PROJETO FINAL DE ELETRÔNICA EMBARCADA - BOTFEEDER + COMEDOURO INTELIGENTE

Daniel Auler

10/0048978
Programa de Graduação em Engenharia Eletrônica, Faculdade Gama
Universidade de Brasília
Gama, DF, Brasil
email: danielauler7@gmail.com

I. RESUMO

Este projeto trata-se de um Comedouro para animais domésticos com interface através de um bot no telegram. O comedouro é controlado por um MSP430G2553ET utilizando um servo motor MG996R para abrir e fechar a escotilha. O MSP430 por sua vez é controlado por um Raspberry Pi Zero W conectado a internet que faz verificações para ativar ou não as ações do MSP430.

II. INTRODUÇÃO

Segundo o Instituto Pet Brasil em conjunto ao IBGE, o Brasil possui uma população próxima a 78 milhões de animais de estimação entre cães e gatos [1]. Com a rotina cada vez mais corrida, os donos se vêem obrigados a utilizar aplicativos [2] ou contratar pessoas para que cuidem de seus pets durante o período que estiver ausente. Assim surgiram algumas soluções no mercado cujo objetivo visa manter garantir que o pet não ficará sem alimento durante o período de ausência do dono [3] porém todas partem do pressuposto que o pet irá comer toda a ração o que na maioria das vezes acontece com cães, porém a recíproca não é verdadeira para gatos.
Se o comedouro apenas encher a tigela de ração deliberadamente, pode ocasionar um transbordo, gerando um desperdício da ração ou atraindo formigas para o ambiente, o que pode gerar problemas gástricos aos pets [4].

III. DESENVOLVIMENTO

Para solucionar o problema levantado foi criado um comedouro eletrônico assim como as outras soluções do mercado, porém com um diferencial de analisarmos utilizando uma câmera, a quantidade de ração que ainda não foi consumida, permitindo assim que o usuário decida o quanto de ração deverá ser adicionado ao pote ou se simplesmente deseja cancelar a operação. A escolha da interface ter sido a API de bots do telegram, foi pelo fato de que a criação de um app cria uma barreira (download e instalação) ao usuário, além de dificultar o seu acesso já que o telegram pode ser acessado via browser.

III-A. Descrição do Hardware

Como hardware, utilizou-se para a solução apenas matérias de baixo custo já que visa-se a criação de um produto competitivo no mercado.

- Servo Motor MG 996R

Fig. 1. Servo motor utilizado retirado do site da fabricante

Entre os atuadores temos um motor bem especial. Os servomotores, também chamados de servos, são muito utilizados quando o assunto é robótica. De forma simplificada, um servomotor é um motor na qual podemos controlar sua posição angular através de um sinal PWM. [5].
Este servo motor especificamente possui mais torque que o mais comum SG90, o que garante que será capaz

de travar o fluxo de ração mesmo com o peso da ração com o reservatório cheio.

- MSP430

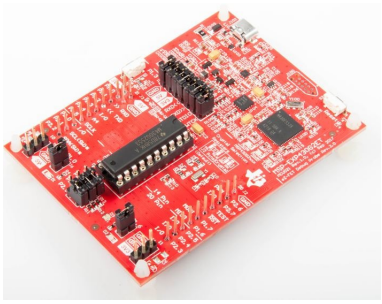

Fig. 2. MSP430G2553ET retirado do site da Texas Instruments

Os MSP430 são microcontroladores RISC de 16 bits voltados para aplicações de baixo consumo de energia e são fabricados pela Texas Instruments. A CPU dos MSP430 possui um conjunto de apenas 51 instruções (27 físicas e 24 emuladas) e um total de 16 registradores de 16 bits. Algumas das principais características do MSP430 é a flexibilidade no que diz respeito à sua arquitetura das portas. Estas possuem funções de entrada, saída e uma função especial de hardware como USARTs, DACs, etc [6].

- Raspberry Pi Zero W

Fig. 3. Raspberry Pi Zero W retirado do site da Fundação Raspberry Pi

Raspberry Pi é uma série de computadores de placa única do tamanho reduzido, que se conecta a um monitor de computador ou TV, e usa um teclado e um mouse padrão, desenvolvido no Reino Unido pela Fundação Raspberry Pi. Todo o hardware é integrado numa única placa. Este em questão, trata-se do Raspberry mais barato do mercado atualmente e já possui Wifi integrado, permitindo o acesso a internet sem fio e sem outros periféricos. As especificações completas do Pi Zero W são: processador single-core de 1 GHz, 512 MB de RAM, slot para cartão microSD, porta Mini-HDMI com saída de 1080p a 60 fps, porta GPIO de 40 pinos, saída para vídeo composto e duas portas Micro-USB.
Ele ainda traz o conector CSI dedicado para câmera, também incluso na versão 1.3 do Pi Zero, além, das conexões Wi-Fi 802.11n e Bluetooth 4.0.

- Webcam Logitech C270

Fig. 4. Logitech C270 retirado do site da fabricante

Está Webcam é um modelo de entrada da Logitech. Possui até 3 megapixels, mais do que o suficiente para efetuarmos o OCR.

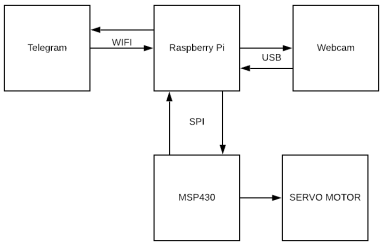

Fig. 5. Diagrama de blocos da aplicação

Foi utilizado o protocolo SPI para a comunicação entre o Raspberry Pi e o MSP430 já que precisávamos de informações básicas passando de um para o outro. Para a aplicação o Raspberry Pi apenas passa a duração que o MSP430 deverá manter a escotilha aberta, influenciando na quantidade de ração colocada na tigela.

III-B. Descrição de Software

Para escrever o software foi utilizado o seguinte diagrama de blocos:

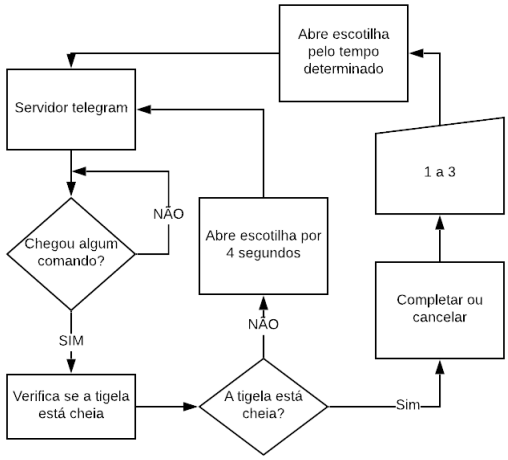

Fig. 6. Diagrama de blocos da aplicação

- Bot Telegram
  Com o objetivo de conectar o comedouro ao usuário, criou-se um servidor em C++ no raspberry utilizando da biblioteca tgbot cpp. Esta, trata toda comunicação via API com o bot do telegram e trata as mensagens afim de usar apenas os comandos pré determinados. Para criar o bot do telegram utilizamos um bot disponibilizado pelo próprio Telegram chamado botFather que automatiza o processo e gera o token que deve ser utilizado como chave de acesso ao bot via API.
  O bot telegram recebe o comando para alimentar o pet e chama a função VerifyBowl que nada mais é do que uma função que tira um foto utilizando a webcam e a envia para a função verifyFood que se encontra em outro arquivo.
  1) Função ScanBowl
     Esta função recebe a foto e funciona em três etapas: segmentação da imagem, tratamento do limiar e o tratamento da imagem em si. O conceito de segmentação de imagem foi desenvolvido por meio do algoritmo K-Means, enquanto que o limiar foi implementado usando o algoritmo Hough Circle Transform e o tratamento da imagem foi realizado usando o conceito de histograma.
     O algoritmo de K-means é um processo iterativo que possui como objetivo agrupar os dados e compartilhá-los em grupos. Um dos recursos importantes que é fornecido pela versão do OpenCV é a possibilidade que o usuário tem para personalizar o grupo para melhor atender a sua funcionalidade no projeto. Um programador pode implementar o código especificando quais iterações irão percorrer, bem como escolher quais grupos serão separados para o tratamento de image. Com isso ele pode decidir que valores associados ao RGB irá servir de critério para agrupar os dados.
     No caso do algoritmo Hough Circle Transform foi utilizado para encontrar círculos em uma imagem. A partir disso, esse círculo é usado para criar uma máscara que foi implementada para separar o que á a cor da tigela em relação a cor da ração, de modo a viabilizar o tratamento da imagem.
     Com esse código é possível determinar se há ração na tigela medindo os valores de intensidade da tigela. Por usa vez, se houver ração, isso indicará que existe uma menor intensidade de cor em relação a tigela vazia. E por fim foi utilizado um tratamento de imagens para reduzir significativamente o número de cores da imagem, além de fazer as bordas dos objetos em análise se destacarem.
  2) CMakeList
     Como foi testado diversas vezes, utilizou-se de um script que chamava o CMakeList.txt para efetuar as builds de todos os arquivos chamando as bibliotecas corretamente.
  3) Main
     Na função main é onde temos os setups das bibliotecas do wiringPi para efetuar-se as comunicações via SPI além de ser onde trata-se os comandos recebidos. O comando /alimentar chama a rotina padrão especificada acima. Já para podermos efetuar uma rotina sem o tratamento de imagens criamos o comando /semVerificarAlimentar que envia o valor 4 via SPI para o MSP430, significando que o mesmo deverá manter a escotilha aberta por 4 segundos, o tempo padrão para encher a tigela de ração.

- MSP430
  Para o MSP430 utilizamos um código baseado no código apresentador pelo professor Diogo Caetano da disciplina de Sistemas Embarcados que recebe os dados do Raspberry e envia para a função Feeder o tempo informado que deverá ser mantido o servo motor na posição de -90º.
  Para otimizar o processo de alimentação, criou-se uma para enviar as informações em assembly retornando os dados de acordo com o que é recebido.

IV. RESULTADOS

Apesar das dificuldades de adaptação ás peculiaridades da linguagem C++, o projeto se mostrou bem sólido em executar seu objetivo principal de alimentar o pet com comandos do telegram. Importante salientar alguns problemas encontrados na execução final: 1 - O sistema só faz a verificação da imagem uma vez, aparentemente a função wait() do openCV está aguardando o fim da operação e fica travado nesse estágio da aplicação; 2 - Se houver sombra na tigela de ração o algoritmo poderá identificar como se houvesse mais ração do que realmente há, esse problema foi o motivador da mensagem contento a foto tirada informando ao usuário que se for do desejo dele, o sistema poderá completar a tigela com ração; 3 - Algumas vezes a função de tirar foto dá uma exceção e acaba encerrando o servidor, deveria ter sido tratado esse tipo de exceção. Para efetuarmos os testes práticos, criou-se uma estrutura MVP:

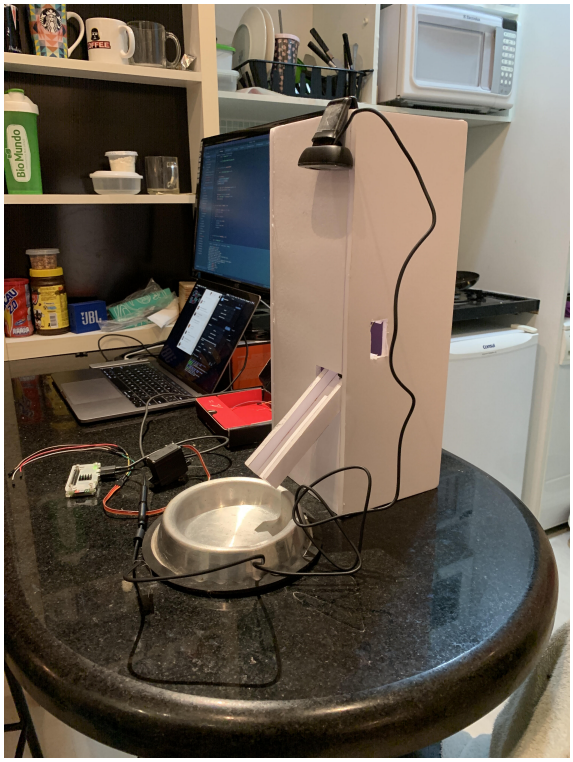

Fig. 7. Estrutura criada para testar o projeto

Assim podemos testar os comandos do bot para verificarmos se ele estava respondendo de acordo:

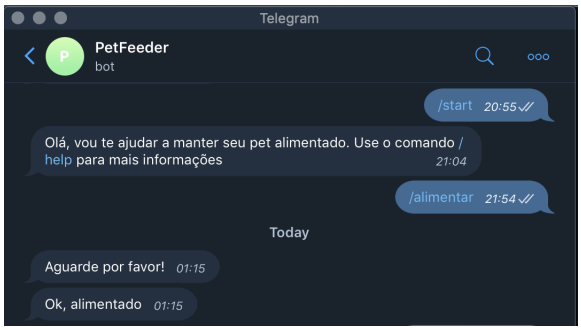

Fig. 8. Bot telegram

Ao enviar o comando alimentar com a tigela cheia, temos o seguinte comportamento:

Fig. 9. Bot telegram

V. CONCLUSÃO

Dado o exposto podemos verificar que de fato o comedouro possui uma importância e consegue com sucesso suprir as dificuldades em manter o pet sempre com alimento disponível, mesmo quando o seu dono não se encontra em casa. A solução da webcam com o tratamendo de imagens se mostrou uma boa estratégia e apesar do projeto não estar funcionando com um desempenho alto, para um MVP pode se considerar que foi entregue um produto que demonstra

as intenções e o caminho que o projeto deverá tomar.
Como aprimoramento sugere-se manter o servidor do raspberry alocado em um servido de cloud dedicado, com um poder de processamento melhor para reduzir o tempo do tratamento da imagem e permitir que o bot tenha uma responsividade mais instantânea.

VI. REFERENCIAS

[1] Censo Pet: 139,3 milhões de animais de estimação no Brasil", Assessoria de Imprensa Instituto Pet Brasil. [Online] Available: http://institutopetbrasil.com/imprensa/censo-pet-1393-milhoes-de-animais-de-estimacao-no-brasil/ . Acessado em 08/12/2019

[2] App ajuda donos de cães a encontrar cuidadores durante as férias de julho", Danilo Martins. [Online] Available: http://g1.globo.com/distrito-federal/noticia/2016/07/app-ajuda-donos-de-caes-encontrar-cuidadores-durante-ferias-de-julho.html . Acessado em 08/12/2019

[3] Os 5 modelos de comedouros para cachorro para facilitar sua vida", Nathalia Perone. [Online] Available: https://mytxai.pet/blog/dicas/os-5-modelos-de-comedouro-para-cachorro-para-facilitar-sua-vida/ . Acessado em 08/12/2019

[4] Não deixe as formigas atacarem a comida do seu cachorro", BitCão e BitGato. [Online] Available: https://www.bitcao.com.br/blog/nao-deixe-as-formigas-atacarem-a-comida-do-seu-cachorro/ . Acessado em 08/12/2019

[5] O que é Servomotor? Controlando um Servo com Arduino", Allan Mota. [Online] Available: https://portal.vidadesilicio.com.br/o-que-e-servomotor/ . Acessado em 08/12/2019

[6] MSP430 ultra-low-power sensing and measurement MCUs". [Online] Available: http://www.ti.com/microcontrollers/msp430-ultra-low-power-mcus/overview.html. Acessado em 08/12/2019

[7] GUINGO, B. C.; RODRIGUES, R. J.; THOME ANTONIO, C. G. Automatic identification for automotives vehicles plates. 2014.

APÊNDICE A
CÓDIGOS RASPBERRY

```
#include <stdio.h>
#include <stdlib.h>
#include <iostream>
#include <string>
#include <signal.h>
#include <thread>
#include <tgbot/tgbot.h>
#include <unistd.h>
#include <wiringPi.h>
#include <wiringPiSPI.h>
```

```
#include "scanBowl.h"

#define SERVO 1

using namespace std;
using namespace TgBot;

int spi_fd;
int feed_default;

void takePic()
{
    cout << "Tirando foto" << endl;

    system("fswebcam -r 320x240 foto_img.jpg");
}

bool verifyBowl(string photoFilePath)
{
    cout << "Verificando tigela" << endl;

    bool haveFood;
    system("rm -rf foto_img.jpg");
    thread takePhoto(takePic);
    takePhoto.join();

    char *tab2 = new char[photoFilePath.length()
    + 1];
    strcpy(tab2, photoFilePath.c_str());

    haveFood = verifyFood(tab2);
    cout << "tem ração: " << haveFood << endl;
    return haveFood;
}

int main()
{
    const string photoFilePath = "foto_img.jpg";
    const string photoMimeType = "image/jpeg";

    if (wiringPiSetup() == -1)
    {
        puts("Erro em wiringPiSetup().");
        return -1;
    }
    spi_fd = wiringPiSPISetup(0, 500000);
    if (spi_fd == -1)
    {
        puts("Erro abrindo a SPI. Garanta que ela
     nao");
        puts("esteja sendo usada por outra
    aplicacao.");
        return -1;
    }
    Bot bot("792286575:AAG0puZ_3PvXAwh6ckXb5r4-
    cOfn56sgibU");

    InlineKeyboardMarkup::Ptr keyboard(new
    InlineKeyboardMarkup);
    InlineKeyboardMarkup::Ptr keyboard2(new
    InlineKeyboardMarkup);
    vector<InlineKeyboardButton::Ptr> row0;
    InlineKeyboardButton::Ptr checkButton(new
    InlineKeyboardButton);
    InlineKeyboardButton::Ptr checkButton2(new
    InlineKeyboardButton);
    InlineKeyboardButton::Ptr checkButton3(new
    InlineKeyboardButton);
    InlineKeyboardButton::Ptr checkButton4(new
    InlineKeyboardButton);

```

```
    vector<InlineKeyboardButton::Ptr> row1;
    vector<InlineKeyboardButton::Ptr> row2;
    vector<InlineKeyboardButton::Ptr> row3;

    checkButton->text = "pouco";
    checkButton->callbackData = "nivel 1";
    row0.push_back(checkButton);
    keyboard->inlineKeyboard.push_back(row0);
    checkButton2->text = "médio";
    checkButton2->callbackData = "nivel 2";
    row0.push_back(checkButton2);
    keyboard->inlineKeyboard.push_back(row0);

    checkButton3->text = "bastante";
    checkButton3->callbackData = "nivel 3";
    row1.push_back(checkButton3);
    keyboard->inlineKeyboard.push_back(row1);

    checkButton4->text = "Cancelar";
    checkButton4->callbackData = "cancelar";
    row1.push_back(checkButton4);
    keyboard->inlineKeyboard.push_back(row1);

    bot.getEvents().onCommand("start", [&bot](
    Message::Ptr message) {
        bot.getApi().sendMessage(message->chat->
    id, "Olá, vou te ajudar a manter seu pet
    alimentado. Use o comando /help para mais
    informações");
    });

    bot.getEvents().onCommand("
    semVerificarAlimentar", [&bot](Message::Ptr
    message) {
        unsigned char time_default = 4;
        wiringPiSPIDataRW(0, &time_default, 1);
        printf("MSP430_return = %d\n",
    time_default);
        sleep(1 + time_default / 2);
        string response = "Ok, alimentado";
        bot.getApi().sendMessage(message->chat->
    id, response);
    });

    bot.getEvents().onCommand("alimentar", [&bot,
     &photoFilePath, &photoMimeType, &keyboard](
    Message::Ptr message) {
        bot.getApi().sendMessage(message->chat->
    id, "Aguarde por favor!");
        bool existencia = verifyBowl(
    photoFilePath);
        if (!existencia)
        {
            unsigned char time_default = 4;
            wiringPiSPIDataRW(0, &time_default,
    1);
            printf("MSP430_return = %d\n",
    time_default);
            sleep(1 + time_default / 2);
            string response = "Ok, alimentado";
            bot.getApi().sendMessage(message->
    chat->id, response);
        }
        else
        {
            string response = "Se quiser que eu
    complete o pote, basta me dizer o quão cheio
    ele já está. Ou pode cancelar!";
            bot.getApi().sendPhoto(message->chat
    ->id, InputFile::fromFile(photoFilePath,
    photoMimeType), "A tigela ainda está cheia!")
    ;
            bot.getApi().sendMessage(message->
    chat->id, response, false, 0, keyboard, "
    Markdown");
        }
    });

    bot.getEvents().onCallbackQuery([&bot](
    CallbackQuery::Ptr query) {
        if (StringTools::startsWith(query->data,
    "nivel"))
        {
            unsigned char value = query->data.
    back();
            cout << "value selected is: " <<
    value << endl;
            if ((value < '0') || (value > '5'))
            {
                puts("Valor invalido");
                bot.getApi().sendMessage(query->
    message->chat->id, "Valor invalido");
            }
            else
            {
                wiringPiSPIDataRW(0, &value, 1);
                printf("MSP430_return = %d\n",
    value);
                sleep(1 + value / 2);
                bot.getApi().sendMessage(query->
    message->chat->id, "Ok, alimentado!");
            }
            puts("");
        }
    });

    bot.getEvents().onCallbackQuery([&bot](
    CallbackQuery::Ptr query) {
        if (StringTools::startsWith(query->data,
    "cancel"))
        {
            string response = "ok";
            bot.getApi().sendMessage(query->
    message->chat->id, response);
        }
    });

    signal(SIGINT, [](int s) {
        printf("SIGINT got\n");
        exit(0);
    });

    try
    {
        printf("Bot username: %s\n", bot.getApi()
    .getMe()->username.c_str());
        bot.getApi().deleteWebhook();
        TgLongPoll longPoll(bot);
        while (true)
        {
            printf("Long poll started\n");
            longPoll.start();
        }
    }
    catch (TgException &e)
    {
        printf("error: %s\n", e.what());
    }
    return 0;
}
```

APÊNDICE B
FUNÇÃO VERIFYBOWL.CPP

```cpp
#include "scanBowl.h"

using namespace cv;
using namespace std;

bool verifyFood(char *filename){

 bool existe_racao;
 int pixeis_racao;
 int pixeis_tigela;

 Mat fonte = imread(filename, 1);
 Mat amostras(fonte.rows * fonte.cols, 3, CV_32F);
 for (int y = 0; y < fonte.rows; y++)
  for (int x = 0; x < fonte.cols; x++)
   for (int z = 0; z < 3; z++)
    amostras.at<float>(y + x*fonte.rows, z) = fonte.at<Vec3b>(y, x)[z];

 int clusterCount = 4;
 Mat labels;
 int tentativas = 5;
 Mat centros;
 kmeans(amostras, clusterCount, labels, TermCriteria(CV_TERMCRIT_ITER | CV_TERMCRIT_EPS, 10000, 0.0001), tentativas, KMEANS_PP_CENTERS, centros);

    Mat new_image(fonte.size(), fonte.type());

 for (int y = 0; y < fonte.rows; y++)
  for (int x = 0; x < fonte.cols; x++){
   int cluster_idx = labels.at<int>(y + x*fonte.rows, 0);
   new_image.at<Vec3b>(y, x)[0] = centros.at<float>(cluster_idx, 0);
   new_image.at<Vec3b>(y, x)[1] = centros.at<float>(cluster_idx, 1);
   new_image.at<Vec3b>(y, x)[2] = centros.at<float>(cluster_idx, 2);
  }

 // imshow("clustered image", new_image);
 imwrite("clustered_image.jpg", new_image);
 Mat img = imread("clustered_image.jpg", 0);
 // Contabilizar quantos pixeis estao associados ao label da racao,
 // e quantos correspondem ao label da tigela
 pixeis_racao = 0;
 pixeis_tigela = 0;
 for (int y = 0; y < fonte.rows; y++){
  for (int x = 0; x < fonte.cols; x++){
   int cluster_idx = labels.at<int>(y + x*fonte.rows, 0);
   if (cluster_idx == 1){ // 1 usar o label da racao
    pixeis_racao++;
   }
   if (cluster_idx == 0){ // 2 usar o label da tigela
    pixeis_tigela++;
   }
  }
 }
 cout<<pixeis_racao<<endl;
 cout<<pixeis_tigela<<endl;
 if ((float)((1.0 * pixeis_racao) / (1.0 * pixeis_tigela) > 0.2)){ // esta fazendo a conversao

            // para float
  existe_racao = true;
 }
 else
  existe_racao = false;

 Mat cimg;
 Mat thresh = Mat::zeros(img.size(), img.type());
 medianBlur(img, img, 5);
 cvtColor(img, cimg, COLOR_GRAY2BGR);
 std::vector<Vec3f> circles;

 HoughCircles(img, circles, HOUGH_GRADIENT, 1, 500, 100, 30, 200, 450); //
 //change the last two parameters
 //(min_radius & max_radius) to detect larger circles

 for (size_t i = 0; i < circles.size(); i++){
  Vec3i c = circles[i];
  ellipse(cimg, Point(c[0], c[1] * 3 / 4), Size(c[2], c[2] * 3 / 4), 0, 0,
  360, Scalar(0, 255, 0), 3, LINE_AA);
  ellipse(thresh, Point(c[0], c[1] * 3 / 4), Size(c[2], c[2] * 3 / 4), 0, 0,
  360, Scalar(255, 255, 255), -1, LINE_AA);
  circle(cimg, Point(c[0], c[1]), 2, Scalar(0, 255, 0), 3, LINE_AA);
 }

 //imshow("detected circles", cimg);
 //imshow("threshold img", thresh);
 Mat hist;
 int histSize = 256;
 float range[] = { 0, 256 };
 const float* histRange = { range };
 calcHist(&img, 1, 0, thresh, hist, 1, &histSize, &histRange, true, false);

 // Draw hist
 int hist_w = 512; int hist_h = 400;
 int bin_w = cvRound((double)hist_w / histSize);
 Mat histImage(hist_h, hist_w, CV_8UC3, Scalar(0, 0, 0));

 /// Normalize the result to [ 0, histImage.rows ]
 normalize(hist, hist, 0, histImage.rows, NORM_MINMAX, -1, Mat());

 /// Draw for each channel
 for (int i = 1; i < histSize; i++){
  line(histImage, Point(bin_w*(i - 1), hist_h - cvRound(hist.at<float>(i - 1))),
  Point(bin_w*(i), hist_h - cvRound(hist.at<float>(i))),
  Scalar(255, 0, 0), 2, 8, 0);
 }

 /// Display
 //namedWindow("calcHist Demo", CV_WINDOW_AUTOSIZE);
 //imshow("calcHist Demo", histImage);
 std::cout << "Mean intensity is: " << mean(hist) << std::endl;
 waitKey();
```

```
 return existe_racao;
}
```

APÊNDICE C
FUNÇÃO VERIFYBOWL.H

```
#ifndef SCANBOWL_h     // To make sure you don't
    declare the function more than once by
    including the header multiple times.
#define SCANBOWL_h
#include "opencv2/imgproc/imgproc.hpp"
#include "opencv2/highgui/highgui.hpp"
#include <iostream>

bool verifyFood(char *filename);

#endif
```

APÊNDICE D
CMAKELIST.TXT

```
cmake_minimum_required(VERSION 2.8.4)
project(ComedouroBot)

set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -std=c
    ++11 -lwiringPi -Wall")
set(Boost_USE_MULTITHREADED ON)

find_package(OpenCV REQUIRED)
find_package(Threads REQUIRED)
find_package(OpenSSL REQUIRED)
find_package(Boost COMPONENTS system REQUIRED)
find_package(CURL)
include_directories(/usr/local/include ${
    OPENSSL_INCLUDE_DIR} ${Boost_INCLUDE_DIR} ${
    OpenCV_INCLUDE_DIRS})
if (CURL_FOUND)
    include_directories(${CURL_INCLUDE_DIRS})
    add_definitions(-DHAVE_CURL)
endif()

add_executable(ComedouroBot main.cpp scanBowl.cpp
    )

target_link_libraries(ComedouroBot /usr/local/lib
    /libTgBot.a ${CMAKE_THREAD_LIBS_INIT} ${
    OPENSSL_LIBRARIES} ${Boost_LIBRARIES} ${
    CURL_LIBRARIES} ${OpenCV_LIBS})
```

APÊNDICE E
MSP430

```
#include <msp430g2553.h>

#define MISO BIT1
#define MOSI BIT2
#define SCLK BIT4
#define LED1 BIT0
#define SERVO BIT6
#define LEDS (LED1|LED2)
#define DLY1 0x6000
#define DLY2 0x3000

#define MCU_CLOCK              1100000
#define PWM_FREQUENCY          46      // In Hertz,
    ideally 50Hz.

#define SERVO_STEPS            180     // Maximum
    amount of steps in degrees (180 is common)
#define SERVO_MIN              700     // The
    minimum duty cycle for this servo
#define SERVO_MAX              3000    // The
    maximum duty cycle

unsigned int PWM_Period      = (MCU_CLOCK /
    PWM_FREQUENCY);  // PWM Period
unsigned int PWM_Duty        = 0;
                // %

void Atraso(volatile unsigned int x)
{
    while(x--);
}

void delay_ms(unsigned int delay)
{
    while (delay--)
    {
        __delay_cycles(1000);
    }
}

void Send_Data(volatile unsigned char c)
{
    __asm__(
        ".dwcfi  cfa_offset, 2"
       ".dwcfi  save_reg_to_mem, 16, -2"
          "  SUB.W     #2,SP                  ; []
     "
        ".dwcfi  cfa_offset, 4"
    "$C$DW$33    .dwtag  DW_TAG_variable"
       " .dwattr $C$DW$33, DW_AT_name("c")"
       " .dwattr $C$DW$33, DW_AT_TI_symbol_name("
    c")"
       " .dwattr $C$DW$33, DW_AT_type(*$C$DW$T$23
    )"
       " .dwattr $C$DW$33, DW_AT_location[
    DW_OP_breg1 0]"

        "    MOV.B     r12,0(SP)              ; []
     |36|  "
    );
}

void Feeder(unsigned char pino, const int
    segundos, unsigned int tempo)
{
    unsigned int servo_stepval;
    unsigned int servo_lut[ SERVO_STEPS+1 ];
    unsigned int i;
    // Move forward toward the maximum step value
    for (i = 90; i < SERVO_STEPS; i++) {
        TACCR1 = servo_lut[i];
        __delay_cycles(20000);
    }
    delay_ms(segundos);
    // Move backward toward the minimum step
    value
    for (i = SERVO_STEPS; i > 90; i--) {
        TACCR1 = servo_lut[i];
        __delay_cycles(20000);
    }
//    Atraso(tempo);
}

void servo_setup(){
    unsigned int servo_stepval, servo_stepnow;
    unsigned int servo_lut[ SERVO_STEPS+1 ];
    unsigned int i;

    // Calculate the step value and define the
    current step, defaults to minimum.
```

```
77      servo_stepval    = ( (SERVO_MAX - SERVO_MIN) /
         SERVO_STEPS );
78      servo_stepnow    = SERVO_MIN;
79
80      // Fill up the LUT
81      for (i = 0; i < SERVO_STEPS; i++) {
82          servo_stepnow += servo_stepval;
83          servo_lut[i] = servo_stepnow;
84      }
85
86      TACCTL1 = OUTMOD_7;              // TACCR1
        reset/set
87      TACTL   = TASSEL_2 + MC_1;       // SMCLK,
        upmode
88      TACCR0  = PWM_Period-1;          // PWM Period
89      TACCR1  = PWM_Duty;              // TACCR1 PWM
        Duty Cycle
90      P1DIR   |= SERVO;
91
92  }
93
94  int main(void)
95  {
96      servo_setup();
97      WDTCTL = WDTPW + WDTHOLD;
98      BCSCTL1 = CALBC1_1MHZ;
99      DCOCTL = CALDCO_1MHZ;
100     P1SEL2 = MOSI+MISO+SCLK;
101     P1SEL = MOSI+MISO+SCLK+SERVO;
102     UCA0CTL1 = UCSWRST + UCSSEL_3;
103     UCA0CTL0 = UCCKPH+UCMSB+UCMODE_0+UCSYNC;
104     UCA0CTL1 &= ~UCSWRST;
105     Send_Data(10);
106     IE2 |= UCA0RXIE;
107     _enable_interrupts();
108     _BIS_SR(LPM0_bits + GIE);
109     return 0;
110 }
111
112 #pragma vector = USCIAB0RX_VECTOR
113 __interrupt void Receive_Data()
114 {
115     volatile unsigned char segundos = UCA0RXBUF;
116     P1OUT ^= LED1;
117     switch (segundos)
118     {
119         case 3:
120            Feeder(SERVO, 3000000, DLY1);
121            Send_Data(segundos);
122         case 2:
123            Feeder(SERVO, 2000000, DLY1);
124            Send_Data(segundos);
125         case 1:
126            Feeder(SERVO, 1000000, DLY1);
127            Send_Data(segundos);
128         default:
129            Feeder(SERVO, 4000000, DLY1);
130            Send_Data(segundos);
131     }
132
133     IFG2 &= ~UCA0RXIFG;
134     P1OUT ^= LED1;
135 }
```