstate;
  boolean solenoid;
  boolean out1;
  boolean out2;
  boolean out3;
  boolean out4;
  boolean out5;

  byte INTtemperature;
  byte DECtemperature;
  byte INTbatteryVoltage;
  byte DECbatteryVoltage;
  byte INTAMPValue;
  byte DECAMPValue;
  int INTLUXlevel;
  byte DECLUXlevel;
  byte humidity;
  word ADCtanklevel;

  float temperature;
  float AMPValue;
  float batteryVoltage;
  word LUXlevel;

} var;


static void loadVar(){
  var.pumpstate=OFF;
  var.solenoid = OFF;
  var.out1= OFF;
  var.out2= OFF;
  var.out3= OFF;
  var.out4= OFF;
  var.out5= OFF;
  var.humidity = 0;
}

// ethernet interface mac address - must be unique on your network
static byte mymac[] = { 0x74,0x69,0x69,0x2D,0x30,0x31 };
static byte myip[] = { 192,168,0,25 };

static BufferFiller bfill;  // used as cursor while filling the buffer
byte Ethernet::buffer[1600];   // tcp/ip send and receive buffer

static void loadConfig() {
    for (byte i = 0; i < sizeof config; ++i)
        ((byte*) &config)[i] = eeprom_read_byte(CONFIG_EEPROM_ADDR + i);
    if (config.valid != 253) {
      config.levelmode = 0;
```

```
      config.calibraMax = 1024;
      config.calibraMin = 0;
      config.minlevel = 10;
      config.maxlevel = 100;
      config.capacity = 200;
      config.refresh = 5;
      config.valid = 253;
    }
}

static void saveConfig() {
    for (byte i = 0; i < sizeof config; ++i)
        eeprom_write_byte(CONFIG_EEPROM_ADDR + i, ((byte*) &config)[i]);
}

#if DEBUG
static int freeRam () {
  extern int __heap_start, *__brkval;
  int v;
  return (int) &v - (__brkval == 0 ? (int) &__heap_start : (int) __brkval);
}
#endif

#ifdef SENSOR_DS18S20
  //#include "digitalSensors.h"
  #include <OneWire.h>
  #include <DallasTemperature.h>
  OneWire oneWire(ONE_WIRE_BUS);
  DallasTemperature sensors(&oneWire);
  DeviceAddress insideThermometer;
#endif

#ifdef SENSOR_DHT22
  #include <DHT22.h>
  // Setup a DHT22 instance
  DHT22 myDHT22(DHT22_PIN);
#endif

//################################################################################
void setup(){
#if SERIAL
    Serial.begin(57600);
    Serial.println("\n[etherNode]");
#endif
    loadConfig();
    loadVar();

    if (ether.begin(sizeof Ethernet::buffer, mymac) == 0)
      Serial.println( "Failed to access Ethernet controller");
      ether.staticSetup(myip);
#if SERIAL
    ether.printIp("IP: ", ether.myip);
#endif
analogReference(DEFAULT);

pinMode(SOLENOID, OUTPUT);
pinMode(PUMP, OUTPUT);
pinMode(SAIDA1, OUTPUT);
pinMode(SAIDA2, OUTPUT);
pinMode(SAIDA3, OUTPUT);
pinMode(SAIDA4, OUTPUT);
```

```
pinMode(SAIDA5, OUTPUT);
pinMode(TANKLEVELSENSOR, INPUT);
pinMode(AMPSENSOR, INPUT);
pinMode(VIN, INPUT);
pinMode(LUMEN, INPUT);

#ifdef __AVR_ATmega1280__
pinMode(10, INPUT);
pinMode(11, INPUT);
pinMode(12, INPUT);
pinMode(13, INPUT);
#endif

#ifdef SENSOR_DS18S20
  sensors.begin();
  sensors.getAddress(insideThermometer, 0);
  sensors.setResolution(insideThermometer, 9);
#endif
}

static int getIntArg(const char* data, const char* key, int value =-1) {
    char temp[10];
    if (ether.findKeyVal(data + 7, temp, sizeof temp, key) > 0)
        value = atoi(temp);
    return value;
}


//FILTRO MATEMATICO PARA SUAVIZAÇÃO DA LEITURA DE SENSORES ANALOGICOS
float filterSmooth(float currentData, float previousData, float
smoothFactor) {
  if (smoothFactor != 1.0) // only apply time compensated filter if
smoothFactor is applied
    return (previousData * (1.0 - smoothFactor) + (currentData *
smoothFactor));
  else
    return currentData; // if smoothFactor == 1.0, do not calculate, just
bypass!
}

#include "standardHeaders.h"
#include "homePage.h"
#include "outPage.h"
#include "inPage.h"
#include "tanklevelPage.h"
#include "configPage.h"
#include "analogSensors.h"

long previousMillis = 0;
long interval = 500;

//#########################################################################
void loop(){

//ROTINA DE LEITURA SENSORES
if (millis() - previousMillis > interval) {
  //LEITURA SENSOR DE CORRENTE
  var.AMPValue = filterSmooth(readAMP(AMPSENSOR), var.AMPValue, 0.1);
  //LEITURA TENSAO BATERIA
  var.batteryVoltage = filterSmooth(readBatteryVoltage(VIN),
var.batteryVoltage, 0.4);
```

```
  //LEITURA SENSOR_NIVEL
  var.ADCtanklevel = int(filterSmooth(readTankLevel(TANKLEVELSENSOR),
var.ADCtanklevel, 0.4));
  //testado experimentalmente = evita funcionamento erratico devido
flutuações na leitura proximo nivel =0
  if( var.ADCtanklevel <= config.calibraMin ) var.ADCtanklevel =
config.calibraMin;
  //LEITURA LUMINOSIDADE
  var.LUXlevel = filterSmooth(readLumen(LUMEN), var.LUXlevel, 0.5);

//================================================================
  //LEITURA SENSORES DIGITAIS uma vez para seis leituras analogicas
  static byte i=0;
  ++i;
  if(i == 6){

    #ifdef SENSOR_DS18S20
      sensors.requestTemperatures();
      var.temperature = sensors.getTempC(insideThermometer);
    #endif

    #ifdef SENSOR_DHT22
    DHT22_ERROR_t errorCode;
    errorCode = myDHT22.readData();
    switch(errorCode){
      case DHT_ERROR_NONE:
      //temperatura = myDHT22.getTemperatureC();
      var.humidity = myDHT22.getHumidity();
      break;
    }
    #endif

    i=0;
  }
//================================================================

  //ROTINA CASO ESTEJA EM NIVEL AUTOMATICO
  tankautolevel();

  //CONVERSÕES DIVERSAS PARA EXIBIÇÃO NAS PAGINAS
  word span = config.calibraMax - config.calibraMin;
  var.levelpercent = ((var.ADCtanklevel - config.calibraMin)*100)/span;
//porcentagem tanque
  var.levelpercentdec = (long((var.ADCtanklevel -
config.calibraMin))*1000)/long(span) - var.levelpercent*10;       //parte
decimal porcentagem
  var.litros = (long(var.ADCtanklevel - config.calibraMin) *
long(config.capacity))/span;                    //volume atual em litros
  var.litrosdec = (long(var.ADCtanklevel - config.calibraMin) *
long(config.capacity) * 10)/span - var.litros*10;      //parte decimal
volume
  var.litroscent = (long(var.ADCtanklevel - config.calibraMin) *
long(config.capacity) * 100)/span - var.litrosdec*10 - var.litros*100;
//parte centesimal volume

  var.INTtemperature = int(var.temperature);
  var.DECtemperature = int(100*(var.temperature -
float(var.INTtemperature)));
  var.INTbatteryVoltage = int(var.batteryVoltage);
  var.DECbatteryVoltage = int(100*(var.batteryVoltage -
float(var.INTbatteryVoltage)));
```

```
  var.INTAMPValue = int(var.AMPValue);
  var.DECAMPValue = int(100*(var.AMPValue - float(var.INTAMPValue)));
  var.INTLUXlevel = int(var.LUXlevel);
  var.DECLUXlevel = int(100*(var.LUXlevel - float(var.INTLUXlevel)));

  previousMillis = millis();
}
//================================================================
    word len = ether.packetReceive();
    word pos = ether.packetLoop(len);
    // check if valid tcp data is received
    if (pos) {
        bfill = ether.tcpOffset();
        char* data = (char *) Ethernet::buffer + pos;
#if SERIAL
        Serial.println(data);
#endif
        // receive buf hasn't been clobbered by reply yet
        if (strncmp("GET / ", data, 6)== 0)
            homePage(bfill);                          //HOMEPAGE
        else if (strncmp("GET /c", data, 6) == 0)
            configPage(data, bfill);                  //CONFIGPAGE
        else if (strncmp("GET /o", data, 6) == 0)
            outPage(data, bfill);                     //OUTPAGE
        else if (strncmp("GET /e", data, 6) == 0)
            inPage(bfill);                            //INPAGE
        else if (strncmp("GET /l", data, 6) == 0)
            levelPage(data, bfill);                   //TANKLEVELPAGE
        else if (strncmp("GET /h", data, 6) == 0)
            bfill.emit_p(PSTR(
                "HTTP/1.0 302 found\r\n"
                "Location: /\r\n"
                "\r\n"));
        else
            bfill.emit_p(PSTR(
                "HTTP/1.0 401 Unauthorized\r\n"
                "Content-Type: text/html\r\n"
                "\r\n"
                "<h1>401 Unauthorized</h1>"));
        ether.httpServerReply(bfill.position()); // send web page data
    }
}
```

6.1.2 analogSensors.h

```
//LEITURA CORRENTE (AMPERES)
const float readAMP(byte channel) {
  float Aref = 5.0;                               //TENSAO REFERENCIA
ANALOGICA
  float sensorzero = Aref/2 - 0.05;               //sensor center reading
  float AMPsensorScaleFactor = (Aref / 1024.0);
  float sensorSensitivity = 0.023;                 //23mV/Amp DATASHEET
    return ((analogRead(channel) * AMPsensorScaleFactor) -
sensorzero)/sensorSensitivity;
}

//LEITURA TENSAO (VOLTS)
const float readBatteryVoltage(byte channel) {
  float R1   = 80100;                              //DIVISOR TENSAO
  float R2   = 21500;                              //DIVISOR TENSAO
  float Aref = 5.0;
```

```
  float diode = 0.7;
  float batteryScaleFactor = ((Aref / 1024.0) * ((R1 + R2) / R2));
    return (analogRead(channel) * batteryScaleFactor) + diode;
}

//LEITURA NIVEL (SENSOR PRESSAO)
const int readTankLevel(byte channel) {
    return (analogRead(channel));
}

//LEITURA LUMINOSIDADE (FOTORESISTOR)
const int readLumen(byte channel) {
    return (analogRead(channel));
}
```

6.1.3 configPage.h

```
static void configPage(const char* data, BufferFiller& buf) {
    // pick up submitted data, if present
    if (data[6] == '?') {
      word c = getIntArg(data, "c");          //tank capacity liters
      const char* p = strstr(data, "h=");
      int h = strncmp(p, "h=Set", 3);         //calibration high level
      p = strstr(data, "l=");
      int l = strncmp(p, "l=Set", 3);         //calibration low level
      byte m = getIntArg(data, "m");          //minimun desired level
      byte M = getIntArg(data, "M");          //maximum desired level
      word r = getIntArg(data, "r");          //page refresh rate
      //save configuration if required parameters are met
      if (0 < c && 0 <= m && M <= 100 && 1 <= r && r <= 3600 || h==0 ||l==0
&& m < M) {
          // store values as new settings
          config.capacity = c;
          config.minlevel = m;
          config.maxlevel = M;
          config.refresh = r;
          if (h == 0) config.calibraMax = var.ADCtanklevel;
          if (l == 0) config.calibraMin = var.ADCtanklevel;

          saveConfig();
          loadConfig();
          // redirect to the home page
          buf.emit_p(PSTR(
              "HTTP/1.0 302 found\r\n"
              "Location: /\r\n"
              "\r\n"));
          return;
      }
    }

    buf.emit_p(PSTR("$F\r\n"
      "$F"), okHeader, pagestyle);

    buf.emit_p(PSTR(
    "<b>Server Configuration</b><form><p>"
    "Maximum Tank Capacity <input type=text name=c value='$D' size=4>
Liters<hr>"
    "Tank Level Calibration <br/>"
    "Set Current Level = Full Tank <input type=submit name=h
value=Set><br/>"
```

```
    "Set Current Level = Empty Tank <input type=submit name=l value=Set>"
    "<hr>Automatic Level Mode Characteristics <br/>"
    "Minimum Desired Level <input type=text name=m value='$D' size=4> %
<br/>"
    "Maximum Desired Level <input type=text name=M value='$D' size=4>
%<hr>"
    "Page Refresh Rate <input type=text name=r value='$D' size=4> (1..3600
seconds)"
    "</p><input type=submit value=Set></form>"), config.capacity,
config.minlevel, config.maxlevel, config.refresh);
}
```

6.1.4 homePage.h

```
static void homePage(BufferFiller& buf) {
    long t = millis() / 1000;
    word h = t / 3600;
    byte m = (t / 60) % 60;
    byte s = t % 60;

    buf.emit_p(PSTR("$F\r\n"
        "<meta http-equiv='refresh' content='$D'/>"
        "$F"
        ), okHeader, config.refresh, pagestyle);

    buf.emit_p(PSTR(
    "<table border=2 cellspacing=5 cellpadding=2 width=1000 height=400>"
    "<tr bgcolor=lightgreen height=50><th colspan=4 align=center>Status
Overview</th></tr>"
    "<tr bgcolor=grey><th>Temperature: $D,$D C</th><th>Output 1:
$S</th><th>Tank Level: $D,$D %</th><th>Max Tank Level: $D L</th></tr>"
    "<tr bgcolor=lightgrey><th>Humidity: $D %</th><th>Output 2:
$S</th><th>Tank Mode: $S</th><th>Max Desired Level: $D %</th></tr>"
    ), var.INTtemperature, var.DECtemperature, var.out1 ? "ON" : "OFF",
var.levelpercent, var.levelpercentdec, config.capacity, var.humidity,
var.out2 ? "ON" : "OFF",
    config.levelmode ? "MANUAL" : "AUTO", config.maxlevel);

    buf.emit_p(PSTR(
    "<tr bgcolor=grey><th>Voltage: $D,$D V</th><th>Output 3:
$S</th><th>Pump: $S</th><th>Min Desired Level: $D %</th></tr>"
    "<tr bgcolor=lightgrey><th>Current: $D,$D A</th><th>Output 4:
$S</th><th>Solenoid: $S</th>"
    ), var.INTbatteryVoltage, var.DECbatteryVoltage, var.out3 ? "ON" :
"OFF", var.pumpstate ? "ON" : "OFF", config.minlevel,var.INTAMPValue,
var.DECAMPValue,
    var.out4 ? "ON" : "OFF", var.solenoid ? "ON" : "OFF");

    buf.emit_p(PSTR(
    "<th>Uptime: $D$D:$D$D:$D$D</th></tr>"
    ), h/10, h%10, m/10, m%10, s/10, s%10);

    buf.emit_p(PSTR(
    "<tr bgcolor=grey><th>Luminosity: $D,$D</th><th>Output 5:
$S</th><th>Liters: $D,$D$D</th><th>RAM: $D bytes free</th></tr>"
    "<tr bgcolor=lightgrey><th>Pressure:
$D</th><th></th><th></th><th></th></tr></table>"
//    "<div style=background-color:#FFA500;clear:both;text-align:center;>"
//    "Copyright &#169 2011 Eletrica do Aroldo"
    ), var.INTLUXlevel,var.DECLUXlevel, var.out5 ? "ON" : "OFF",
var.litros, var.litrosdec, var.litroscent, freeRam(), var.ADCtanklevel);
```

```
}
```

6.1.5 inPage.h

```
static void inPage( BufferFiller& buf) {

    buf.emit_p(PSTR("$F\r\n"
    "<meta http-equiv='refresh' content='$D'/>"
    "$F"
    ), okHeader, config.refresh, pagestyle);

    // else show a configuration form
  buf.emit_p(PSTR(
      "<table border=2 cellspacing=5 cellpadding=2 width=1000 height=400>"
       "<tr bgcolor=lightblue height=50><th colspan=2 align=center>Digital
Inputs</th>"
       "<th colspan=4 align=center>Analog Inputs</th></tr>"
       "<tr bgcolor=lightgrey height=50>"
       "<th >Temperature</th><th >Humidity</th><th >Voltage</th><th
>Current</th><th >Luminosity</th><th >Pressure</th></tr>"
       "<tr bgcolor=lightgrey>"
       "<th>$D,$D C</th>"
       "<th>$D %</th>"
       "<th>$D,$D V</th>"
       "<th>$D,$D A</th>"
       "<th>$D,$D</th>"
       "<th>$D</th>"
       "</tr>"
       "<tr><td bgcolor=black></td><td bgcolor=red></td><td
bgcolor=green></td><td bgcolor=yellow></td><td bgcolor=blue></td>"
       "<td bgcolor=brown></td></tr></table><br/>"), var.INTtemperature,
var.DECtemperature, var.humidity, var.INTbatteryVoltage,
var.DECbatteryVoltage,
       var.INTAMPValue, var.DECAMPValue, var.INTLUXlevel, var.DECLUXlevel,
var.ADCtanklevel);
}
```

6.1.6 outPage.h

```
//#define ON 1
//#define OFF 0

static void outPage(const char* data, BufferFiller& buf) {
    // pick up submitted data, if present
    if (data[6] == '?') {
      switch (data[7]){
        case 'r':
        var.solenoid = (data[10] == 'N') ? ON : OFF;
        digitalWrite(SOLENOID, var.solenoid);
        break;
        case 's':
        var.out2 = (data[10] == 'N') ? ON : OFF;
        digitalWrite(SAIDA2, var.out2);
        break;
        case 't':
        var.out3 = (data[10] == 'N') ? ON : OFF;
        digitalWrite(SAIDA3, var.out3);
        break;
        case 'u':
```

```
        var.out4 = (data[10] == 'N') ? ON : OFF;
        digitalWrite(SAIDA4, var.out4);
        break;
        case 'v':
        var.out5 = (data[10] == 'N') ? ON : OFF;
        digitalWrite(SAIDA5, var.out5);
        break;
      }
    }

    //standard pagestyle
    buf.emit_p(PSTR("$F\r\n"
    "<meta http-equiv='refresh' content='$D'/>"
    "$F"
    ), okHeader, config.refresh, pagestyle);

    //send outpage
    buf.emit_p(PSTR(
    "<table border=2 cellspacing=5 cellpadding=2 width=1000 height=400>"
    "<tr bgcolor=lightblue height=50><th colspan=5
align=center>Outputs</th></tr>"
    "<tr bgcolor=lightgrey height=50>"
    "<th >Solenoid</th><th >Output 2</th><th >Output 3</th><th >Output
4</th><th >Output 5</th></tr>"
    "<form><tr bgcolor=lightgrey>"
    "<th>Turn:<br/><input type=submit name=r value=$S></th>"
    "<th>Turn:<br/><input type=submit name=s value=$S></th>"
    "<th>Turn:<br/><input type=submit name=t value=$S></th>"
    "<th>Turn:<br/><input type=submit name=u value=$S></th>"
    "<th>Turn:<br/><input type=submit name=v value=$S></th>"
//    "<th><input type=radio name=out1 value=on /> Ligado<br /><input
type=radio name=out1 value=off /> Desligado</th>"
    "</tr>"
    "<tr><td bgcolor=$S></td><td bgcolor=$S></td><td bgcolor=$S></td><td
bgcolor=$S></td><td bgcolor=$S></td>"
    "</tr></table><br/>"
    "</form>"), var.solenoid ? "OFF" : "ON", var.out2 ? "OFF" : "ON",
var.out3 ? "OFF" : "ON", var.out4 ? "OFF" : "ON", var.out5 ? "OFF" : "ON",
    var.solenoid ? "green" : "red", var.out2 ? "green" : "red", var.out3 ?
"green" : "red", var.out4 ? "green" : "red", var.out5 ? "green" : "red");
}
```

6.1.7 standardHeaders.h

```
char okHeader[] PROGMEM =
    "HTTP/1.0 200 OK\r\n"
    "Content-Type: text/html\r\n"
    "Pragma: no-cache\r\n"
;

char pagestyle[] PROGMEM =
//    "<title>Automa&#231&#227o de Maquinas</title>"
    "<title>Engine Room Automation</title>"
    "<div style=background-color:#FFA500;>"
//    "<h1 style=margin-bottom:0;text-align:center;>Projeto de
Automa&#231&#227o</h1></div>"
    "<h1 style=margin-bottom:0;text-align:center;>Engine Room Automation
System</h1></div>"
    "<div style=background-
color:#FFD700;height:400px;width:200px;float:left;>"
    "<h2><a href='h'>Menu</a></h2><br />"
```

```
    "<h4><a href='e'>Inputs</a>"
    "<h4><a href='o'>Outputs</a>"
    "<h4><a href='l'>Tank Level</a>"
    "<h4><a href='c'>Configuration</a></div>"
//    "<h4><a href='c'>Configura&#231oes</a></div>"
    "<div style=background-color:#EEEEEE;height:400px;>"
;
```

6.1.8 tanklevelPage.h

```
//#define MANUAL 1
//#define AUTO 0

//FUNÇAO PARA NIVEL AUTOMATICO DO TANQUE
void tankautolevel(){
  word span = config.calibraMax - config.calibraMin;
  if(config.levelmode == AUTO){
    if((var.ADCtanklevel - config.calibraMin) <
(word(config.minlevel)*span)/100){  //nivel < desejado
      digitalWrite(PUMP, HIGH);
      var.pumpstate = ON;
    }
    if((var.ADCtanklevel - config.calibraMin) >=
(word(config.maxlevel)*span)/100){  //nivel >= desejado
      digitalWrite(PUMP, LOW);
      var.pumpstate = OFF;
    }
  }
}

//PAGINA TANKLEVEL
static void levelPage(const char* data, BufferFiller& buf) {
  loadConfig();
    // pick up submitted data, if present
    if (data[6] == '?') {
      switch (data[7]){
        case 'm':
        config.levelmode = (data[9] == 'M') ? MANUAL : AUTO;  //levelmode =
manual/auto
        saveConfig();
        break;
        case 'p':
        var.pumpstate = (data[10] == 'N') ? ON : OFF;    //pump state
        break;
        case 'v':
        var.solenoid = (data[10] == 'N') ? ON : OFF;    //solenoid state
        break;
        case 'r': //return to homepage
        buf.emit_p(PSTR(
                "HTTP/1.0 302 found\r\n"
                "Location: /\r\n"
                "\r\n"));
        break;
      }
    }

    //tklevel entre 0 e 400 para exibiçao do grafico
    word level = var.levelpercent*4;

    //pump ON/OFF
    if(var.pumpstate==ON){
```

```
      digitalWrite(PUMP, HIGH);
    } else {
      digitalWrite(PUMP, LOW);
    }

    //solenoid ON/OFF
    if(var.solenoid==ON){
      digitalWrite(SOLENOID, HIGH);
    } else {
      digitalWrite(SOLENOID, LOW);
    }

    //envia pagina html
    buf.emit_p(PSTR("$F\r\n"
//    "<title>Automa&#231&#227o de Maquinas</title>"
    "<title>Engine Room Automation</title>"
    "<meta http-equiv='refresh' content='$D'/>"
    "<TABLE BORDER=2 CELLSPACING=5 CELLPADDING=2 width=800 height=550>"
    "<tr style=font-size:30px;background-color:orange;>"
    "<th height=50 colspan=2>Tank Overview</th></tr>"
    "<TR height=50 style=font-size:30px;background-color:#b0c4de;>"
    "<th width=400>Configuration</th>"
    "<TH>Tank Level: $D,$D%</TH></TR>"
    "<TR><TD align=justify bgcolor=lightgrey><form>"
    "<p><b> Current Mode: $S </p>"
    "<input type=submit name=m value=Manual>"
    "<input type=submit name=m value=Auto><hr noshade>"), okHeader,
config.refresh, var.levelpercent, var.levelpercentdec, config.levelmode ?
"MANUAL" : "AUTO");

    if(config.levelmode==MANUAL){
      buf.emit_p(PSTR(
      "<p> Pump </p>"
      "<input type=submit name=p value=ON>"
      "<input type=submit name=p value=OFF><hr>"
      "<p> Solenoid Valve </p>"
      "<input type=submit name=v value=ON>"
      "<input type=submit name=v value=OFF><hr noshade>"
      "</form>"));
    }

    buf.emit_p(PSTR(
    "<p>Pump is: $S</p>"
    "<p>Solenoid is: $S</p>"
    "<p>Current Volume: $D,$D$D Liters</b></p>"
    "</TD>"
    "<TD bgcolor=lightgreen valign=bottom align=center>"
    "<table><TR><TD bgcolor=blue width=200 height=$D></TD></table>"
    "</TD>"
    "</TR></TABLE><br />"
    "<form><input type=submit name=r value='Return to Homepage'></form>"),
var.pumpstate ? "ON" : "OFF",
    var.solenoid ? "ON" : "OFF", var.litros, var.litrosdec, var.litroscent,
level);
}
```