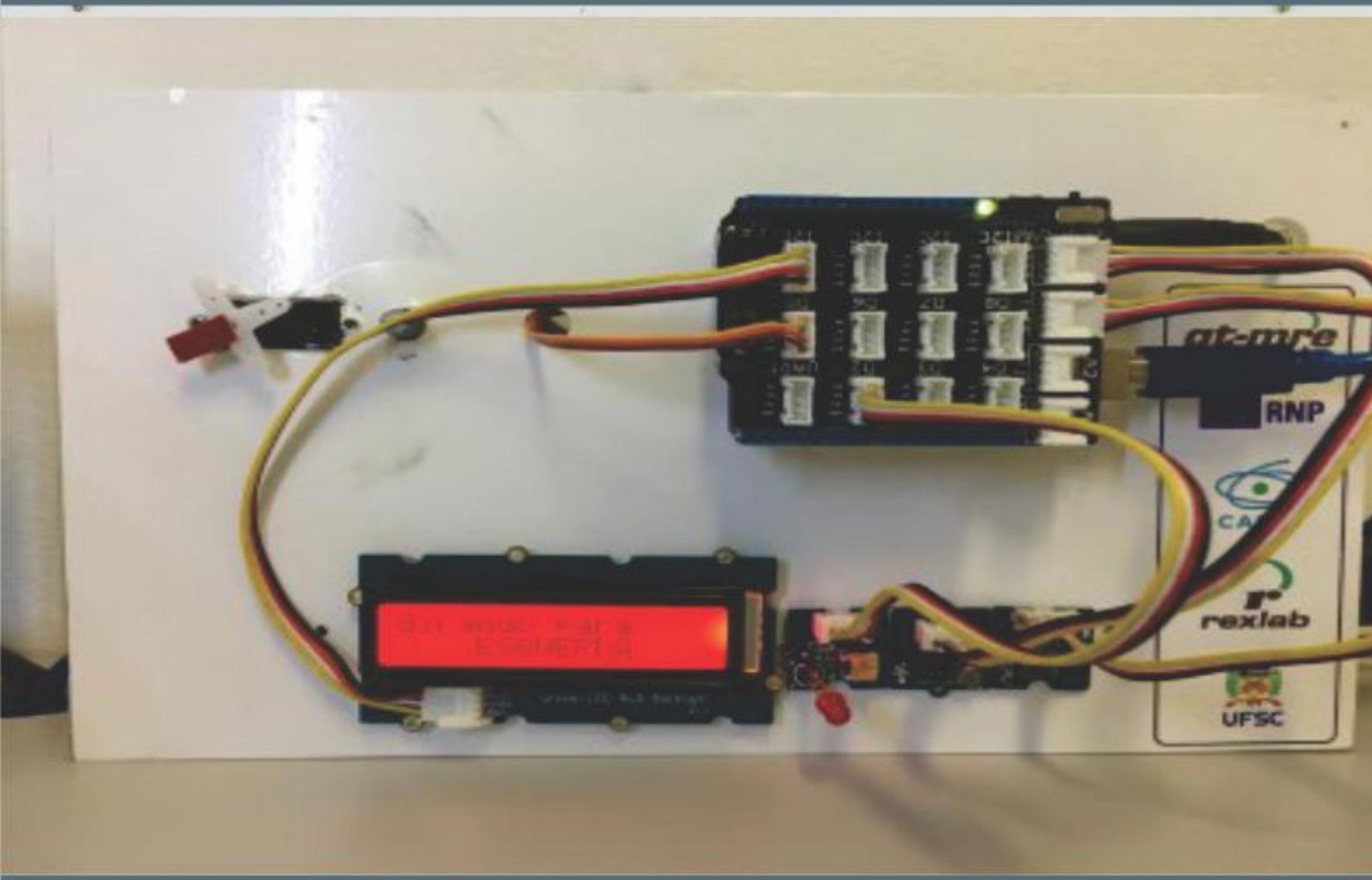
Manual Técnico

Ambiente para Desenvolvimento em Arduino



Experimentação Remota Móvel para o Ensino Básico e Superior

rexlab
LABORATÓRIO DE EXPERIMENTAÇÃO REMOTA
UNIVERSIDADE FEDERAL DE SANTA CATARINA
RNP
CAPES

Manual Técnico do Experimento Remoto Ambiente para Desenvolvimento em Arduino:
Experimentação Remota Móvel para a Educação Básica e Superior
Este guia, cada capítulo e suas imagens estão licenciados sob a licença Creative Commons
Rua Pedro João Pereira, 150, Mato Alto – CEP 88900-000
http://rexlab.ufsc.br/
rexlabufsc@gmail.com

Elaboração
Juarez Bento da Silva
João Paulo Cardoso de Lima
José Pedro Schardosim Simão
Josiel Pereira
Lucas Mellos Carlos
Editoria de arte, projeto gráfico e capa
Isabela Nardi da Silva
Ilustrações
Alex Moretti

Manual Técnico do Experimento Remoto Ambiente para Desenvolvimento em Arduino:
Experimentação Remota Móvel para a Educação Básica e Superior / obra coletiva concebida, desenvolvida e produzida pelo Laboratório de Experimentação Remota (RExLab)

Araranguá - SC, Brasil, 2016

Experimento Ambiente de Desenvolvimento para Arduino

Apresentação

O experimento Ambiente para Desenvolvimento em Arduino tem como objetivo auxiliar nos estudos e práticas relacionadas a programação para o microcontrolador Arduino, e permite a verificação, carregamento de códigos e manipulação de um Arduino remotamente, além do controle de sensores e atuadores destinados à alunos do Ensino Médio e Superior.

Arquitetura

O dispositivo está implementado a partir da estrutura padrão de hardware e software básico. Na Figura 1 pode ser visualizado o diagrama da arquitetura doexperimento remoto Ambiente para Desenvolvimento em Arduino.

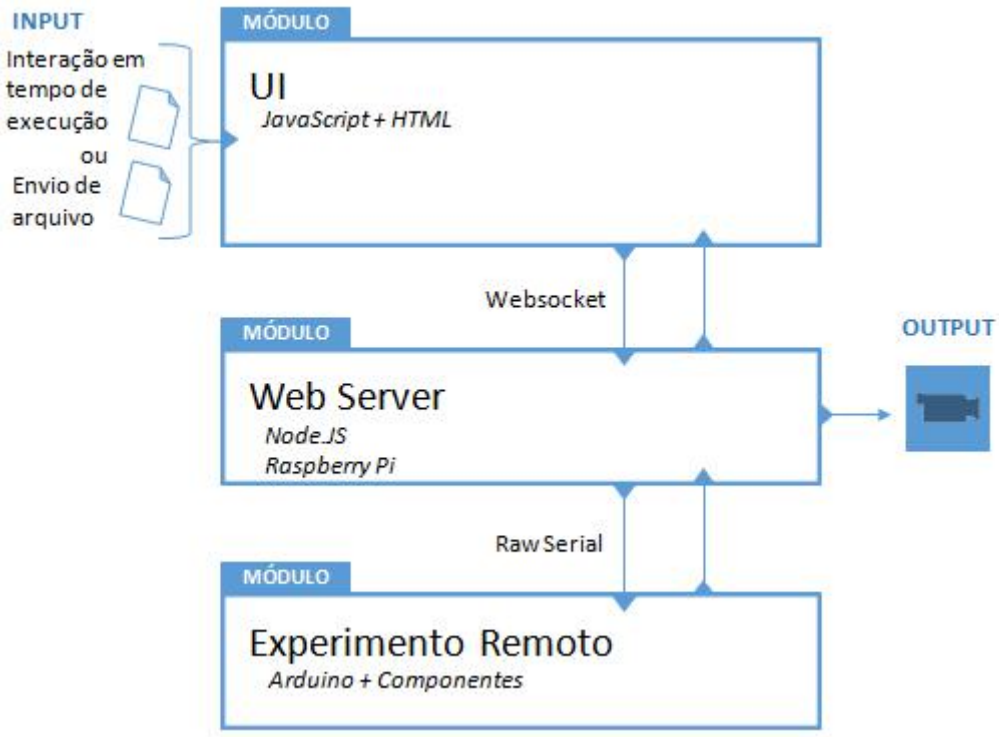


Figura 1 - Arquitetura do experimento: Ambiente para Desenvolvimento em Arduino.

Interface de Usuário (UI)

O experimento está disponível no sistema de gerenciamento RELLE(Remote Labs Learning Enviromment), que provê uma série de funcionalidades necessárias para a administração de experimentos remotos.

A interface de acesso ao experimento foi criada utilizando HTML juntamente com o framework front-end Bootstrap, o mesmo traz uma série de componentes prontos para o desenvolvimento além de prover tratamento para diferentes tipos de resoluções de telas. Além de HTML e Bootstrap, é utilizada a biblioteca jQuery que traz uma série de funções JavaScript que simplificam o desenvolvimento.

A Figura 2 mostra a organização do experimento Ambiente de Desenvolvimento para Arduino no RELLE.

Figura 2 - Interface do usuário no RELLE

Web Server

Atualmente, há uma ampla gama de bibliotecas e frameworks para construção de serviços web. Apesar de serviços baseados em HTTP predominarem na Internet, o uso do protocolo WebSocket é uma tendência em aplicações corporativas de grande porte. Uma das plataformas para desenvolvimento web para construção de serviços baseados em WebSocket é o framework NodeJS.

O NodeJS permite construir aplicações de servidor e de rede facilmente escaláveis. Ele é composto por um ambiente de execução multiplataforma e de código fonte aberto que interpreta códigos de aplicações escritas em Javascript. O NodeJS usa um modelo orientado a evento, com operações de entrada e saída não bloqueantes. Por este motivo, ele é ideal para aplicações em tempo real com troca intensa de dados entre dispositivos distribuídos.

A API para acesso às funcionalidades do SmartDevice[1] contém funções vinculadas à *listeners*, comuns ao paradigma de orientação a eventos. Este módulo usa a biblioteca Socket.io e é o ponto de partida da aplicação, onde o servidor é iniciado e eventos são vinculados. O Socket.io é formado por dois componentes: servidor e cliente, ao qual usa principalmente o protocolo WebSocket, e polling HTTP como compatibilidade reversa.

A autorização de sessão no SmartDevice garante a integridade do acesso exclusivo, já que o dispositivo exposto como um serviço pode ser utilizado concorrentemente por outro cliente. Apesar de algumas funcionalidades poderem ser utilizadas no modo observador, como consultar o estado das chaves e metadados, as funcionalidades de controle necessitam de consulta ao sistema de fila.

O sistema de fila, ou mesmo agendamento, pode ser externo ou interno ao SmartDevice. O primeiro é baseado em um token de autenticação provido pelo usuário e validado pelo SmartDevice. As implementações dos experimentos de física exemplificam o uso do sistema de reserva externo (próprio do RELLE). Já o controle de acesso no próprio SmartDevice é exemplificado pela implementação do Laboratório de desenvolvimento em Arduino, pois neste encontra-se um modelo de acesso diferente dos anteriores.

O código fonte desenvolvido para comunicação serial e gerência dos sensores e atuadores são complementos para o NodeJS escritos em C++. Estes complementos são objetos compartilhados de vínculo dinâmico que pretendem dar suporte a códigos nativos, rapidez e portabilidade. Esses objetos compõem a abstração de cada experimento físico, que é representado por métodos e atributos intrínsecos a cada um. Por exemplo, são definidos os métodos de “get” e “set” para saídas digitais, “get” para valores de sensores, “get” e “set” para calibragem e configuração dos sensores.

O dispositivo central do experimento é o servidor de laboratório, que na plataforma desenvolvida pelo GT-MRE a escolha recaiu sobre o RaspberryPi[2], (Figura 4) modelo B+, que tem como principal função intermediar os acessos aos demais dispositivos de hardware dos experimentos via rede.

O servidor de laboratório (SL) tem função prover interfaceamento e gerenciamento para a conexão entre a rede (web) e a “placa de aquisição e

[1] DOI: 10.1109/REV.2015.7087292

[2] O Raspberry Pi é um computador é baseado em um system on a chip (SoC) Broadcom BCM2835, que inclui um processador ARM1176JZFS rodando a 700 MHz, GPU VideoCore IV, e 512 MB de memória RAM em sua última revisão. O Raspeberry PI foi desenvolvido no Reino Unido pela Fundação Raspberry Pi.

controle” (PAC). O SL acessa a PAC para a coletar os dados dos sensores ou para enviar comandos para os atuadores. Essa comunicação é feita via porta UART(Universal asynchronous Receiver/Transmitter) que se comunica via protocolo MODBUS[3].

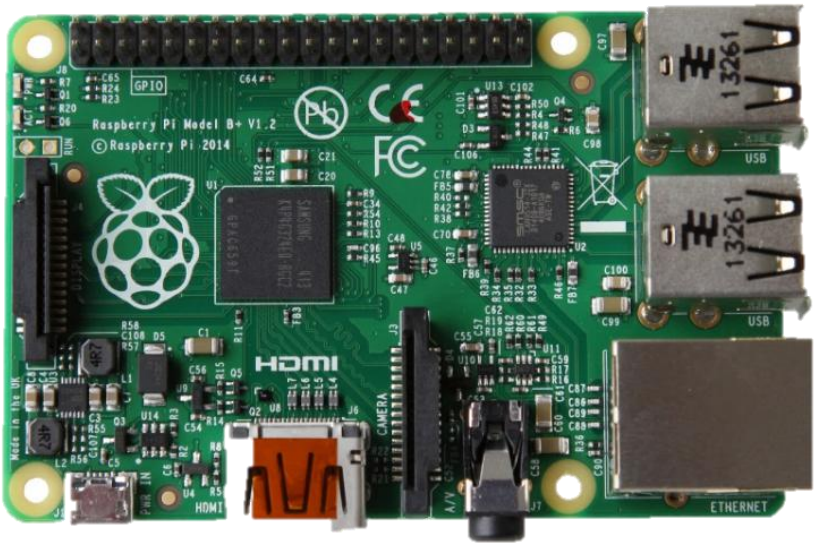


Figura 3 – Raspberry Pi, Model B+

API WebSocket

Os componentes da aplicação são suficientemente leves para serem executados por uma placa Raspberry Pi ou outro computador Linux de baixo custo. Um dos componentes, a API WebSocket, oferece uma interface aos sensores e atuadores na estrutura de um serviço web. A aplicação não requer alto uso da memória e pode ser utilizada em qualquer sistema Linux.

O resultado é uma arquitetura fracamente acoplada, adotada pelo GT-MRE, que habilita o compartilhamento dos experimentos em outras plataformas. Esse paradigma, chamado de SmartDevice já é utilizado no projeto Go-Lab[4], no qual estão bem destacadas aplicações clientes e servidor, e fornecem interfaces bem definidas entre o usuário e o sistema.

Os tópicos seguintes apresentam com mais detalhes aspectos do serviço web utilizado no servidor de experimento, bem como as funcionalidades

[3]Modbus é um protocolo de comunicação de dados utilizado em sistemas de automação industrial. É um dos protocolos mais utilizados em redes de Controladores lógicos programáveis (PLC) para aquisição de sinais (0 ou 1) de instrumentos e comandar atuadores. É de utilização livre e sem taxas de licenciamento.
[4]http://www.go-lab-project.eu/

internas e as motivações para o uso de certos protocolos, padrões e ferramentas de desenvolvimento, conforme a Figura 4.

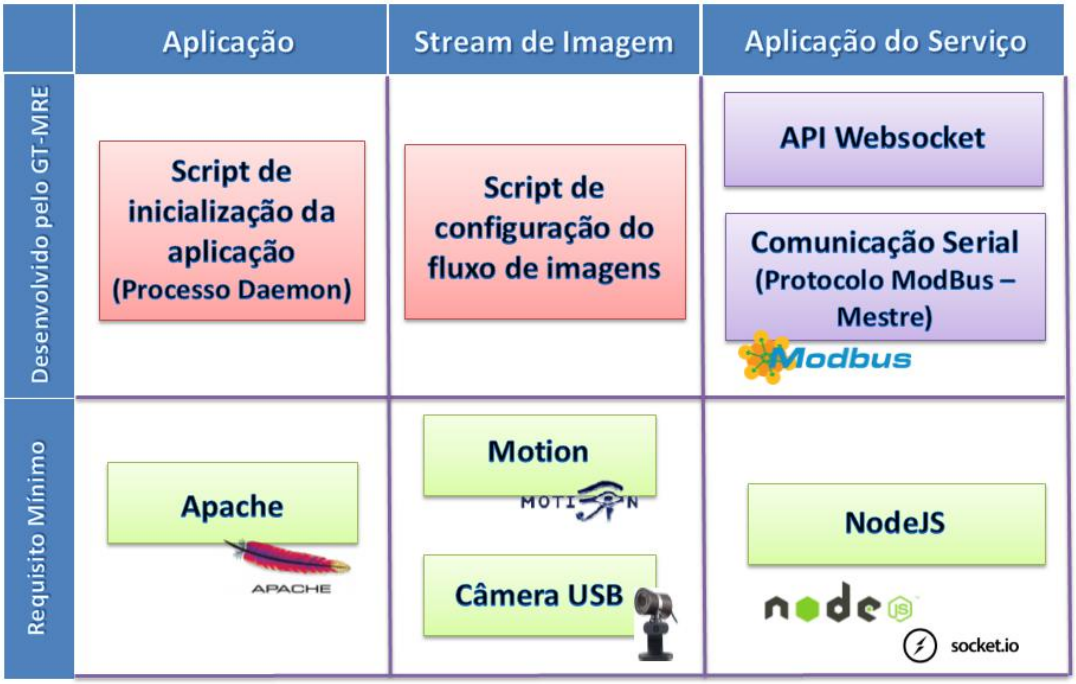


Figura 4 - Esquema de aplicação embarcada. Fonte: GT-MRE.

Controle e monitoramento do experimento

O SmartDevice é capaz de comunicar-se com sensores através do barramento serial (Porta UART). Ao invés de usar o protocolo serial em sua forma bruta, optamos por incluir o protocolo Modbus na camada de aplicação para identificação de erros, endereçamento e controle de colisão. Conectados ao mesmo barramento (rede), cada sistema embarcado, responsável por um ou mais sensores ou atuadores, é um dispositivo escravo que responde às requisições da aplicação que é executada no Raspberry Pi.

Um dos módulos desenvolvidos para aplicação é responsável pela função de fila externo ou interno, sendo possível acoplar o serviço de fila provido pelo RELLE ou habilitar serviços internos. No primeiro caso, a aplicação utiliza a lógica necessária para validação de token de sessão enviado pelo cliente. Na segunda, todo processo realizado pela web API de fila é realizado pelo SmartDevice.

Acesso à web API pelo cliente

A Figura 5 apresenta o esquema de comunicação no uso da API desenvolvido para o serviço/protótipo.

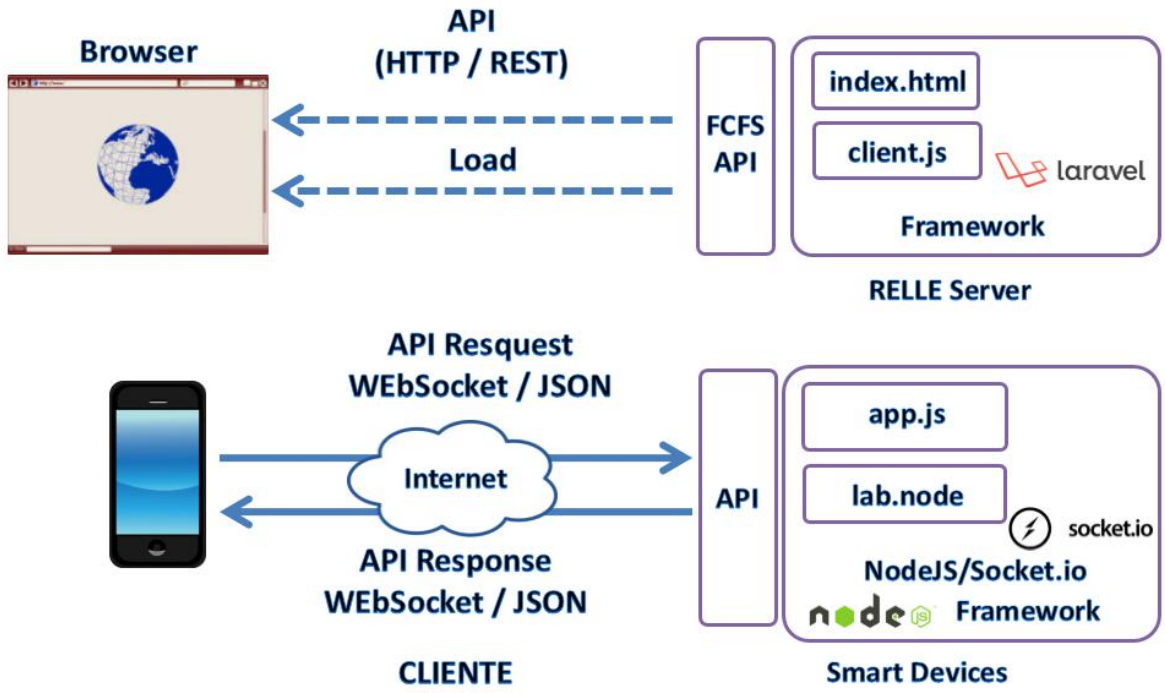


Figura 5 - Esquema de comunicação crossdomain no uso da API desenvolvida pelo GT-MRE. Fonte: Autores.

O cliente web disponibilizado pelo sistema RELLE é composto por um arquivo html, css e javascript diferentes para cada experimento. O RELLE provê uma página comum para cada experimento onde carrega os dados que foram inseridos no momento da publicação do experimento (armazenados numa base de dados). Por exemplo, o experimento de ID 1 é acessível pela URL “relle.ufsc.br/labs/1” pelo método GET e contém suas informações dentro do layout padrão do sistema. A partir do botão “Acessar” é possível disparar um evento para comunicação com a Web API FCFS (first-come first-served).

Ao obter a permissão no navegador, o cliente navegador poderá carregar os arquivos (html, css e js), pois a API já tem o seu token de sessão como usuário sendo servido. Após carregar o cliente para o SmartDevice (client.js), uma conexão WebSocket com este dispositivo é estabelecida.

Streaming de imagens

No GT-MRE foi optado pelo uso de câmeras web com conexão USB devido ao baixo custo e a facilidade de aquisição. O mesmo computador

embarcado utilizado para controle do experimento também é o responsável pelo gerenciamento e disponibilização do streaming no formato MJPEG (Motion JPEG). O MJPEG é um formato de compressão de vídeo na qual cada frame de vídeo é comprimido separadamente como uma imagem JPEG.

Visto que existem muitos servidores de streaming de código aberto, optou-se pelo Motion[5] para explorar aspectos de leveza (utilização de poucos recursos) e configuração flexível. O Motion é um software escrito em C para sistemas Linux que usa a API de vídeo Linux, e é capaz de detectar se uma parte significante da imagem tem mudado. Algumas variáveis são ajustadas através de seu arquivo de configuração principal para adequar-se aos requisitos de nossa aplicação.

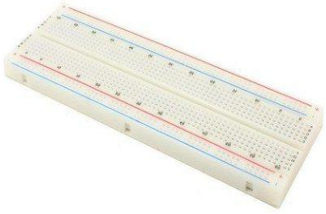
Atualmente, os principais navegadores do mercado como Firefox, Google Chrome e Safari já possuem o suporte nativo para o streaming MJPEG. Para clientes Android existem bibliotecas de código fonte aberto para incluir um visualizador MJPEG em aplicações de código nativo.

[5]http://www.lavrsen.dk/foswiki/bin/view/Motion/WebHome

Experimento Remoto

O sistema físico implementado em cada um dos experimentos remotos. A seguir serão listados alguns dos componentes presente no experimento.

	– Sensor de temperatura LM35. Usado na experiência para que o usuário possa medir a temperatura do ambiente, com uma taxa de variação de 0,15ºC e uma faixa de -55ºC a 150ºC.
	– O diodo emissor de luz, conhecido como LED é um atuador usado para a emissão de luz, a qual se torna mais conveniente a aplicação.
	– Servo Motor.
	– Visor LCD 16x2. O visor LCD possui 16 colunas em cada uma de suas 2 linhas. Sua função é exibir qualquer valor que o usuário deseje mostrar. Possui um controlador HD44780.

	– Arduino Uno. O micro controlador Arduino Uno é uma placa onde o usuário pode conectar componentes tais como os usados neste experimento e realizar a programação para que execute as funções desejadas pelo mesmo.
	– A protoboard de 830 serve para que os componentes possam ser espetados na placa e se liguem com a placa através de jumpers, sendo assim, a protoboard um local mais adequado para trabalhar com resistores, capacitores e demais componentes.

O dispositivo está implementado a partir da comunicação serial via cabo USB com a placa Arduino Uno que recebe dados advindo do computador embarcado Raspberry Pi. O circuito da aplicação é composto por uma protoboard, onde são “espetados” os componentes, dos quais pode-se utilizar uma grande variedade de recursos como o uso misto de componentes, explorar funções como as de Fading. O modelo atual implementado da arquitetura com seus sensores e atuadores devidamente encaixados é exibido na figura abaixo.

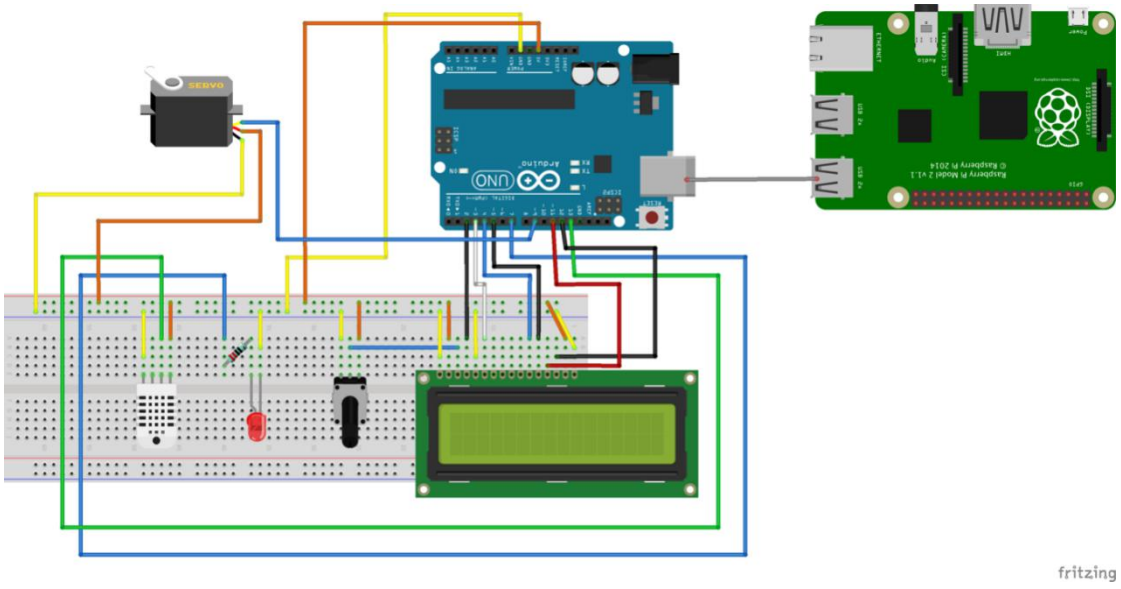


Figura 6 - Ambiente para Desenvolvimento em Arduino.

A Figura 7 apresenta o diagrama elétrico do “Ambiente para Desenvolvimento em Arduino”. O circuito é composto por uma placa Arduino Uno, uma protoboard de 830 furos, um LED vermelho, um resistor de 220 Ohms, um potenciômetro, um visor LCD 16x2, um servo motor, um sensor de temperatura e umidade DHT11 e jumpers.

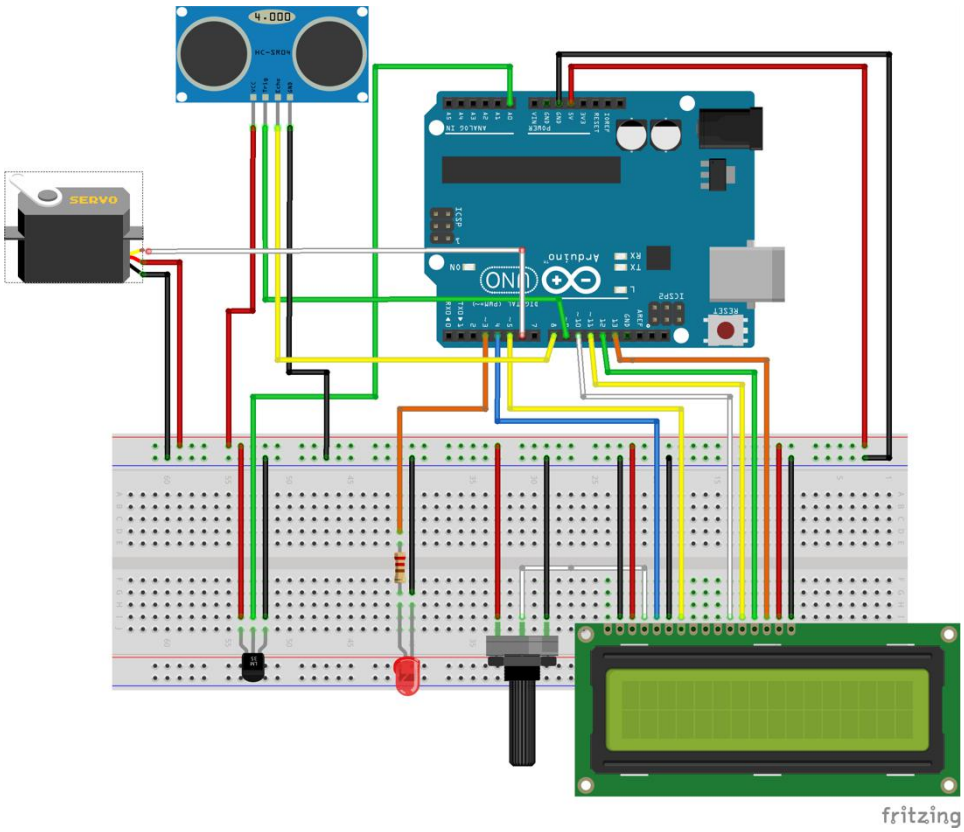


Figura 7 - Ambiente para Desenvolvimento em Arduino – Diagrama Elétrico.

Circuito elétrico nº 1

A Figura 8 apresenta um exemplo de uso do circuito, onde o visor LCD está em uso, podendo assim, o usuário exibir quais quer dígitos no dispositivo. Para o uso deste dispositivo, faz-se presente um potenciômetro capaz de regular o contrates do mesmo.

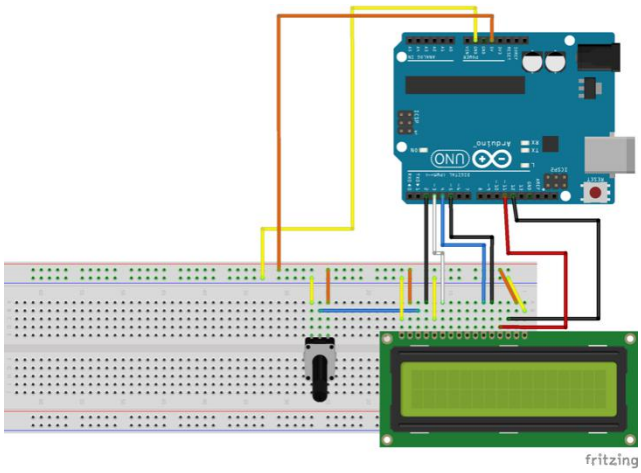


Figura 8 - Ambiente para Desenvolvimento em Arduino – Circuito nº 1

Circuito elétrico nº 2

O circuito possui um LED Vermelho que quando acionado proporciona uma queda de tensão de 1.8V e suporta uma corrente máxima de 0.02A, tal é acionado por um pino digital de 5V, logo faz-se necessário o uso de um resistor para conter a tensão elétrica a qual é fornecida pela placa Arduino.

A Figura 9 apresenta o uso de um LED, a qual encontra-se plugado no pino digital 7. Assim, o usuário pode definir o acionamento como bem entender.

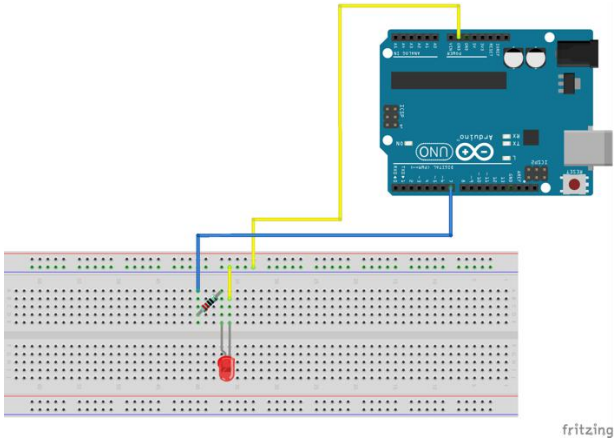


Figura 9 - Ambiente para Desenvolvimento em Arduino – Circuito nº 2

Circuito elétrico nº 3

A Figura 10 apresenta o uso do Servo motor onde este encontra-se plugado no pino 9 da placa Arduino através da *protoboard* e acionado por uma tensão de 5V. Logo, o usuário tem a possibilidade de fazer com que o atuado gire de 0º até 180º no sentido horário e anti-horário.

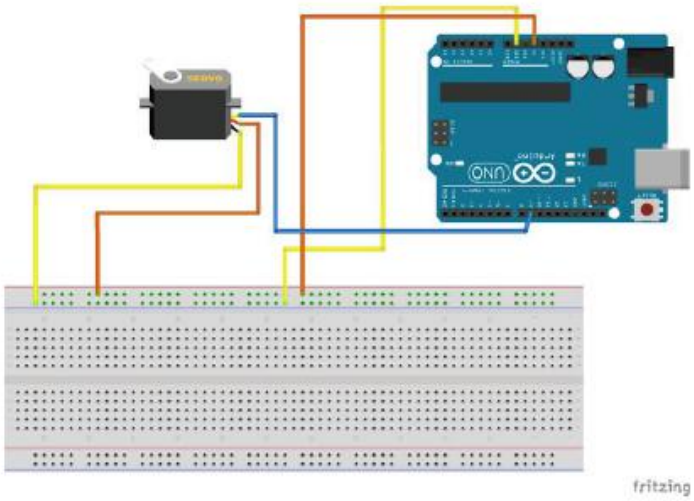


Figura 10 - Ambiente para Desenvolvimento em Arduino – Circuito nº 3

Circuito elétrico nº 4

O sensor LM35 realiza a medição da temperatura do ambiente, sendo possível realizar a medida de -55 a 150°C para com uma taxa de erro de 0.5°C.

Sua tensão de alimentação é de 4 a 30V em corrente contínua, este sensor possui 3 pinos, a qual sua pinagem é descrita na Figura 11.

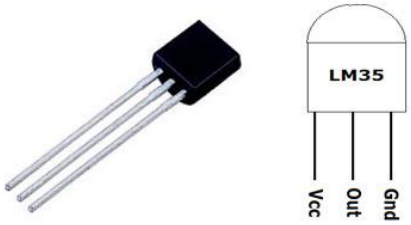


Figura 11 - Pinagem Sensor LM35

A Figura 12 apresenta o uso do sensor de temperatura e umidade LM35. Para o uso deste sensor faz-se necessário o uso da comunicação serial e/ou do visor LCD para que os valores sejam exibidos.

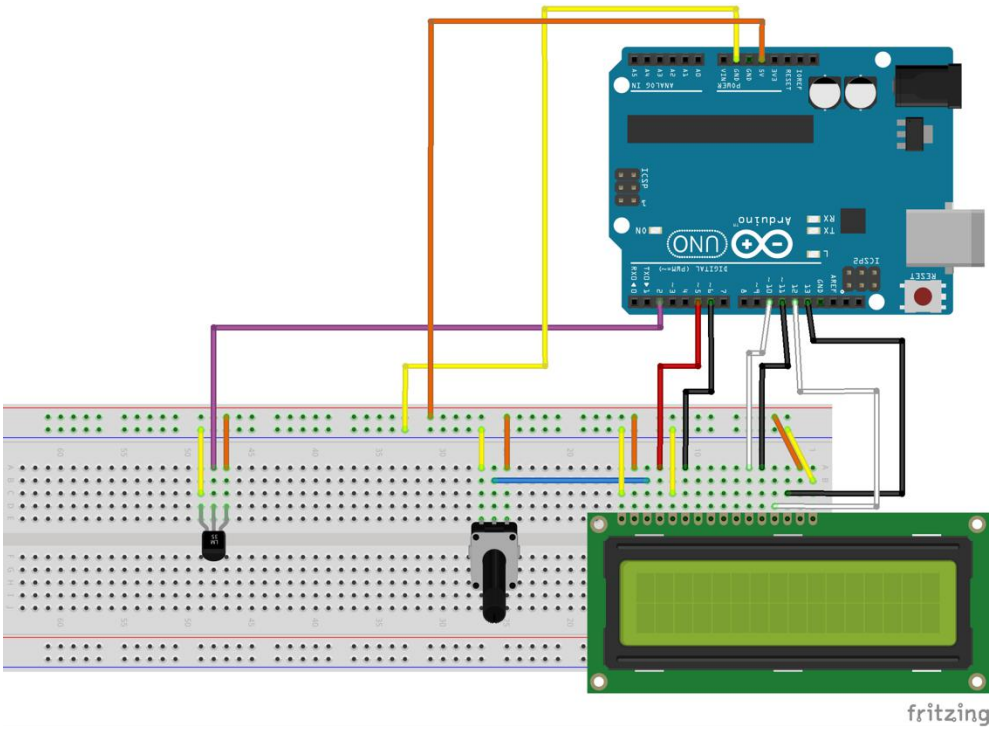


Figura 12 - Ambiente para Desenvolvimento em Arduino – Circuito de aplicação nº 4

Circuito elétrico nº 5

Os circuitos encontram-se de forma mixada a combinar o uso de alguns dos sensores e atuadores presente na experiência de tal forma que podem ser combinados com funções de delay[6], entre outros. Ou ainda para esse exemplo o uso de um sensor ultrassônico HC-SR04 a Figura 13 exibe o modelo deste sensor, servo motor e visor LCD/terminal serial para exibir os valores referente a distância apontada pelo movimento do sensor ultrassônico através do giro sobre o servo motor.

Figura 13 - Sensor HC-SR04

A Figura 14 exibe como foi realizada a configuração deste circuito.

[6] Documentação da função delay(). Disponível em https://www.arduino.cc/en/Reference/Delay. Acessado em 28 de out. de 2015.

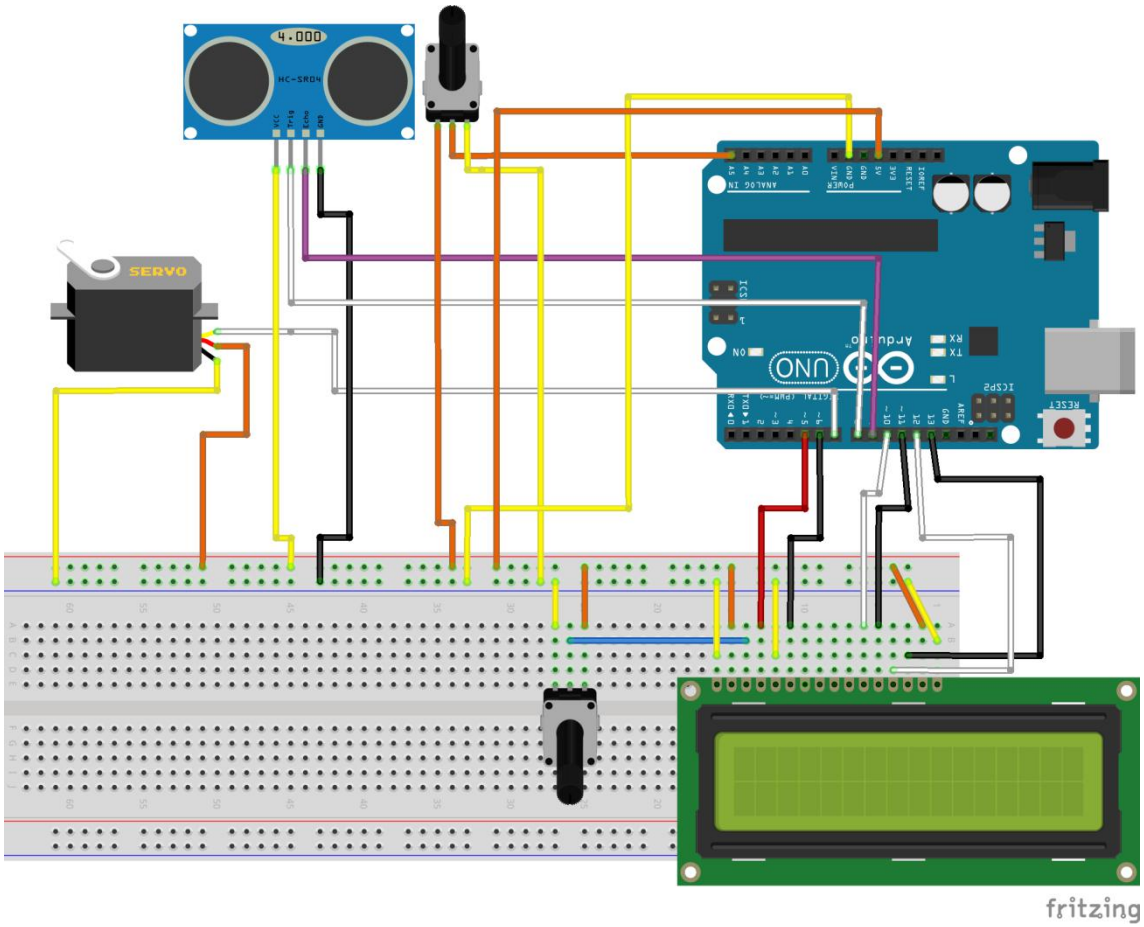


Figura 14 - Ambiente para Desenvolvimento em Arduino – Circuito de aplicação nº 5

Apêndices

Tutorial de reinicialização do experimento

Para reiniciar o experimento usa-se um terminal para conexão ssh, por exemplo o software PuTTY, o qual pode ser baixado pelo seguinte endereço: http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html. Utilizando o PuTTY, basta inserir o endereço IP do experimento que se deseja reinicializar.

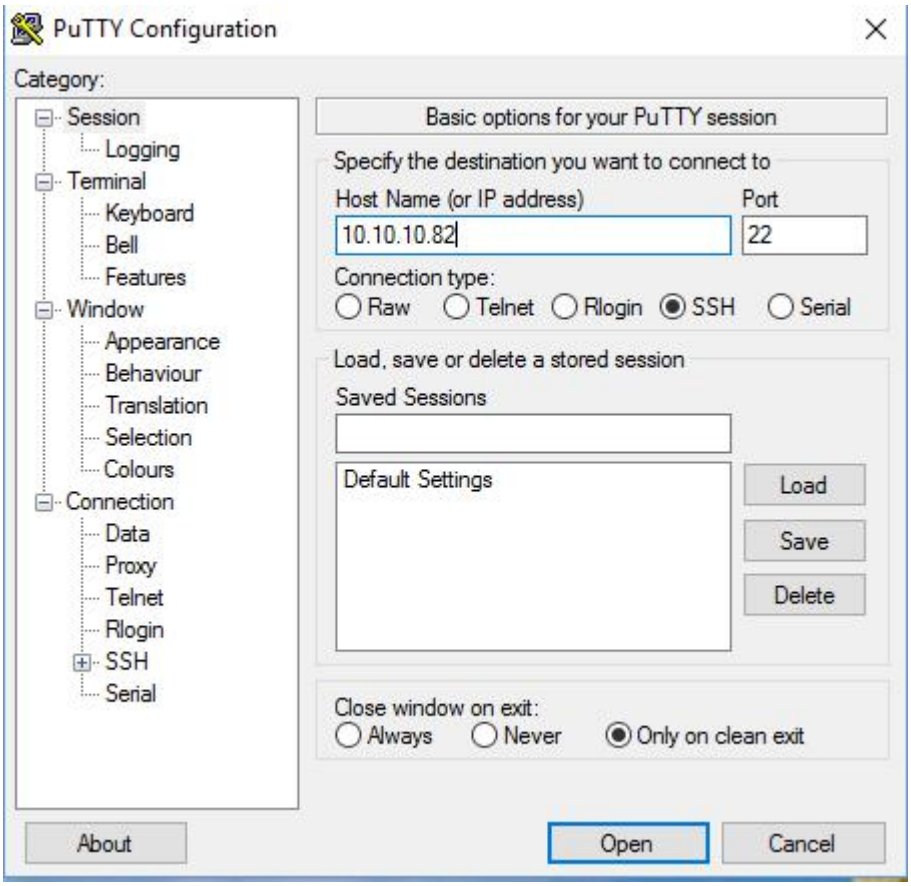


Figura 15 – PuTTY

Ao ativar a conexão será aberto um terminal (Figura 15), o qual solicitará um usuário (user) para autenticação. Recomenda-se autenticar com o usuário root, logo em seguida será requisitada a senha do computador embarcado. E por fim, para reiniciar o computador embarcado, digite o comando *reboot* no terminal.

Figura 15 - Terminal SSH com experimento

Verificação e reinicio do serviço

Para verificar se os serviços do laboratório remoto estão rodando, basta usar o comando "*ps –aux | grep node"* que irão verificar os processos rodando referente ao servidor web Node.JS responsável por executar o serviço da aplicação. Caso o serviço esteja rodando, o resultado será algo similar a Figura 16 que exibe o usuário e número do processo em execução. Neste caso o processo PID 2434.

```
root@raspberrypi:~# ps -aux | grep node
warning: bad ps syntax, perhaps a bogus '-'?
See http://gitorious.org/procps/procps/blobs/master/Documentation/FAQ
root      2434  0.1  9.7 1182324 43412 ?       Sl   May12 102:36 /usr/local/bin/node /home/conducao_app/apps.js
root     21479  0.0  0.3   3520  1740 pts/0    S+   14:17   0:00 grep node
```

Figura 16 - Verificação do serviço

Ações de iniciar, pausar ou verificar status do serviço podem serem executadas usando os comandos *"service labx start|stop|status".*

Manutenção do streaming de vídeo

O vídeo é transmitido pelo software Motion. Para instalação do software pode-se fazer seu download via repositório através do comando *"apt-get install motion"* e acessar os arquivos de configurações *motion* e *motion.conf*através de algum editor de código no diretório */etc/default/motion* definindo o parâmetro *start_motion_deamon* para o valor *yes.*

As configurações relacionadas a qualidade da imagem e a transmissão ficam disponíveis no arquivo motion.conf no diretório /etc/motion/. Ainda para inicio da transmissão os parâmetros *deamon* e *webcam_localhost* devem ser mudados para *on* e *off*, respectivamente.