Concurso Casa Conectada

PROJETO: PETIOT

Desenvolvido por: João Henrique Bellincanta Gomes
Empresa: CloudHome Serviços e Tecnologia

Problema: como monitorar e alimentar os pets em uma residência, tudo isso em um projeto IoT (internet das coisas)?

Inicialmente quando pensei no projeto, havia tido a ideia de fazer algo relacionado ao consumo de água, mas como achei que esse tema já seria abordado por algumas pessoas, pensei em desenvolver algo que atendesse as minhas necessidades - e a de muitas pessoas - naquele momento.

Eu e minha esposa temos um cachorro, que há um tempo foi diagnosticado com obesidade. Tínhamos um grande problema, pois trabalhamos fora o dia inteiro e não teríamos como dosar a alimentação dele durante o dia. Ele precisa comer nas horas determinadas pelo veterinário e deve comer medidas exatas. Como poderíamos alimenta-lo da forma correta para que melhorássemos sua qualidade de vida?

Além da minha situação atual, pesquisei muito sobre esse mercado PET no Brasil e realmente fiquei muito surpreso com o que encontrei. Recentemente a revista Veja, publicou dados do IBGE que apontam que as famílias brasileiras já possuem mais cães do que crianças, e que esse mercado movimenta cerca de 9 bilhões por ano!!

Foi então que pensei em algo que utilizando o Freescale adotasse o conceito IoT, internet das coisas para me ajudar e atender a esse mercado que a cada ano só vem aumentando.

Solução: Desenvolvi o PetIot, baseado em Freescale / mbed.

Trata-se de um equipamento que permite:
- Alimentar o pet;
- Monitorar o reservatório de alimento;
- Detectar quando o animal procura e deseja alimentar;
- Informar a temperatura e humidade da comida ou água;
- Filmar e enviar em tempo real quando o animal estiver utilizando o PetIot;
- Possibilidade de interação com o animal remotamente.

Porque atende o conceito Iot?

O equipamento possui conexão com um Websocket, o que permite enviar e receber informações com rapidez e segurança.

Esclarecimentos

Gostaria de deixar claro, que o PetIot se trata de um protótipo. Eu não sou engenheiro e nunca fiz nenhum tipo de curso técnico em eletrônica. Todo conhecimento utilizado no projeto, foi baseado em exemplos do site MBED e pesquisas no Google, além de alguns conhecimentos adquiridos em eventos como IotWeekend. Portanto peço que considerem o projeto como um protótipo, e que pode ser melhorado, inclusive conforme a demanda de mercado.

Primeiras impressões Freescale

O primeiro contato ao receber o freescale, em termos de apresentação visual e tamanho, me interessou bastante, entretanto nos primeiros minutos de uso confesso que não foi das melhores experiências devido a:

- Falta de documentação para instalar no MAC;
- Não conseguir depurar usando Kinects;
- No início achei pouca documentação e confuso o processo de desenvolvimento no Kinects;

Entretanto após conhecer a plataforma MBED, as coisas começaram a ficar mais claras. Tive como desenvolvimento a plataforma “developer.mbed.org” e foi mais fácil encontrar documentação e exemplos, porém pouco documentados. Há também limitação de apenas programar em C, que confesso me incomodou um pouco.

Após as primeiras impressões e algumas horas de desenvolvimento, o processo fica mais simples e mais fácil de desenvolver, embora a plataforma online fique um

pouco lenta. Contudo o equipamento é bem intuitivo e funcional, além de ser prático na hora de carregar o software, e usar os recursos já existem como botões, led RGB e acelerômetro.

Ao final, posso afirmar que gostei do equipamento e com certeza irei usar em projetos embarcados onde precise fazer I/O e integração com sensores e outros equipamentos eletrônicos.

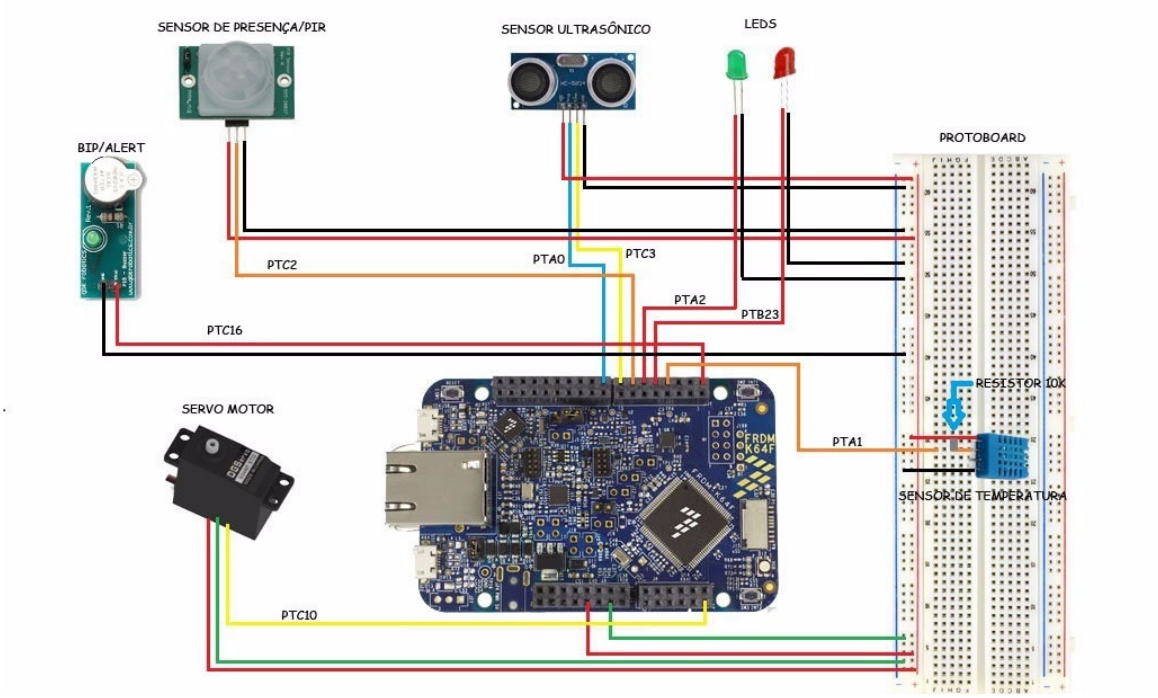
Montagem eletrônica

Para este projeto tentei usar o máximo de sensores e componentes que pudesse, pois desta forma iria maximizar a experiência e testes com a plataforma Freescale / MBED.

Desta forma utilizei:

- HCSR04: sensor ultrassônico para calcular o nível do reservatório através da distância.
- PIR: sensor para detectar movimento, para alertar quando o animal se aproximar.
- DHT15: sensor de temperatura e humidade, para alerta de alta temperatura e baixa humidade para recomendar mais água ao animal.
- LED vermelho: para informar falha de conexão iot.
- LED verde: para informar falha de conexão iot.
- Ethernet: uso da interface de rede para conexão.
- Servo Motor: para liberar a ração do animal de estimação.

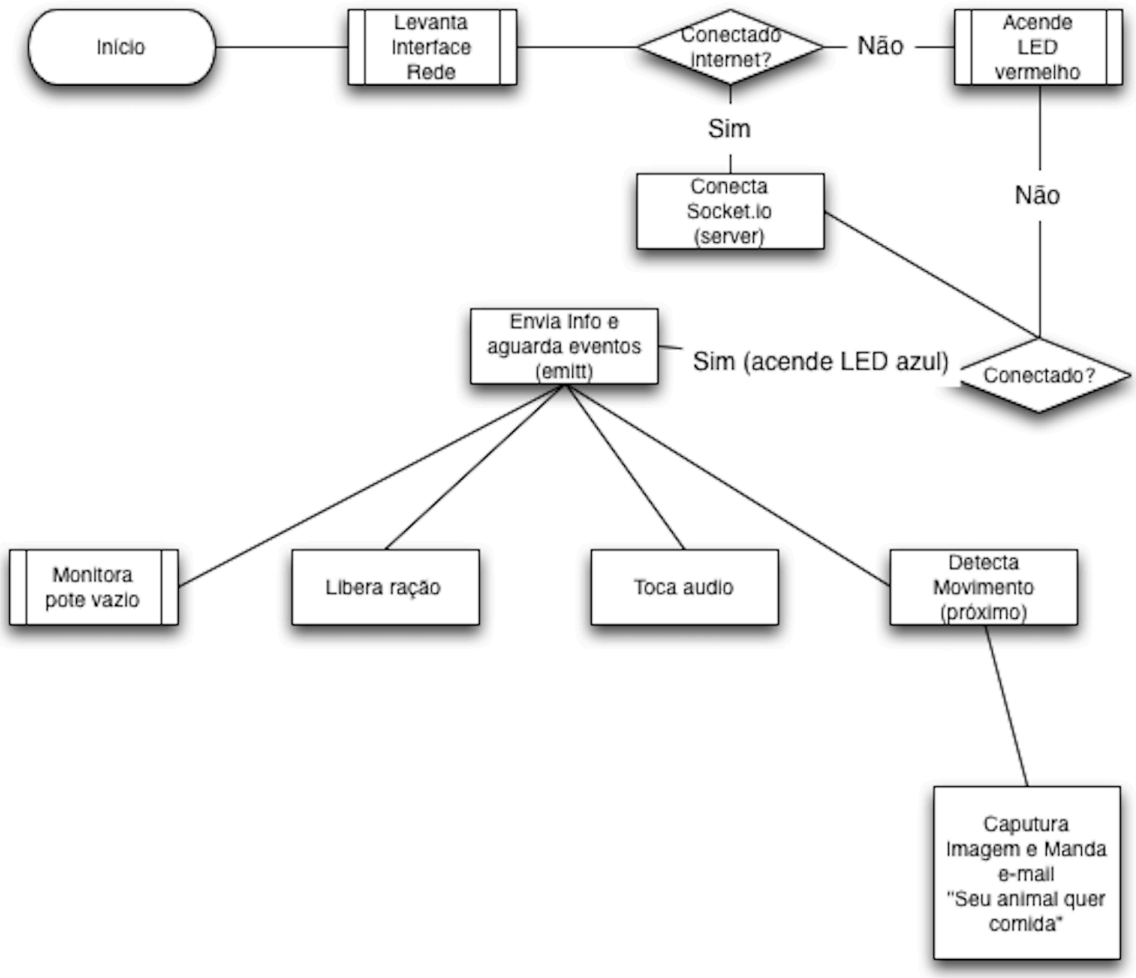
O esquema eletrônico ficou da seguinte forma:



As portas de comunicação usadas foram:

Equipamento	Porta Comunicação
Ultrasonico	Echo: PTC3 trigger : PTA0
PIR	PTC2
LED Verde	PTA2
LED Vermelho	PTB23
DHT15	PTA1
Buzzer	PTA16
Servo	PTC10

Resultado após montagem no protótipo foi:



Programação

Para que tivesse uma comunicação direta da placa Freescale com um aplicativo remoto e intuitivo, usei a tecnologia de Websocket.

O fluxo de troca de informação ficou da seguinte forma:

[Freescale] <> WebSocket Server <> Client HTML5

Tecnologia usada foi:

- Linguagem C no Freescale;
- Linguagem NodeJS para criar o Websocket Server;
- HTML5 + CSS + Javascript para permitir criar um frontend atrativo e intuitivo, além de permitir transformar em um aplicativo através do Phonegap, Ionic e ou outros.

https://developer.mbed.org/users/cloudhome/code/Pet-iot-ws/

O código de programação do Freescale ficou:

```
#include "mbed.h"
#include "DHT.h"
#include "hcsr04.h"
#include "EthernetInterface.h"
#include "Websocket.h"
#include "MbedJSONValue.h"
#include <string>

//### Declaracao do sensor de movimento
InterruptIn motion(PTC2);
int motion_detected = 0;
void irq_handler(void)
{
   motion_detected = 1;
}

//### Declaracao do sensor de distancia
HCSR04 usensor(PTA0, PTC3); //usensor(TRIGGER, ECHO)
unsigned int dist;

//### Inicio do programa
int main() {
   //### Inicio da interface de rede
   EthernetInterface eth;
   //### Declaracao do LED para indicar conexao de WS
   DigitalOut ledON(PTA2);
   //### Declaracao do LED para indicar erro de conexao de WS
   DigitalOut ledOFF(PTB23);
   //### Declaracao do SERVO que ira liberar alimento
   DigitalOut servo(PTC10);
   //### Declaracao de sensor de temperatura
   DHT sensor(PTA1,SEN11301P);
   //### Inicializacao do sensor de movimento
   motion.rise(&irq_handler);
   //### Declaracao do buzzer
   DigitalOut buzzer(PTC16);

   //### Variavel para armazener sinal de erro
   int err;

   printf("Inciando programa Pet-iot-ws\r\n");
```

```
    //### Iniciar inteface de rede
    eth.init(); //Use DHCP
    eth.connect();
    printf("IP adquirido %s\n\r", eth.getIPAddress());

    //### Tenta se conectar ao socket server
    printf("Inciando conexao com o servidor Websocket\r\n");
    Websocket ws("ws://IP_SERVIDOR:3000/ws/freescale");

    if(ws.connect()){
        //### Variavel para armazenar as mensagems JSON
        MbedJSONValue mensagemJSON;
        //### Variavel para os parses do JSON
        std::string tipo;
        std::string mensagem;

        printf("Conectado com sucesso\r\n");

        //### Liga o LED verde
        ledON  = 1;
        //### Desliga o LED vermelho
        ledOFF = 0;

        char recv[256];
        int res = ws.send("{\"tipo\": \"conexao\", \"mensagem\": \"Freescale Pet IoT -
Conectado\", \"destino\": \"ws/cliente\" }");
        while (1) {
          if (ws.read(recv)) {
            printf("Dados recebidos: %s\r\n", recv);
            parse(mensagemJSON, recv);
            tipo     = mensagemJSON["tipo"].get<std::string>();
            mensagem  = mensagemJSON["mensagem"].get<std::string>();

            if (tipo == "comida"){
                  if(mensagem == "liberar"){
                    printf("Recebendo evento comida\r\n");
                    int i;
                    for (i=0; i < 200; ++i) {
                      servo = 1; // Toggle the LED state
                      wait_ms(1); // 200 ms
                      servo = 0; // LED is OFF
                      wait_ms(10); // 1 sec
                    }
                    MbedJSONValue json;
                    std::string str;
```

```
                json["tipo"]      = "resp_comida";
                json["mensagem"]   = "Comida liberada com sucesso";
                json["destino"]    = "ws/cliente";
                str             = json.serialize();
                char *jsonChar = new char[str.length() + 1];
                strcpy(jsonChar, str.c_str());
                ws.send(jsonChar);
                delete [] jsonChar;
            }
        }
        else if(tipo == "reservatorio"){
            if(mensagem == "consultar"){
                int total = 20; //Distancia total em CM do reservatorio vazio
                int percentual = 0;
                usensor.start();
                wait_ms(500);
                dist=usensor.get_dist_cm();
                printf("Distancia em CM:%ld\r\n",dist);
                if(dist != 0 && dist != NULL){
                   percentual = (int)((dist*100)/total);
                   percentual = 100-percentual;
                }
                MbedJSONValue json;
                std::string str;
                json["tipo"]      = "resp_reservatorio";
                json["mensagem"]   = percentual;
                json["destino"]    = "ws/cliente";
                str             = json.serialize();
                char *jsonChar = new char[str.length() + 1];
                strcpy(jsonChar, str.c_str());
                ws.send(jsonChar);
                delete [] jsonChar;
            }
        }
        else if(tipo == "sensor_movimento"){
            if(motion_detected) {
                MbedJSONValue json;
                std::string str;
                json["tipo"]      = "resp_sensor_movimento";
                json["mensagem"]       = "Movimento detectado, seu animal quer
comida.";
                json["destino"]    = "ws/cliente";
                str             = json.serialize();
                char *jsonChar = new char[str.length() + 1];
                strcpy(jsonChar, str.c_str());
```

```
            ws.send(jsonChar);
            delete [] jsonChar;
        }
    }
    else if(tipo == "toca_audio") {
        int i;
        //### Toca 3 vezes o beep
        for (i=0; i < 3; ++i) {
            buzzer = 1;
            wait(.5);
            buzzer = 0;
            wait(.5);
            printf("Conectado com sucesso\r\n");
        }
        MbedJSONValue json;
        std::string str;
        json["tipo"]       = "resp_toca_audio";
        json["mensagem"]   = "Audio emitido com sucesso";
        json["destino"]    = "ws/cliente";
        str                = json.serialize();
        char *jsonChar = new char[str.length() + 1];
        strcpy(jsonChar, str.c_str());
        ws.send(jsonChar);
        delete [] jsonChar;
    }
    else if(tipo == "temp_humidade"){
        int tentativasLeitura = 10;
        for(int i=0;i < tentativasLeitura; ++i){
            err = sensor.readData();
            if(err == 0) {
                break;
            }
            wait(0.5);
        }
        if (err == 0) {
            MbedJSONValue json;
            std::string str;
            json["tipo"]       = "resp_temp_humidade";
            json["mensagem"]     = "Dados de temperatura e humidade capitado
com sucesso";
            json["temperatura"] = (int)(sensor.ReadTemperature(CELCIUS) * 1 +
0.5);
            json["humidade"]   = (int)sensor.ReadHumidity();
            json["destino"]    = "ws/cliente";
            str                = json.serialize();
```

```
                char *jsonChar = new char[str.length() + 1];
                strcpy(jsonChar, str.c_str());
                ws.send(jsonChar);
                delete [] jsonChar;
            } else {
                printf("Erro ao ler a temperatura e humidade\r\n");
            }
        } else {
            //### Mensagem nao reconhecida
        }
    }
    /*
     *  Mensagens serem enviadas como notificacao e
     */
    //### Verifica o nivel do reservatorio para ver se precisa avisar o usuario
    int total = 18; //Distancia total em CM do reservatorio vazio
    int percentual = 0;
    usensor.start();
    wait_ms(500);
    dist=usensor.get_dist_cm();
    printf("Distancia em CM:%ld\r\n",dist);
    if(dist != 0 && dist != NULL){
        percentual = (int)((dist*100)/total);
        percentual = 100-percentual;
    }
    if(percentual > 5 && percentual < 10){
        printf("Nivel do reservatorio esta baixo, avisar usuario\r\n");
        MbedJSONValue json;
        std::string str;
        json["tipo"]      = "alerta";
        json["mensagem"]    = "Nivel do reservatorio esta baixo, coloque mais racao
no reservatorio";
        json["destino"]    = "ws/cliente";
        str             = json.serialize();
        char *jsonChar = new char[str.length() + 1];
        strcpy(jsonChar, str.c_str());
        ws.send(jsonChar);
        delete [] jsonChar;
    }
    //### Verifica se teve movimento para avisar o usuario
    if(motion_detected) {
        printf("Movimento detectado\r\n");
        MbedJSONValue json;
        std::string str;
        json["tipo"]      = "alerta";
```

```
            json["mensagem"]    = "Movimento detectado, seu animal quer comida.";
            json["destino"]     = "ws/cliente";
            str                 = json.serialize();
            char *jsonChar = new char[str.length() + 1];
            strcpy(jsonChar, str.c_str());
            ws.send(jsonChar);
            delete [] jsonChar;
            motion_detected = 0;
          }
          //### Verifica alera para alta temperatura e ou baixa humidade
          err = sensor.readData();
          if (err == 0) {
            int temperatura = (int)(sensor.ReadTemperature(CELCIUS) * 1 + 0.5);
            int humidade    = (int)sensor.ReadHumidity();
            if(temperatura > 30 || humidade < 30){
              MbedJSONValue json;
              std::string str;
              json["tipo"]          = "alerta";
              if(temperatura > 30){
                json["mensagem"]    = "Atencao, alta temperatura nao deixe de dar agua
para o seu animal.";
              } else {
                json["mensagem"]    = "Atencao, baixa humidade, nao deixe de dar agua
para o seu animal.";
              }
              json["temperatura"]   =   (int)(sensor.ReadTemperature(CELCIUS)   *   1   +
0.5);
              json["humidade"]    = (int)sensor.ReadHumidity();
              json["destino"]     = "ws/cliente";
              str                 = json.serialize();
              char *jsonChar = new char[str.length() + 1];
              strcpy(jsonChar, str.c_str());
              printf(str.c_str());
              ws.send(jsonChar);
              delete [] jsonChar;
            }
          }
          //### Verifica termperatura / humidade
          wait(0.5);
        }
      } else {
        //Ligado led vermelho
        printf("Nao foi possivel conectar com servidor\r\n");
        //### Desliga o LED verde
        ledON  = 0;
```

```
      //### Liga o LED vermelho
      ledOFF = 1;
   }
}
```

O código de programação do servidor de Websocket, está disponível em:

https://github.com/Cloudhomebr/PetIot-ws-server

Código bem comentado e simples de entender. Para instalar e rodar é necessário ter o NojdeJS instalado e executar os seguintes comandos:

- npm install (para instalar os módulos);
- npm start ou node app.js (para iniciar o servidor);

A implementação está bem simples, pois a ideia é ser apenas um protótipo podendo ser melhorado no futuro.

Anexado ao ZIP do projeto.

O código de programação do Aplicativo, interface para o cliente está disponível em:

https://github.com/Cloudhomebr/PetIot-ws-client

Basta baixar o projeto e colocar em servidor web, apache ou outro totalmente compatível, ou seja, existe a possibilidade de transformar em um aplicativo e colocar na Apple Store e ou Play Store.

Anexado ao ZIP do projeto.

CONSIDERAÇÃO:

Por se tratar de um protótipo, muito ainda deve ser feito, como:

- Implementar segurança na comunicação socket;
- Criar uma interface para criação de contas de usuários e para cada cliente uma instância de serviço para rodar o server e ou, implementar camadas para cada cliente.

Por fim, melhorar o conceito "blackbox", ou seja, encapsular tudo em uma placa e fabricar um equipamento completo. Entretanto o resultado final foi obtido.

CONCLUSÃO FINAL: Como resultado final posso dizer que gostei bastante de trabalhar neste projeto, apesar de uma primeira impressão depois que peguei o jeito com o Freescale gostei bastante e já tive muitas ideias para outros projetos, inclusive em soluções comercias em desenvolvimento de produtos em minhas

empresas, como a de automação residencial CloudHome. Gostaria muito de agradecer a oportunidade do pessoal do Embarcados / Freescale por esta oportunidade.